UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

OTIMIZAÇÃO DINÂMICA EM TEMPO REAL: ARQUITETURA DE SOFTWARE, DIAGNÓSTICO E ANÁLISE DE INVIABILIDADES

TESE DE DOUTORADO

EUCLIDES ALMEIDA NETO

PORTO ALEGRE, RS
Maio, 2011

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

OTIMIZAÇÃO DINÂMICA EM TEMPO REAL: ARQUITETURA DE SOFTWARE, DIAGNÓSTICO E ANÁLISE DE INVIABILIDADES

EUCLIDES ALMEIDA NETO

Tese de Doutorado apresentada como requisito parcial para obtenção do título de Doutor em Engenharia.

Área de Concentração: Pesquisa e Desenvolvimento de Processos

Subárea: Projetos, Simulação, Controle e Otimização de Processos Químicos e Biotecnológicos

Orientador:
Prof. Argimiro Resende Secchi, D.Sc.

Co-orientadores no exterior:
Prof. Dr. Lorenz T. Biegler (Carnegie Mellon University, Pittsburgh, PA - Estados Unidos)

Prof. Dr. Wolfgang Marquardt (RWTH Aachen University, Aachen - Alemanha)

PORTO ALEGRE, RS
Maio, 2011

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

A Comissão Examinadora, abaixo assinada, aprova a Tese *Otimização Dinâmica em Tempo Real: Arquitetura de Software, Diagnóstico e Análise de Inviabilidades*, elaborada por **Euclides Almeida Neto**, como requisito parcial para obtenção do título de Doutor em Engenharia.

Comissão Examinadora:

_____
Prof. Dr. Rafael de Pelegrini Soares

_____
Eng. Dr. Oscar Rotava

_____
Prof. Dr. Luís Gustavo Soares Longhi

Agradecimentos

Ao Prof. Dr. Argimiro R. Secchi pela grande contribuição para meu crescimento no assunto e pela maior paciência ainda com minhas convicções (cabeça dura).

À minha querida esposa Vanda e meus filhos Diego e Alessandra, pelo apoio e compreensão nos meus períodos de recolhimento.

Ao Prof. Dr. Larry Biegler (meu grande pai) pelas afortunadas discussões, contribuições e amizade na minha passagem pela Universidade de Carnegie Mellon em Pittsburgh.

Ao Prof. Dr. Wolfgang Marquardt pela orientação, boas sugestões e acolhimento na minha passagem pelo Lehrstuhl für Prozesstechnik (*PT*) - Aachener Verfahrenstecnik (*AVT*) da Rheinisch-Westfälische Technische Hochschule (*RWTH*) – Aachen.

Ao grande Rafa pela enorme colaboração para meu crescimento e agradeço pela sua grande amizade.

Às amigas Paulinha e Nina pelas belas discussões e trabalho em conjunto, e aos demais colegas do *DEQUI*/*UFRGS* pela excelente convivência no período de trabalho.

Aos amigos Rotava e Lincoln, pelas ajudas nas revisões de artigos e apoio no meu trabalho.

Ao pessoal do *LADES* e G-130, pela ótima convivência por todos estes anos. E mais recentemente à Danielle, pelas suas discussões e ajuda nesta reta final.

À PETROBRÁS S.A pelo suporte durante parte do meu doutorado no Departamento de Engenharia Química (*DEQUI*) da *UFRGS*.

Resumo

As constantes pressões por redução de margem de lucro, melhoria de qualidade de produtos e segurança operacional que a indústria de processamento vem sofrendo, tem levado as mesmas a utilizarem ferramentas especializadas de otimização de processos. Nas décadas de 1980 e 1990, esta indústria investiu fortemente na utilização de ferramentas de otimização estacionária em tempo real (*RTO*) nas formas *online* e *offline*. Esta tecnologia já atingiu o seu grau de maturidade. Porém está limitada pelas suas características estacionárias, não tendo capacidade de otimizar processos diante das perturbações freqüentes, tais como alterações de qualidade e quantidade de carga, transições decorrentes de alterações de programação de produção ou de receita de uma produção em batelada ou semi-batelada, dentre outros. Para cobrir este espaço, a otimização dinâmica em tempo real (*DRTO*) é a tecnologia adequada para reduzir a quantidade de produtos fora de especificação e otimizar o lucro operacional diante destas perturbações. Porém, esta tecnologia ainda não atingiu o seu grau de maturidade, tendo ferramentas comerciais apenas na sua versão *offline*. Com o objetivo de contribuir com a consolidação desta tecnologia, propõe-se estudar e desenvolver uma arquitetura de sistema de *DRTO* para operar nas plantas de processo, promover melhorias conceituais nesta tecnologia, e desenvolver uma ferramenta de diagnóstico e sintonia de *DRTO*. Esta estrutura é bem completa e fornece a flexibilidade necessária para uso industrial. A ferramenta de diagnóstico permite resolver os problemas que ocorrerem ao longo do uso do *DRTO*. Além disso, é apresentada uma nova metodologia de análise e solução de inviabilidades, baseada em otimização multiobjetivos aplicada à otimização dinâmica. Esta técnica pode ser utilizada tanto *online* quanto *offline*, para diagnóstico, e na solução de problemas de otimização dinâmica, conferindo mais robustez ao *DRTO*, evitando insucessos do otimizador.

Palavras chave: Otimização Dinâmica, *DRTO*, Controle Ótimo, Diagnóstico em *DRTO*, Solução de Inviabilidades.

Abstract

Considering the constant pressures for profit margins reduction, improvement of products quality and operational safety, that the processing industry has been submitted, has led them to the use of specialized tools of processes optimization. In the 1980 and 1990 decades, this industry has invested strongly in the use of stationary real-time optimization tools (*RTO*), in the online and offline forms. This technology already reached its degree of maturity, but limited to its stationary characteristics, having no ability to optimize processes due to frequent disturbances, such as quality and feed flows transitions, consequence of frequent changes in the production scheduling or recipes in batch or semi-batch operations, and others. To cover this space, the dynamic real-time optimization (*DRTO*) is the appropriate technology to reduce the off-spec production and optimize the operational profit when the process is submitted to these disturbances. However, this technology has not reached its maturity, having only commercial tools available only in yuor offline version. In order to contribute to the consolidation of this technology, it is proposed to study and develop a *DRTO* system architecture to operate in process plants, to promote a conceptual improvements in this technology, and to develop a *DRTO* diagnostic and tuning tool. This structure is quite complete and provides the flexibility required for industrial application. The diagnostic tool allows us to solve problems that occur during the use of the *DRTO* system. In addition, a new methodology for infeasibility analysis and solution in *DAOP* is proposed here, and it is based on the solution of the multiobjective dynamic optimization. This technique can be used in *online* and *offline* modes, for diagnostics and troubleshooting dynamic optimization problems, giving more robustness to the *DRTO* systems, avoiding some optimizer failures.

Sumário

Lista de Figuras

Lista de Tabelas

Esta tese reproduziu alguns trechos apresentados no meu exame de qualificação ao doutorado, submetido ao DEQUI/UFRGS em Agosto/ 2006 com o título: DESENVOLVIMENTO DE UMA FERRAMENTA DE OTIMIZAÇÃO DINÂMICA EM TEMPO REAL.

Trechos desta qualificação foram utilizados parcialmente, com o meu consentimento embora não devidamente creditado, por Otto Indio do Brasil Magalhães, em sua dissertação de mestrado intitulada "DESENVOLVIMENTO DE UM SISTEMA DE OTIMIZAÇÃO DINÂMICA EM TEMPO REAL", defendida na COPPE/UFRJ em Setembro de 2010; e também por Thiago Corrêa do Quinto, em sua dissertação de mestrado intitulada "ABORDAGEM ALGÉBRICO-DIFERENCIAL DA OTIMIZAÇÃO DINÂMICA DE PROCESSOS COM ÍNDICE FLUTUANTE", defendida na COPPE/UFRJ em Setembro de 2010

1 Introdução

Com a crescente competitividade estabelecida entre as indústrias concorrentes no mercado, regido pelas quedas nas margens de lucro e na pressão pela redução do tempo decorrido entre o pedido e a entrega de produtos de forma confiável, as unidades de processo precisam operar próximas dos seus limites, de forma segura e confiável. Além disso, o mercado consumidor exige, cada vez mais, flexibilidades por parte das unidades operacionais de forma a produzir de acordo com os desejos do cliente (foco no cliente). Estes fatores fizeram com que a indústria quebrasse certos paradigmas operacionais. Como conseqüência, a indústria de processamento passou a se interessar em operar as plantas de forma otimizada através de computadores, isto é, o problema de otimização na indústria passou a ser formulado como:

> ***"Operar a unidade de processo definindo as ações de controle que otimizem o seu desempenho, respeitando os critérios de segurança operacional, as especificações de qualidade dos produtos, as restrições operacionais e de mercado e os limites de emissões ambientais."***

Diante de tais pressões, a otimização de processos passa a ter uma grande importância na excelência operacional. Para resolver estas questões, a otimização de processos é uma técnica recomendada para atender estas fortes pressões. Usualmente, um problema de otimização pode ser escrito conceitualmente como:

Minimizar/ Maximizar *Índice de Desempenho*

Sujeito à:

Comportamento do Processo
Restrições de Processo
Restrições de Produção
Restrições de Qualidade
Restrições de Equipamentos
Restrições de Segurança
Restrições de Órgãos Reguladores

Os índices de desempenho são normalmente fatores como eficiência, produtividade, custos e lucros operacionais, dentre outros. As restrições são normalmente ligadas aos limites de projeto dos equipamentos, restrições de mercado, inventário, matéria prima e insumos, restrições de especificações de qualidade de produtos e limites de emissões impostos por órgãos reguladores.

Para atingir plenamente a excelência operacional é necessária organização empresarial onde se gerencia o processo produtivo através de indicadores de desempenho para ligar a operação da planta com os objetivos de negócio e atender os requisitos dos órgãos reguladores e de gerenciamento de riscos. Um item fundamental para que se tenha excelência operacional é a utilização de ferramentas de automação de processos e de sistema de informações. As ferramentas de otimização de processos são extremamente importantes para auxiliar na operação da planta na zona de desconforto, isto é, operar nos

limites do processo. Como vocação natural das ferramentas de otimização de processo tem-se: a flexibilidade de resolver diferentes problemas, a condução da planta para seus limites operacionais e a baixa variabilidade do processo em relação aos alvos de produção. Porém, há uma série de desafios tecnológicos para se atingir a otimização plena do processo. São desafios ligados à representação adequada dos objetivos de produção, da política de operação, da modelagem de processo, ligados à instrumentação e controle, à tecnologia de otimização e à tecnologia de monitoração e diagnóstico do processo.

Para resolver este problema, a indústria vem aprimorando as suas ferramentas de controle e otimização. Há mais de 50 anos, a comunidade cientifica, juntamente com a indústria, vem desenvolvendo e aprimorando técnicas de controle e otimização de processos. Na década de 1960, a indústria de processamento procurou utilizar os conceitos de controle ótimo apresentado por Pontryagin (1962), porém sem muitos resultados práticos. No final da década de 1970, aproveitando os conceitos de controle ótimo da década de 1960, surgiram as primeiras soluções de controle preditivo baseado em modelos (*MPC - model predictive control*). Estes controladores tiveram uma grande aceitação nas indústrias de processamento e se difundiram rapidamente. Nesta mesma época, a academia investiu esforços de seus pesquisadores em resolver os problemas de otimização estática e dinâmica. Na década de 1980, surgiram as primeiras soluções de otimização estática em tempo real (*RTO – real-time optimization*), porém somente se estabeleceram na década de 1990, devido a uma série de problemas recorrentes de implementação e de manutenção de modelos do processo. As soluções de otimização dinâmica tiveram sua evolução somente no modo *offline*, isto é, em estudos de casos. Nestes casos, são estabelecidas as condições iniciais do processo e um determinado cenário a ser otimizado. As soluções de otimização dinâmica em tempo real (*DRTO - dynamic real-time optimization*) e de controle preditivo não linear (*NMPC - nonlinear model predictive control*) somente passaram a ter um interesse efetivo pela indústria de processamento a partir do ano 2000, depois que houve melhorarias de robustez, desempenho e precisão dos métodos de otimização e dos algoritmos de *NLP* (*nonlinear programming*). Até então o interesse da indústria se restringia à simulação dinâmica, mas hoje se percebe um interesse em sua otimização. Isto justifica o fato de ainda não haver uma solução comercial para *DRTO*.

Há uma grande gama de aplicações de soluções de simulação/otimização dinâmica, podendo se destacar as seguintes:

- Determinação de condições ótimas de operação
- Análise e otimização de partidas e paradas de unidades
- Treinamento de operadores em tempo real ou acelerado
- Geração de modelos lineares para o controle avançado
- Calibração de modelos com dados experimentais
- Sintonia ótima de controladores *PID*
- Síntese e análise de estratégias de controle
- Monitoração de desempenho do processo
- Análise de sensibilidade do processo a perturbações
- Controle ótimo
- Planejamento de produção
- Simulação de situações anormais e procedimentos de segurança

Cabe lembrar que a utilização de técnicas de otimização é precedida de dois fatores importantes: ter um potencial de ganho econômico e esforços de modelagem e engenharia com retorno econômico. Isto posto, normalmente é recomendável fazer estudo de viabilidade técnico-econômico da aplicação de uma tecnologia de otimização.

1.1 Motivação

Com o grau de maturidade alcançado com os *MPC's* e *RTO's*, os rigores na operação das unidades de processo vem aumentando sistematicamente. Hoje os recursos oferecidos pelos controladores preditivos e otimizadores estáticos estão se esgotando. Diante das perturbações cada vez mais freqüentes nas matérias primas, programações de produção e transições operacionais (principalmente em processos em batelada ou semi-batelada), faz com que a solução de *DRTO* seja atraente por parte das indústrias de processamento.

Cada vez mais a indústria de processos em bateladas e semi-bateladas estão pressionadas a executar suas operações de forma otimizada. Além disso, indústrias de processamento contínuo necessitam executar partidas, paradas e transições de operação de forma otimizada, reduzindo os tempos dessas operações e minimizando a quantidade de insumos e produtos fora de especificação. Também é desejo da indústria que se tenha uma integração adequada entre as camadas de controle e otimização de processos com a programação de produção. Com isso, o problema de otimização passa a incluir receitas, onde a tancagem e expedição passam a fazer parte do problema de otimização. Neste caso, um processo contínuo passa a assumir características de processos semi-batelada.

Acredita-se que, no futuro, será possível realizar as atividades de controle avançado, otimização em tempo real e programação de produção local (*scheduling*) em uma só camada, e integrado com a camada de planejamento e programação de produção da planta como um todo.

Constatou-se que não há produto comercial de *DRTO* disponível no mercado. Há somente iniciativas feitas sob medida pela academia. Percebe-se também a existência de *NMPC's* e *BMPC's* (*batch model predictive control*), mas estes pacotes não executam receitas genéricas de forma natural e otimizada. Atualmente, para executar bateladas de forma otimizada é necessário realizar a otimização *offline* e implementar a receita batelada-a-batelada (rodada-a-rodada) ou através da implementação de estratégias de controle auto-otimizantes. Por outro lado, os *BMPC's* são normalmente customizados e aplicáveis às bateladas com receitas fixas (repetitivas).

A otimização dinâmica em tempo real é uma área particularmente interessante, pois representa um desafio e um salto nas soluções atuais de otimização em tempo real. Há muitos trabalhos e vários *softwares* que resolvem o problema de otimização dinâmica, porém não em malha fechada e integrada com a planta real.

Portanto, percebe-se a necessidade da construção de um sistema de *DRTO* que seja aplicável a processos químicos, principalmente no ramo de petróleo e petroquímica. Esta ferramenta deverá ser genérica e configurável, de forma que possa ser aplicável a diferentes naturezas de processos (em batelada/semi-batelada e contínuos).

No âmbito do processamento de petróleo, identificam-se imediatamente oportunidades de otimização dinâmica aplicada à operação de unidades de coqueamento retardado (tambor de coque, fracionadora principal e área fria), na operação de torre deisobutanizadora de unidades de alcoilação (na transição de carga saturada/olefínica e vice-versa), e na operação de sistemas de mistura em linha e tanque de diesel e gasolina. No futuro, poderá se otimizar partidas e paradas de unidades de processo. Acredita-se também que seja possível integrar a otimização de unidades de processo (destilação de petróleo, craqueamento catalítico (*FCC*), coqueamento retardado e hidrotratamento) com o parque de tanques de uma refinaria e começar a produzir derivados de petróleo especificados para venda de forma integrada com a programação de produção da refinaria.

A otimização estática (*RTO*) apresenta algumas limitações. Nota-se que o *RTO* integrado com o controle avançado pode não realizar todo o potencial de otimização da unidade de processo. Isto porque a otimização estática visualiza somente o estado final da planta, e poderá ocorrer do controle avançado (*MPC*) não poder atingir o ponto ótimo ao longo da trajetória. Já a otimização dinâmica (*DRTO*) não só otimiza o estado final, como também o caminho do processo, podendo ter uma realização maior do que o *RTO* (Almeida e Secchi, 2011).

Além da utilização de uma ferramenta de *DRTO* per si, há a necessidade de resolver problemas ocorridos durante o seu uso. Um dos maiores problemas que se enfrenta no uso das aplicações desta natureza é a capacidade de dar suporte à operação, na solução de falhas ocorridas ao longo do uso destas ferramentas e na busca de novas oportunidades de otimização. Para realizar estas tarefas, de forma eficiente, é necessário se municiar de ferramentas de monitoração, diagnóstico e sintonia de *DRTO*. O uso deste tipo de aplicativo permite ao engenheiro identificar e solucionar problemas e, conseqüentemente, obter maior confiança e utilização do *DRTO* por parte da operação.

Não se tem conhecimento da existência de ferramentas eficientes e eficazes de monitoração e diagnóstico de otimização dinâmica. Cabe ressaltar também que as ferramentas de monitoração e diagnóstico de otimização estática (*RTO*) ainda deixam muito a desejar.

Uma das principais falhas que ocorrem no uso da otimização de processos, e não é diferente na otimização dinâmica, é a inviabilidade de solução por especificações inadequadas das restrições de processo. Este assunto tem sido objeto de muitos estudos em *LP*, *NLP* e de forma insipiente em otimização dinâmica. Portanto, há uma necessidade de se tratar este problema de forma adequada em aplicações de *DRTO*.

Esta é uma área em fase de maturação, apesar de terem muitos estudos e há muito tempo, ainda há um grande caminho a ser percorrido até chegarmos a uma solução geral de aplicação industrial. Há ainda questões fundamentais a serem resolvidas, tendo muitos desafios a serem enfrentados como:

- Quais as diferenças das qualidades das soluções utilizando os métodos de *single-shooting*, *multiple-shooting*, colocação em elementos finitos e métodos híbridos?
- Quais os esforços computacionais e o tempo de execução para aplicação em tempo real de cada método?

- Qual a robustez de cada método?
- Qual a taxa de convergência de cada método?

Além disso, uma vez que se obtenha alguma solução do problema de otimização de equações algébrico-diferenciais (*DAOP – Differential-Algebraic Optimization Problem*), deve-se fazer uma análise crítica da solução obtida. Neste momento surgem algumas questões como:

- Como analisar os resultados do otimizador?
- Como analisar as oportunidades e cenários?

Ainda há uma série de dúvidas quanto ao comportamento dos algoritmos de NLP diante de problemas de larga escala. Os problemas em unidades de processo de refinarias de petróleo são da ordem de 100.000 variáveis de estado depois do problema discretizado.

Também é importante que a indústria consiga perceber os benefícios da otimização dinâmica frente à otimização estática. Neste caso deseja-se responder a seguinte questão:

- Quais as oportunidades de otimização dinâmica na operação de uma unidade de processo?

Finalmente, após avaliar os *softwares* de otimização dinâmica *offline* acadêmicos e comerciais, percebe-se que há um conjunto deles que tem boas metodologias. Porém, eles não proporcionam opções de escolhas de vários algoritmos. Por exemplo, o mesmo *software* não permite escolher entre a abordagem seqüencial ou simultânea, adaptação por elementos finitos móveis ou multi-escalas. Além disso, não apresentam mecanismos de avaliação de resultados e otimalidade. Enfim, seria conveniente que um *software* de otimização dinâmica proporcionasse todas as funcionalidades importantes para um sistema de *DRTO*. Por isso é necessário desenvolver algum aplicativo que agregue todas as funcionalidades importantes para um *DRTO*, incluindo mecanismos de estimação de estados da planta, disparo do otimizador, implementação de resultados, dentre outros.

Associado ao sistema de *DRTO*, é importante que se tenha mecanismos de diagnóstico, avaliação de soluções do otimizador (através da otimização *offline*), ou até mesmo fazer estudos de casos e reproduzir um caso real tratado pelo otimizador *online*. Contudo, as aplicações existentes não têm estas funcionalidades.

1.2 Objetivos

Este trabalho tem por objetivo a obtenção de uma solução eficaz de *DRTO*, menos sujeita a falhas e que resolva a maioria dos problemas de otimização dinâmica propostos em processos químicos. Para isso, também é necessário o desenvolvimento de um sistema de monitoração, análise e diagnóstico do sistema de *DRTO*. Esta ferramenta deverá ter a capacidade de identificar e notificar quando ocorre uma falha na solução do problema de otimização dinâmica, deverá seguir um procedimento sistemático para identificar possíveis falhas ou ineficiências do sistema, a analisar os resultados do otimizador com o objetivo de se certificar que o algoritmo esteja resolvendo adequadamente o problema de otimização dinâmica. Além disso, há um interesse especial na identificação e solução de inviabilidades

nos problemas de otimização dinâmica, pois este é um dos problemas mais freqüentemente encontrados em operações em tempo real. Em 20 anos de uso de *MPC's* na Petrobras, tem sido observada uma incidência não desprezível deste tipo de problema. Ao utilizar soluções de *DRTO* e ferramentas de diagnóstico, os operadores e engenheiros das plantas poderão auferir mais benefícios desses ativos de produção, aumentando as margens de lucros nas empresas e a confiabilidade da produção. Além disso, deve simplificar a atividade de programação de produção, pela maior integração desta atividade com a operação da planta, pois as ordens de produção tendem a ser mais simples.

1.3 Proposta

Com base no exposto acima, a presente tese se propõe a definir a estrutura de um sistema de otimização dinâmica em tempo real (*DRTO*), onde são reunidas as técnicas de otimização dinâmica desenvolvidas até o momento, e propor uma estrutura genérica para este tipo de aplicação. Além disso, esta tese define uma estrutura de um sistema de monitoração e diagnóstico de problemas nas soluções de *DRTO*. Isto passa por recomendações de modelagem de problemas de otimização dinâmica e a proposta de uma metodologia de solução de problemas de inviabilidades em otimização dinâmica. Esta metodologia serve tanto para sugerir uma solução viável como para realizar o diagnóstico de inviabilidade da solução do problema.

Sistema de *DRTO*

Esta tese propõe uma estrutura de sistema de *DRTO*, utilizando como *software* base o *EMSO* (*Environment for Modeling, Simulation and Optimization*), desenvolvido na *UFRGS*. Nesta estrutura, propõe-se utilizar os melhores métodos diretos de solução de *DAOP*, incluindo as opções das abordagens seqüenciais e simultâneas. Nesta estrutura, são utilizadas as metodologias de aquisição, validação e tratamento de dados já utilizadas na Petrobras S.A. Além disso, propõe-se aqui a utilização de uma formulação única de *DAOP* para a reconciliação de dados e detecção de erros grosseiros, estimação de parâmetros e de estados, podendo realizá-las de forma seqüencial ou simultânea. Nesta proposta, incluem-se também as técnicas de refinamento da solução, onde estão presentes a adaptação de malhas (como elementos finitos móveis e multi-escalas - *wavelets*) e detecção de estruturas.

Também serão implementadas algumas características usuais das soluções de *MPC*, como restrições de movimentos das variáveis de controle, para respeitar as regras de operação e suavizar as ações de controle. Além disso, também é proposto um mecanismo de implementação adequada dos perfis de controle computados pelo otimizador e um procedimento de monitoração da planta (perturbações e eventos) de forma a disparar o otimizador no momento adequado, além da própria monitoração dos resultados do otimizador. Cabe ressaltar aqui, que alguns tópicos são abordados de forma mais teórica e outros são implementados e testados.

Aqui também foram analisados os aspectos de desempenho, onde é chamado à atenção sobre as conseqüências da natureza do modelo sobre o desempenho do sistema. Os casos da otimização do *FCC* da *REFAP S.A.* – Petrobras S.A. (levou de 45-60 minutos para obter a solução com horizonte de 24 horas - Almeida e Secchi, 2011) e da torre

desisobutanizadora da *RBPC* – Petrobras S.A. (levou 106 segundos para simular 10 horas de operação - Staudt, 2007), mostraram o comportamento do algoritmo de otimização frente a problemas de grandes dimensões e modelos complexos. Isto remete a três medidas para procurar reduzir o tempo de processamento do sistema: a utilização de processamento paralelo na obtenção da solução, a redução de modelos e a análise de discretização de forma a avaliar o seu efeito na precisão e robustez da solução de *DAOP*.

Na definição desta estrutura são selecionados os métodos adotados em cada módulo do sistema, é definida a sua aplicabilidade e as opções de algoritmos, bem como a forma de uso e integração de cada método. São apresentadas questões conceituais envolvendo os sistemas de *DRTO*, tais como: a forma de abordar os problemas de incertezas na otimização dinâmica, discretização, construção e análise de modelos de otimização, análise de resultados do otimizador, dentre outros. Estes tópicos todos são abordados com foco nas aplicações industriais em tempo real.

Sistema de monitoração e diagnóstico de *DRTO*

Esta tese também propõe a estrutura de um sistema de monitoração e diagnóstico do *DRTO*. As aplicações de monitoração e diagnóstico são integradas de forma a fazer uma análise consistente. Esta ferramenta tem três níveis de diagnóstico, sendo que no primeiro nível tem-se uma visão geral do problema e da solução, num segundo nível tem-se uma visão detalhada dos mesmos aspectos (incluindo iteração-a-iteração) e num terceiro nível é realizado um diagnóstico detalhado de cada tarefa do otimizador.

A aplicação de diagnóstico é baseada na infra-estrutura do *EMSO*, onde se pode reproduzir um caso real tratado pelo otimizador *online* através da otimização *offline*, sem qualquer esforço adicional. Esta infra-estrutura permite paralisar a execução de um algoritmo de otimização e possibilitar ao usuário realizar uma análise do problema sobre os aspectos de natureza do problema, qualidade da discretização e problemas de sintonia ou estruturais do algoritmo de otimização. Ela incorpora ao sistema uma análise de sensibilidade do modelo para auxiliar na análise da solução de *DAOP*. Esta análise permitirá ao usuário avaliar o efeito da sensibilidade do sistema na solução do *DAOP*, bem como a sensibilidade do sistema diante da remoção de determinadas restrições de processo.

Além disso, é definido o procedimento de identificação de possíveis causas de uma determinada falha. Este procedimento tem diferentes níveis de detalhamento e, para cada rota, utilizam-se métodos baseados no problema de otimização dinâmica na sua forma original ou no *NLP* resultante do processo de discretização aplicado ao problema original. Na monitoração, tem-se uma análise das últimas rodadas do otimizador, onde são apresentados os últimos resultados, falhas e desempenhos.

Solução de inviabilidades em *DAOP*

Esta tese propõe uma nova metodologia de diagnóstico e solução automática de inviabilidades nos *DAOP's*, baseada na reformulação do *DAOP* original como um problema de otimização multi-objetivos com relaxamento das restrições. O problema de inviabilidade é resolvido utilizando a técnica de programação por metas (*goal-programming*), onde a solução pode ser encontrada com a utilização de qualquer algoritmo

de *DAOP's*. Esta proposta inclui uma linguagem incorporada na sintaxe de *DAOP* do *EMSO*, onde é somente necessário habilitar a detecção de inviabilidade, já pré-configurada. Esta técnica pode ser utilizada tanto no modo *online* como no *offline*, para diagnóstico ou para solucionar um problema de falha do otimizador.

Esta tese se propõe a dar um pequeno passo no sentido de encaminhar uma solução geral e robusta para aplicação industrial. Claro que há ainda questões fundamentais a serem resolvidas. Itens como: resolver o problema de otimização dinâmica no seu formato original diretamente em vez de resolver um *NLP* resultante, pois os *NLP's* não têm informações precisas sobre o comportamento dinâmico do sistema em instantes de tempo distantes de um determinado ponto no horizonte de tempo; em problemas de difícil solução, as técnicas de discretização se mostram deficientes e precisam ser aprimoradas, tendo em vista a competição entre precisão e graus de liberdade da solução e competência computacional para resolver o problema; a escolha dos instantes de tempo adequados na discretização continua sendo um desafio pelos motivos apresentados acima, sendo que a técnica de adaptação de malhas usando a técnica de *wavelets* parece promissora, necessitando de alguns aprimoramentos. A redução de modelos também é um item relevante na melhoria das técnicas de otimização, principalmente associada às técnicas de diagnóstico do *DRTO*. Isto pode simplificar a solução e torná-la mais robusta, principalmente para problemas de larga escala onde as questões numéricas são mais relevantes e mais difíceis de resolver. Além disso, as fortes não-linearidades do modelo podem causar problemas de convergência na solução de otimização. Uma abordagem sistemática auxilia na melhoria e robustez da solução.

1.4 Estrutura da tese

Esta tese contém duas partes e oito capítulos, distribuindo os assuntos na seguinte forma:

A **Parte 1 (Sistema de *DRTO*)** introduz os conceitos e revisão bibliográfica referentes a um sistema de *DRTO* e apresenta uma proposta de construção de um sistema de *DRTO*. Esta parte é dividida em três capítulos:

Capítulo 2 – Introdução. Neste capítulo, apresenta-se um histórico do cálculo variacional, do controle ótimo, da teoria de controle moderno, do controle preditivo e da otimização dinâmica. Também são introduzidas as definições básicas contidas nas aplicações de *DRTO* e suas formas de uso. Finalmente são apresentadas conceituações, características e formas de usos de ferramentas de diagnóstico e sintonia de *DRTO*.

Capítulo 3 – Revisão bibliográfica da otimização dinâmica. Neste capítulo, é feita uma revisão bibliográfica dos aspectos envolvidos no desenvolvimento de um sistema de *DRTO*. Abordam-se assuntos como a formulação e métodos de solução de *DAOP's*, funcionalidades importantes de um sistema de *DRTO*, diferentes estruturas de otimização dinâmica e aspectos de monitoração e diagnóstico de *DRTO*.

Capítulo 4 – Proposta de sistema de DRTO. Neste capítulo é proposto um sistema de *DRTO,* abordando aspectos de construção de modelos de *DAOP*, onde se propõe uma sintaxe e tradução para problemas de otimização dinâmica. Também se apresentam propostas de usos de técnicas de ajustes de modelos, reconciliação de dados e estimação de

estado, voltados para *DRTO*. Propõe-se uma estrutura para o sistema de *DRTO*, tanto as partes *online* como as *offline*. Além disso, agregam-se funcionalidades importantes como adaptação de malhas, detecção de estruturas, análise de otimalidade, dentre outras. Também é conceituada e idealizada uma ferramenta de monitoração e diagnóstico de sistemas de *DRTO* e, finalmente, feita uma análise das contribuições propostas no presente trabalho.

A **Parte 2 (Solução de problema de inviabilidade)** introduz os conceitos e revisão bibliográfica referentes ao problema de inviabilidade em otimização, e apresenta uma proposta de solução de problema de inviabilidade a ser usado no sistema de *DRTO*. Esta parte é dividida em três capítulos:

Capítulo 5 – Introdução. Introduzem-se os conceitos de inviabilidade em otimização dinâmica, tipos e suas causas básicas.

Capítulo 6 – Revisão Bibliográfica do problema de inviabilidade. Neste capítulo, apresentam-se os métodos de solução de inviabilidade em LP e NLP e a problemática em DAOP.

Capítulo 7 – Proposta de solução de inviabilidade. Neste capítulo, propõe-se uma metodologia de solução automática de inviabilidade de DAOP baseada no relaxamento das restrições. Nesta proposta são demonstrados os diferentes casos de inviabilidades que aparecem num DAOP, mostrando a efetividade da detecção e solução da inviabilidade.

Capítulo 8 – Conclusão e recomendações. Neste capítulo são apresentadas as principais questões abordadas na tese e as conclusões sobre os tópicos abordados. Além disso, também são reforçados os pontos de contribuição deste trabalho e as recomendações para trabalhos futuros.

Parte 1 – Sistema de DRTO

2 Introdução

O *DRTO* (*Dynamic Real-Time Optimization*) é uma tecnologia que tem características semelhantes a dos *MPC* (*Model Predictive Control*), onde são coletados dados da planta, calculadas ações de controle com base em modelos dinâmicos e implementadas as ações nos controladores da planta ou um *MPC*. Também é semelhante às aplicações de otimização estática, *RTO* (*Real-Time Optimization*), que utilizam modelos rigorosos do processo e se definem a função objetivo e as restrições do processo.

A origem das técnicas de otimização dinâmica vem de longa data. Os primeiros conceitos surgiram do cálculo variacional no século XVII e deram subsídios para o estabelecimento da teoria do controle ótimo em meados do século XX. Esta teoria foi consolidada com a conceituação do princípio do máximo de Pontryagin (1962). Com a evolução da teoria de controle ótimo e da necessidade de resolver problemas de otimização de difícil solução, surgiu uma ramificação utilizando aproximações dos problemas de controle ótimo, possibilitando a solução de problemas de larga escala e viabilizando a otimização dinâmica em tempo real (*DRTO*). Uma breve linha do histórico do controle variacional até o *DRTO* é apresentada da Figura 2.1.

O estudo de problemas do cálculo variacional é muito antigo, e considera-se o surgimento da teoria matemática do cálculo das variações com a formulação do problema de Braquistócrona, como bem comentam Kamien & Schwartz (1991) e Sussmann & Willems (1997). Em Junho de 1696, Johann Bernoulli publicou um desafio à comunidade matemática com este problema. Alguns matemáticos consideram este como sendo o marco do cálculo variacional. Por outro lado, outros pesquisadores consideram as datas 1728 ou 1744 para o nascimento da teoria do cálculo variacional quando Leonard Euler, em 1728, escreveu sobre a solução de equações para curvas geodésicas. Em 1744, Euler publicou o seu livro de referência a métodos para descobrir curvas que tem a propriedade de máximo ou de mínimo.

Em meados dos anos 1830, William Rowan Hamilton desenvolveu as bases da mecânica Hamiltoniana. Ele reescreveu o sistema de Euler-Lagrange de uma forma diferente, e mostrou que os problemas envolvendo muitas variáveis e restrições podem ser reduzidas a um conjunto de derivadas parciais de uma única função, que ficou conhecido como Hamiltoniano. Nos seus artigos de 1834 e 1835, Hamilton não mostrou em que condições a sua função possuía solução. Somente em 1838, Jacobi interveio e retificou esta teoria.

O Desenvolvimento matemático da teoria do controle ótimo começou em meados da década de 50, em resposta às demandas de várias áreas da engenharia e economia. Do ponto de vista matemático, a teoria de controle ótimo é uma ramificação da teoria do cálculo variacional. Na sua origem estão problemas da engenharia, voltados às áreas de balística, aeronáutica, astronáutica e robótica. Como regra geral, considera-se que a teoria de controle ótimo surgiu em finais dos anos cinqüenta do século vinte, na antiga União Soviética, com a formulação e demonstração do Princípio do Máximo de Pontryagin por

L.S. Pontryagin e seu grupo de colaboradores: V.G. Boltyanskii, R.V. Gamkrelidze e E.F. Mishchenko, principalmente depois que seu livro foi publicado na língua inglesa, em 1962.

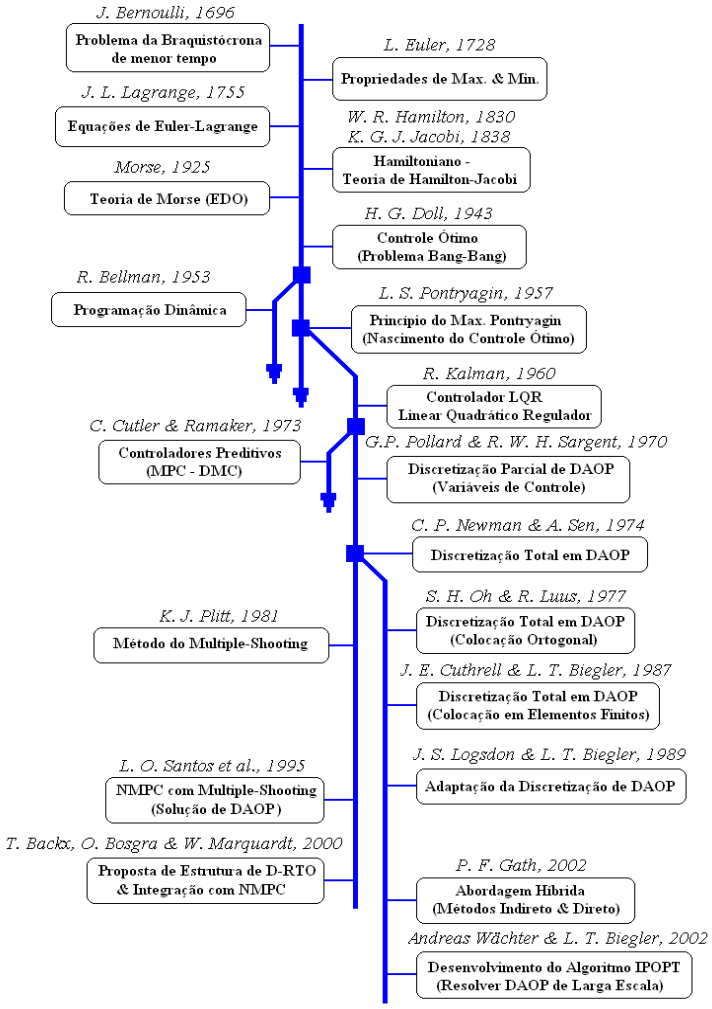


Figura 2.1 – Evolução da Otimização Dinâmica

A teoria de controle moderno é considerada uma extensão da teoria de controle ótimo. O desenvolvimento de conceitos da teoria de controle moderno pode ser atribuído ao trabalho de Kalman (1960) no início dos anos 60. Kalman procurou determinar quando um sistema de controle linear pode ser denominado ótimo. Ele estudou o chamado controlador *LQR*, projetado para minimizar uma função objetivo quadrática, demonstrando que a solução do problema *LQR* é um controlador proporcional, com o ganho matricial calculado através da solução de uma equação de Riccati. Esta teoria foi desenvolvida para estimar os estados internos de uma planta linear através de medidas com ruído. Esta técnica é utilizada atualmente como *Filtro de Kalman*, e o controlador *LQR* combinado ao Filtro de Kalman deu origem ao conhecido controlador Linear Quadrático Gaussiano (*LQG - Linear Quadratic Gaussian Control*).

A teoria de controle ótimo teve seu grande desenvolvimento voltado à análise e síntese de controladores usando variáveis de estado como uma alternativa à abordagem clássica. O

modelo representado na forma de espaço de estados permitiu o uso de modelos não-lineares. Neste mesmo período foram estabelecidos conceitos de controlabilidade, observabilidade, teoria da realização, etc. Além disso, surgiram desenvolvimentos de análise de estabilidade de sistemas lineares e sua extensão para sistemas não-lineares, usando a teoria de estabilidade de Lyapunov.

As aplicações bem sucedidas da teoria de controle moderno surgiram na engenharia aeroespacial nos *EUA* e *URSS*. Em processos químicos, não foi bem sucedida devido à incapacidade de representar as incertezas presentes nos modelos de processos químicos. As questões ligadas à robustez foram colocadas de forma inadequada. Por isso, o impacto do controle moderno na indústria química foi muito baixo.

A tecnologia de controle preditivo teve sua inspiração na teoria de controle moderno, onde no inicio da década de 1960, Kalman estudou o controlador *LQR – Linear Quadratic Regulator*, que minimizava uma função objetivo quadrática e tinha os primeiro conceitos de controle preditivo baseado em modelos. A tecnologia de controle preditivo teve seu grande impulso fornecido pela indústria, onde foram utilizados os conceitos de controle ótimo. Uma representação simplificada desta evolução é apresentada na Figura 2.2.

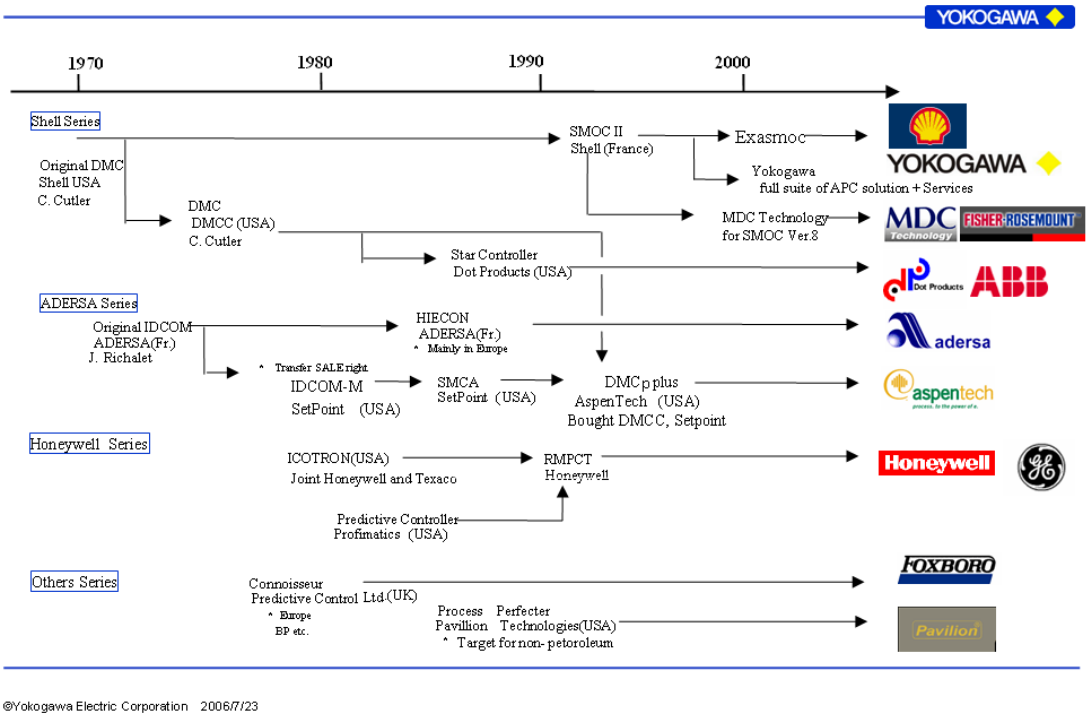


Figura 2.2 – Histórico do *MPC* Industrial

A primeira descrição de uma aplicação real de *MPC* na indústria foi apresentada por Richalet numa conferência em 1976 e resumida em um artigo da revista *Automatica* em 1978 (Richalet et al., 1978). O algoritmo ficou conhecido como *MPHC – Model Predictive Heuristic Control,* e o *software* implementado receberam a denominação de *IDCOM – Identification and Command* (da companhia Setpoint, Inc.). Nestes trabalhos, as ações de controle eram calculadas através de um problema de otimização da trajetória de saída do processo em malha aberta ao longo de um horizonte de predição futuro. Estes

controladores utilizam modelos de convolução lineares discretos na forma de resposta ao degrau ou impulso. Em 1982, Mehra e Rouhani introduziram melhorias teóricas na formulação do *IDCOM*, dando origem ao controlador *MAC – Model Algorithm Control*. Este controlador incluiu a análise de estabilidade e tratamento de ruídos de sinais.

Foi desenvolvido no início dos anos 70, um controlador independente que teve sua primeira aplicação em 1973. Cutler e Ramaker (1979) apresentaram os detalhes deste controlador multivariável sem restrições, denominado *DMC – Dynamic Matrix Control* no *AIChE meeting* em 1979. Em 1980, Prett e Gillette descreveram uma aplicação de um algoritmo *DMC* modificado para lidar com não-linearidades e restrições.

Com uma origem mais teórica, Brosilow (1979), Garcia e Morari (1982) desenvolveram o *IMC – Internal Model Control*, fazendo uma ligação com os controladores preditivos então apresentados. Este controlador não teve muita aceitação pela indústria de processamento, porém este controlador tem sido muito usado como ferramenta de sintonia de PID.

O *DMC* e *IDCOM* podem ser considerados como os representantes da primeira geração de algoritmos de controle preditivo industriais.

Em 1985, Morshedi et al. introduziu o controlador *LDMC – Linear Dynamic Matrix Control*. Este controlador incluiu as restrições do processo no problema de otimização, resolvendo o problema através de algoritmos de programação linear (*LP*). Cutler et al. (1983) e Garcia e Morshedi (1986) aplicaram um algoritmo de programação quadrática (*QP*) para resolver o problema de otimização do controlador. Estes trabalhos deram origem ao controlador *QDMC – Quadratic Dynamic Matrix Control*.

QDMC pode ser considerado como o representante de uma segunda geração de algoritmos de controle preditivo industriais, geração esta composta de algoritmos capazes de lidar com restrições de entrada/saída de forma sistemática.

O algoritmo *QDMC* apresentava várias deficiências do ponto de vista operacional, tais como resolver problemas com restrições não-rígidas (*soft constraints*) que precisam ser corrigidas para uma aplicação industrial mais consciente e lucrativa. Todos estes fatores motivaram os engenheiros das companhias Adersa e Setpoint, Inc. a desenvolver uma nova versão do software *IDCOM*. A versão desenvolvida pela Setpoint, Inc. foi denominada *IDCOM-M* (o M no final representa multivariável e visa à distinção em relação a uma versão *SISO* anterior) e a versão desenvolvida pela Adersa, Inc. denominou-se *HIECON* ("*Hierarchical Constraint Control*"). O *IDCOM-M* foi apresentado pela primeira vez por Grosdidier et al. na conferência *ACC* de 1988.

IDCOM-M e *HIECON* são apenas dois dos vários algoritmos que representam o que se conhece hoje como a terceira geração de algoritmos de controle preditivo industriais.

Na mesma época a empresa Profimatics vendeu o controlador *PCT* à Honeywell (1995), que promoveu a fusão deste com o *RMPC* desenvolvido internamente. Nesta fusão, foi criado um novo controlador *RMPCT* que introduziu as primeiras características de robustez aos controladores preditivos. O controlador tem mecanismos para evitar soluções inviáveis

mudando a estrutura do controlador baseado nos valores singulares dos ganhos de processo. Este controlador deu origem à quarta geração de controladores preditivos.

A representação de um *MPC* em variáveis de estado apareceu primeiramente formulada por Li et al. (1989) que explicam suas propriedades e mostram a utilização do filtro de Kalman para estimar o estado e perturbações não medidas do processo. Lee et al. (1994) também desenvolveram um *MPC* em variáveis de estado. Lee e Xiao (2000) e Lee e Cooley (2000) apresentam abordagens em variáveis de estado na forma incremental.

Algoritmos de *MPC* adaptativos como o *GPC – Generalized Predictive Control* desenvolvido por Clarke et al. (1987a/b) deram origem ao controlador *STAR* da Dot Products, porém não teve muita aceitação no mercado devido às dificuldades de parametrização para ter bom desempenho.

O caminho natural da teoria de controle foi o surgimento de uma teoria de controle não-linear a partir da década de 1980, tendo como marco principal o lançamento do livro de Isidori (1989), pois a partir desta publicação, as publicações sobre controle não-linear têm crescido continuamente. As principais abordagens de controle não-linear puro podem ser divididas em três grupos: controle geométrico, controle ótimo não-linear e controle preditivo não-linear. Controle ótimo não-linear apresenta problemas de falta de robustez. Outra limitação sua, que ocorre desde a década de 1950, é a grande dificuldade em se resolver as equações de síntese do controlador. Atualmente, uma nova ramificação do controle ótimo tem tentado incorporar robustez na formulação do problema de controle. Um exemplo desta tentativa baseado nas idéias do controle robusto moderno é a teoria de H-infinito não-linear (Van Der Schaft, 1991; Isidori, 1994; Isidori e Kang, 1995; Longhi, 2001).

Nos últimos anos, o *NMPC* (*Nonlinear Model Predictive Control*) tem atingido um grau de maturidade que permite a aplicação desta tecnologia em escala industrial. Um dos exemplos é o controle preditivo não-linear utilizando linearizações ao longo da trajetória (*LLT*), proposto por Duraiski et al. (2001). Há ainda alguns desafios a serem vencidos, tais como um mecanismo eficiente e seguro de estimar o estado da planta.

A otimização dinâmica surgiu como uma derivação do controle ótimo, pois o mesmo era resolvido através da utilização da abordagem variacional. Esta abordagem resultava em um problema de valor de contorno de difícil solução. Por este motivo, a maioria das aplicações era voltada a problemas de pequenas dimensões. Com isso, este tipo de solução se ajustava a problemas da indústria aeronáutica e aeroespacial, não sendo adequada à indústria de processamento, pois os problemas envolviam modelos mais complexos e um número muito maior de variáveis.

2.1 Otimização Dinâmica

Na década de 1970, vários pesquisadores começaram a procurar formas de simplificar as soluções dos problemas de otimização dinâmica. Inicialmente tiveram algumas iniciativas tímidas utilizando método *single-shooting*, herdado da área de balística, daí o nome. As primeiras iniciativas mais significativas ocorreram com Pollard e Sargent (1970) e Sargent e Sullivan (1977), utilizando algumas técnicas de discretização dos perfis de controle. Eles

foram os primeiros a produzir pacotes para resolver problemas de larga escala usando parametrização dos controles e a avaliação dos gradientes usando as equações adjuntas para poder resolver o problema de otimização resultante. Esta abordagem era do tipo de caminho viável (*feasible path*), pois produziam trajetórias viáveis para os estados a partir de perfis de controle sugeridos.

No inicio da década de 1980, já se conhecia bem as limitações do método *single-shooting*, e já haviam sido experimentadas algumas opções de solução de *DAOP*. Em 1981, Plitt apresentou o método de discretização *multiple-shooting*, sendo implementado no pacote chamado *MUSCOD*. E foi consolidado logo após em 1984 por Bock e Plitt, sendo aplicado na engenharia mecânica e, muito tempo depois, na otimização de movimentos de robôs (Schulz, 1994).

Cuthrell (1986) na sua dissertação e Cuthrell e Biegler (1989) logo depois aplicaram o método simultâneo (discretização total do *DAOP* transformando-o em *NLP*) em uma série de processos químicos. O método permitiu tratar descontinuidades nos controles e em alguns estados. Com isso, a abordagem simultânea se consolidou como uma boa opção para solução de problemas de otimização dinâmica, pois evitava as dificuldades da parametrização do controle e estado encontradas nos outros métodos, resolvendo o problema de otimização de forma simultânea via *NLP*. Estes métodos também foram chamados de *infeasible path*, pois as trajetórias geradas ao longo do procedimento de solução só são viáveis quando a solução ótima é obtida. Neste caso, a solução do problema de otimização e a satisfação das restrições do modelo e do processo são obtidas simultaneamente sem a necessidade de integrar as equações diferencias do sistema. Trajetórias de controle se tornaram parte das variáveis de otimização e as instabilidades e não-linearidades dos modelos dinâmicos puderam ser mais bem controladas. A partir daí começaram a resolver *DAOP's* de dimensões maiores.

Cuthrell e Biegler, em 1989, começaram a se defrontar com problemas *NLP* esparsos e de dimensões muito elevadas e sistemas rígidos (*stiff*). Isto forçou o desenvolvimento de algoritmos especializados para resolver problemas de *NLP* de grandes dimensões. Em 1983 surgiu uma aplicação pioneira (Locke et al., 1983) onde se utilizou o método *SQP* parcialmente reduzido (*PRSQP*) muito usado mais tarde para resolver *DAOP* de engenharia (Vasantharajan e Biegler, 1988; Vasantharajan et al., 1990; Biegler et al., 1995). Esta teoria só foi formalizada bem depois por Schulz (1996), que mostrou que o *PRSQP* é uma generalização de ambos *full-space SQP* e *reduced-space SQP* (*rSQP*).

Na década de 1990, os métodos seqüenciais tiveram uma retomada, pois os computadores melhoraram seus desempenhos e os métodos tiveram uma série de aperfeiçoamentos. Em 1993, Vassiliadis utilizou a discretização parcial (das variáveis de controle) com polinômios de Lagrange, perfis constantes e lineares por partes. Tendo uma série de relatos da utilização de discretização dos controles logo em seguida (Vassiliadis et al., 1994; Pytlak e Vinter, 1996; Feehery e Barton, 1998). Em 1994, Vassiliadis et al. resolveram um problema de controle ótimo usando o *DAEOPT*, um pacote desenvolvido por eles. Em 1994, Barton e Pantelides resolveram problemas *DAOP* utilizando o *gOPT*/*gPROMS* desenvolvido pela mesma equipe na Inglaterra. Em 1996, Leinweber implementou uma estratégia *reduced-space* de solução de *NLP* de grande porte no pacote *MUSCOD-II*. Em 1997, Leineweber et al. reapresentou o método do *multiple-shooting* na sua versão

simultânea, inspirado no método já apresentado por Plitt (1981). Esta abordagem permite o uso de valores inconsistentes dos estados através de uma formulação de *DAE* relaxada proposta por Bock et al. (1988). Este método formou a base das aplicações de otimização dinâmica em tempo real. Estratégia simultânea do *multiple-shooting* também foi utilizada por Tanartkit e Biegler (1995 e 1996).

Também houve iniciativas na combinação dos métodos direto e indireto e, em 2002, Gath desenvolveu o pacote *CAMTOS – Collocation and Multiple Shooting Trajectory Optimization Software*. Dos métodos diretos podem ser utilizados os métodos *multiple-shooting* e colocação direta – *DIRCOL* (Stryk, 1999).

Na última década e meia, uma série de ferramentas de simulação e otimização dinâmicas comerciais e acadêmicas foi desenvolvida para atender as demandas da indústria. Estas ferramentas, tais como *ASPEN Custom Modeler* (Aspentech, 2002), *gPROMS* (PSE, 2004), *DYNOPC* (Lang e Biegler, 2007) e *INCOOP - DyOS* (Brendel *et al.*, 2008), se propõem a resolver estes problemas. Contudo, estas ferramentas podem ser utilizadas somente *offline* com a planta.

A aplicação de otimização dinâmica tem basicamente três grandes módulos. Primeiramente é necessário construir o modelo de otimização. Este modelo consiste da formulação do *DAOP*, onde são definidas a função objetivo, as variáveis de controle (decisão), o modelo do comportamento dinâmico do processo e as demais restrições do problema. As restrições podem ser estabelecidas ao longo do caminho (horizonte de otimização), pontuais (em instantes de tempos interiores) e terminais (de tempo final). Em segundo lugar, resolve-se o problema de otimização e por último, se optar pelo refinamento da solução (em se tratando do problema parametrizado na variável de controle), pode-se adaptar a malha discreta e refinar os arcos de controle.

Os *DAOP's* normalmente têm uma receita fixa, sendo representado por um problema de simples estágio. Porém, há outros *DAOP's* onde a receita é alterada ao longo do tempo (diferentes estruturas do *DAOP*), neste caso o problema pode ser tratado como de múltiplos estágios.

Com o modelo de otimização construído, pode-se resolver o problema de otimização utilizando métodos indiretos ou diretos. Dentre os métodos indiretos, destaca-se o que utiliza o princípio do máximo de Pontryagin (*PMP*). Este princípio é muito utilizado na solução de problemas de balística e astronáutica, e em alguns poucos casos de otimização de sistemas em batelada. As aplicações do *PMP* em otimização de processos químicos estão mais voltadas à análise da estrutura da solução do que na obtenção da solução propriamente dita. O *PMP* é utilizado para definir a estrutura de arcos de controle da solução ótima, utilizado no processo de definição da estratégia de controle otimizante ou no refinamento da solução. Outra classe são os métodos diretos, onde o *DAOP* é discretizado e transformado em um *NLP* (*Nonlinear programming*). Os métodos podem ser classificados em seqüenciais, onde o *DAOP* é parcialmente discretizado (somente variáveis de controle) e totalmente discretizado (as variáveis de estado e controle).

A escolha do método de solução é dependente do problema a ser resolvido. Normalmente a escolha é feita entre os métodos diretos e indiretos, entre a abordagem seqüencial e

simultânea, e na opção pela abordagem seqüencial, a definição pelo uso do método *single-shooting* ou *multi-shooting*.

Além disso, há os métodos de solução de *DAOP's* multi-objetivos. Estes métodos podem utilizar a abordagem paramétrica (como: pesos ponderados e programação por metas) ou estocástica (como: algoritmos genéticos). Caso o problema de otimização tenha eventos discretos, há a necessidade de resolver um *DAOP* com decisões inteiras. A técnica recomendada, neste caso, é a utilização de métodos de otimização dinâmica inteira mista (*MIDO - Mixed-Integer Dynamic Optimization*).

Ao obter a solução do *DAOP* pelo uso de métodos diretos, é necessário discretizar o horizonte de otimização. A escolha do número e posição dos elementos discretos afeta diretamente as oportunidades de otimização (pela limitação dos graus de liberdade do problema) e a precisão na solução ótima. Para minimizar estes efeitos, pode-se refinar a solução ótima com técnicas de adaptação de malha. Além disso, ao escolher os pontos discretos, não é possível saber exatamente os instantes onde as restrições se tornam ativas ou desativas. Ao fazer uma análise a posteriori da solução ótima, é possível refinar a solução através da reformulação do *DAOP* como um problema de múltiplos estágios onde o tempo final de cada estágio passa a ser um parâmetro a ser determinado.

Ao longo dos últimos anos, foram desenvolvidos aplicativos de otimização dinâmica comerciais e acadêmicos com características e funcionalidades diferentes. O ponto comum mais importante está no fato destes softwares serem aplicáveis somente à otimização dinâmica *offline*, ou seja, desconectado da planta. Não há soluções de *DRTO* disponíveis no mercado. Há somente algumas soluções projetadas sob encomenda e aplicáveis a *NMPC's*, para processos contínuos e em bateladas.

2.2 Sistema de DRTO

Um sistema de DRTO pode ser construído acrescentando os módulos de comunicação com os sistemas de controle da planta à aplicação de otimização dinâmica. Desta forma, o mesmo pode se tornar uma aplicação *online*, onde por um lado realiza-se a estimação de estados da planta e por outro lado, executa-se a implementação das ações de controle no sistema de controle.

Recentemente, surgiram as primeiras iniciativas no intuito de criar uma estrutura de *DRTO* tal qual a estrutura de *RTO* (otimização estacionária). Backx, Bosgra e Marquardt (2000) propuseram uma estrutura de *DRTO* integrando com *NMPC*. Kadam et al. (2002) fizeram uma análise das diferentes estruturas de *DRTO*, introduzindo o conceito de disparador de *DRTO*. É importante ressaltar que os sistemas de otimização dinâmica *offline*, como o *gPROMS* (*PSE*, 2004), estão consolidados e maduros, porém as aplicações em tempo real ainda estão em fase de desenvolvimento e ainda há grandes evoluções a serem feitas nesta área.

Algumas otimizações dinâmicas em tempo real foram aplicadas a *NMPC's* usando a abordagem seqüencial, sendo chamadas de múltiplos passos (*multistep*). O algoritmo de controle era do tipo Newton (Li e Biegler, 1988). Em 2000, Biegler propôs o uso do método de colocação em *NMPC's*, não tendo muita aceitação. Os resultados apresentados

estimularam o desenvolvimento nesta área, permitindo que se pensasse em executá-lo *online,* em tempo real. Esta iniciativa reacendeu o interesse pelo desenvolvimento de *NMPC*, que somente hoje está começando a ter aplicações industriais consolidadas. No entanto, os *DRTO's*, que tem os mesmos fundamentos do *NMPC*, estão tendo um impulso para resolver problemas de otimização dinâmica em tempo real. Somente agora, esta área está se fortalecendo.

A otimização dinâmica em tempo real (*DRTO*) é uma aplicação *online*, sendo executada de forma cíclica e conectada à planta, onde se coletam informações de instrumentos da planta e se realiza o cálculo da trajetória ótima das variáveis de controle, com o auxílio de algoritmos de otimização dinâmica. Os valores das ações de controle calculadas, por sua vez, são implementados na planta, alterando o ponto de operação. Estas mesmas ferramentas podem ser adaptadas para utilizá-las como controladores preditivos não lineares, para resolver problemas de controle e otimização de processos com dinâmicas não lineares.

Uma aplicação de *DRTO* tem funcionalidades a mais das aplicações de otimização *offline*, relacionadas às etapas de *feedback* da planta e de implementação das ações de controle ótimo. Associado a este *feedback*, tem-se a atualização dos parâmetros do modelo e do estado da planta (condições iniciais). A estimação do estado da planta é acompanhada da reconciliação de dados e detecção de erros grosseiros. Estas três tarefas utilizam as mesmas informações da planta, e podem ser feitas de forma seqüencial ou simultânea. A decisão é dependente do problema e da necessidade de atualização dos parâmetros em determinados momentos. Esta tarefa passa por uma etapa de aquisição, validação e tratamento de dados dos sistemas de controle (*SDCD - Sistema digital de controle distribuído*) da planta. Do outro lado, a tarefa de avaliação e implementação dos resultados do otimizador deve verificar as condições de otimalidade da solução, efetuar análise discriminante dos perfis de controle (evitar atuações desnecessárias) e de sensibilidades com os controles (verificar a robustez e flexibilidade da solução). Além disso, devem ser calculados os indicadores básicos para a monitoração e emissão dos relatórios das soluções.

Um sistema de *DRTO* pode ter diferentes estruturas, sendo esta escolha função da aplicação e da infra-estrutura computacional e disponibilidade de informações da planta. Em função deste ponto, podem-se ter algumas funcionalidades adicionais tais como: o separador de escalas de tempo (compatibilizar as freqüências de amostragem do *DRTO* e *NMPC*) e do disparador do *DRTO*. Além de estruturas hierárquicas como algumas soluções de otimização dinâmica de processos em batelada.

2.3 Aplicações de DRTO

Um sistema de *DRTO* deve ser geral, tendo capacidade de modelar e resolver diferentes classes de problemas de otimização dinâmica. Este sistema deve ser eficiente e de uso fácil, onde é possível resolver o problema num tempo viável e utilizável por pessoas que não são especialistas no conhecimento da técnica. Além disso, o sistema deverá ser capaz de obter soluções de boa qualidade, próximas ao ótimo real.

O sistema de *DRTO* deve ter módulos de construção do modelo, de atualização do modelo do processo e do estado da planta. Além disso, este sistema deve ter um módulo de solução

do *DAOP*, de avaliação e implementação de resultados e de gerenciamento de execução do otimizador. A estrutura do sistema de *DRTO* deverá permitir que seja realizada tanto a otimização *online* como a *offline*, e que permita visualizar a solução do otimizador *online*, estudos de casos de otimização dinâmica, além de permitir fazer diagnóstico e sintonia do otimizador.

Existem duas classes de aplicações em tempo real que utilizam algoritmos de otimização dinâmica. A primeira classe é o *MPC - Model Predictive Control* (Zavala, 2008; Zavala et. al., 2008) e o segundo é o *DRTO* (Kadam et al, 2002; Kadam et al, 2003). É importante distinguir estes dois tipos de aplicações. O *MPC* tem função objetivo pré-estabelecida, o tempo final e freqüência de execução fixada. Todas as ações de controle futuras são calculadas a cada ciclo de execução, mas apenas a primeira ação de cada variável de controle é efetivamente aplicada no processo. A formulação não-linear (*NMPC*) é normalmente aplicada aos processos químicos contínuos com comportamento dinâmico complexo e não-linear. Por outro lado, em aplicações de *DRTO* a função objetivo é definida de acordo com o problema a ser resolvido e todas as ações de controle são aplicadas no processo como trajetórias de referência para a camada de controle avançado. A otimização das receitas de processos em batelada ou semi-bateladas é um candidato interessante para o uso de *DRTO*, que é executada sob demanda.

As aplicações de *MPC*, *RTO* e *DRTO* usam a técnica de otimização conhecida como baseadas em medições, onde as informações do processo são atualizadas com as medidas da planta. Isto é necessário devido às incertezas causadas por mudanças nas condições iniciais, deficiências do modelo (parâmetros do modelo incerto) e perturbações do processo (Kadam et al., 2007). Existem dois esquemas comuns que podem ser adotados para os problemas de aplicação em tempo real (Srinivasan e Bonvin, 2007). O primeiro é o esquema explícito em que o algoritmo realiza a estimação de estado (com validação de medição, reconciliação de dados, estimação de parâmetros do modelo e avaliação do estado), otimização dinâmica e implementação da ação de controle. Este esquema é usualmente adotado em *MPC*, *RTO* e aplicações *DRTO*. O segundo esquema, que pode ser usado é o método implícito, mais conhecido como *NCO-Tracking* (Condições Necessárias de Otimalidade), que explora o chaveamento de estrutura do problema de otimização (Srinivasan et al., 2003) e utiliza as medições diretamente para adaptar trajetórias dos controles. Esta é também conhecida como re-otimização on-line através de feedback da planta e é muito adequado para resolver problemas de grande porte, que são muito custosos em termos de esforço da *CPU*, uma vez que não há necessidade de resolver o problema de otimização dinâmica em tempo real (Srinivasan e Bonvin, 2007). Esse tipo de aplicação tem um sistema disparador que monitora o processo e inicia o otimizador dinâmico cada vez que o problema de otimização sofre mudanças da estrutura ou uma perturbação significativa invalida a solução obtida pelo controlador auto-otimizante.

2.4 Monitoração e Diagnóstico de DRTO

O sucesso de uma ferramenta de *DRTO* está associado à capacidade do sistema fornecer informações que permitam encontrar as causas de eventuais problemas ocorridos com o otimizador e identificados pelos usuários. Um aspecto importante que contribui para a boa aceitação do *DRTO* por parte dos usuários está ligado à capacidade de se rastrear os

resultados do otimizador, bem como de responder as questões formuladas pelos seus usuários e permitir a solução de problemas ocorridos durante a sua utilização.

Há momentos, durante a utilização do *DRTO*, onde podem ocorrer instabilidades ou inadequações nos resultados do otimizador, desempenho inadequado ou até mesmo a ocorrência de falhas na solução do *DAOP*. Perguntas que normalmente surgem ao longo da utilização do sistema de *DRTO* são do tipo:

- Quais são os desempenhos e desvios observados no *DRTO*?
- Por que não se atinge o objetivo esperado?
- Por que determinadas restrições estão indevidamente ativas ou violadas?
- Por que houve falha na obtenção da solução?

Para responder a estas perguntas e outras mais, é importante que se tenha ferramentas de monitoração, diagnóstico e sintonia do *DRTO*. Desafortunadamente, não se tem conhecimento da existência deste tipo de ferramenta. Há apenas algumas iniciativas de monitoração de *RTO's*, onde são acompanhados alguns indicadores básicos das suas rodadas e históricos dos resultados do otimizador. Esta ferramenta, em *RTO*, tem um nível bom de monitoração, mas não para o diagnóstico. Há somente relatórios gerados pelos algoritmos de *NLP* e registros de alguns eventos de sistema. Estes relatórios são de difícil entendimento e se consome muito tempo para encontrar as possíveis causas dos problemas ocorridos.

Na área de algoritmos de *LP* e *NLP*, houve iniciativas de construção de ferramentas de diagnóstico, onde são focalizados no diagnóstico de inviabilidades na formulação do problema. Podem-se citar aqui alguns pacotes como o *AIMMS* (Roelofs e Bisschop, 2009), *Analyze* (Greenberg, 1996) e *GAMSCHK* (McCarl, 1998), que procuram principalmente identificar estas inviabilidades e analisar o modelo de otimização (principalmente em *LP's*).

Diante da ausência destas ferramentas em *DRTO*, está sendo proposto, neste trabalho, o projeto conceitual de uma ferramenta de monitoração, diagnóstico e sintonia de *DRTO*. Para construir este conceito, recorreu-se a aqueles utilizados no processo de melhoria de qualidade "seis sigma", e principalmente a metodologia *DMAIC* (*define*/ *measure*/ *analyze*/ *improve*/ *control*). Esta metodologia foi aqui adaptada para o enfoque de *DRTO*. Além da aplicação de conceitos de solução sistemática de problemas, também são agregados métodos de análises específicas de módulos do *DRTO*, tais como a análise das condições de otimalidade, avaliação de erros de discretização, avaliação de desempenho do otimizador e análise de sensibilidade da solução.

A ferramenta de monitoração é uma ferramenta que roda em tempo real e acompanha cada resultado do otimizador. A mesma verifica o cumprimento da receita e calcula os indicadores gerenciais e técnicos da otimização. O usuário poderá visualizar estes resultados de forma *offline* e organizada. A ferramenta de diagnóstico e sintonia do *DRTO* é uma aplicação *offline* que analisa os resultados de uma determinada rodada do otimizador. Este diagnóstico é feito utilizando as informações fornecidas pelo próprio otimizador, ou com a obtenção de informações adicionais através da reprodução, no modo *offline*, da rodada problemática do sistema *online*.

Quanto à forma de uso do ferramental de monitoração e diagnóstico do *DRTO*, esta infra-estrutura pode ser utilizada para analisar falhas ocorridas com o otimizador, bem como a realização da análise de processos do mesmo. As ferramentas de monitoração e diagnóstico devem ser utilizadas em conjunto e de forma colaborativa. A ferramenta de monitoração detecta o problema ocorrido ou até mesmo relata bons resultados. A ferramenta de diagnóstico auxilia ao engenheiro na busca das causas básicas do problema e na definição das ações mitigadoras a serem adotadas.

3 Revisão bibliográfica

O sistema de otimização dinâmica em tempo real (*DRTO - Dynamic Real-Time Optimization*) é uma aplicação que é executada de forma periódica e conectada à planta. De forma semelhante às aplicações de otimização estacionária em tempo real (*RTO – Real-Time Optimization*), para se realizar uma otimização dinâmica em tempo real, precisa-se formular o problema de otimização dinâmica (*DAOP - Differential Algebraic Optimization Problem*), executar a estimação de estados do processo, resolver o *DAOP* e implementar os perfis ótimos de controle na planta ou em aplicações de controles avançados. Isto pode ser visto na Figura 3.1a.

Esta mesma aplicação também pode ser executada de forma *offline*, onde a estrutura da otimização dinâmica é semelhante à aplicação *online*, sendo que a estimação de estados é substituída por um arquivo de dados e a implementação das ações de controle é substituída por um arquivo de relatório ou um visualizador de resultados. Ao contrário da aplicação *online*, a *offline* não é executada de forma periódica, podendo ser feito sob demanda através de uma interface homem-máquina (*IHM*) amigável. A Figura 3.1b mostra a sua estrutura geral.

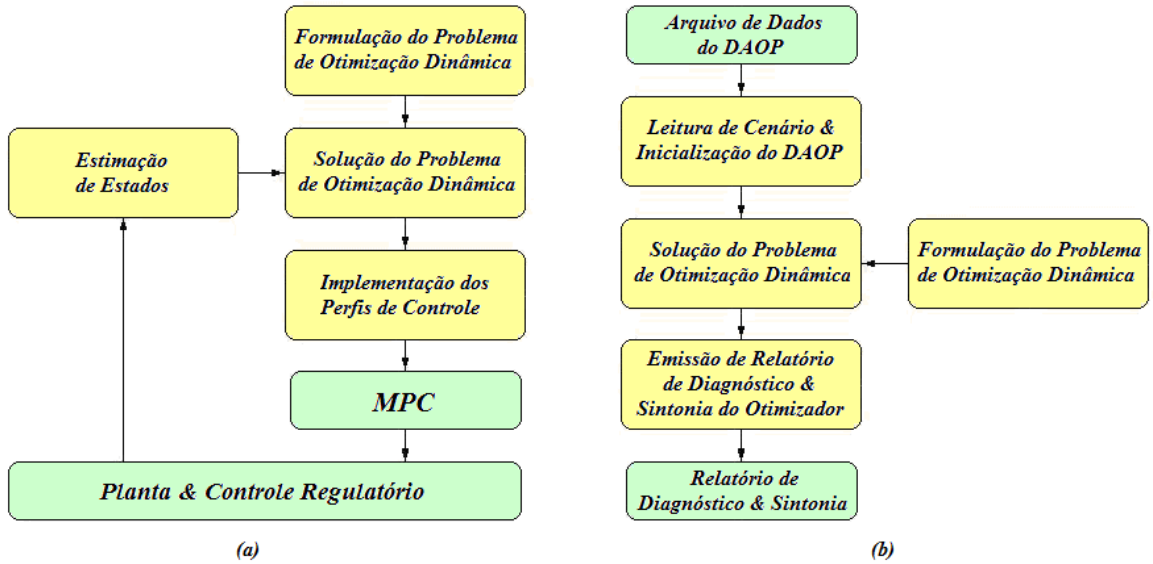


Figura 3.1 – Estrutura geral da otimização dinâmica – *online* e *offline*.
(a) Estrutura geral do *DRTO* – *online*; (b) Estrutura geral da otimização dinâmica – *offline*.

O *DRTO* minimiza/maximiza um ou mais objetivos de produção, manipulando as variáveis de controle (ex.: *setpoints* de controladores ou até mesmo definindo os perfis ótimos para a camada de controle avançado) ao longo do horizonte de otimização, e respeitando todos os limites operacionais e dos equipamentos.

Foi desenvolvida na última década uma série de ferramentas de simulação e otimização dinâmicas comerciais e acadêmicas para atender as demandas da indústria. Estas ferramentas, tais como *ASPEN Custom Modeler* (Aspentech, 2002), *gPROMS* (PSE, 2004), *DYNOPC* (Lang e Biegler, 2007) e *INCOOP - DyOS* (Brendel *et al.*, 2008), se propõem a

resolver estes problemas. Estas ferramentas podem ser utilizadas somente *offline* com a planta.

Estas mesmas ferramentas podem ser adaptadas para utilizá-las como controladores preditivos não lineares (*NMPC – Nonlinear Model Predictive Control*), para resolver problemas de controle e otimização de processos com dinâmicas não lineares. Além disso, pode ser usado para realizar a identificação de modelos não lineares do processo para até mesmo atualizar modelos de controladores lineares, por linearização no ponto.

Neste capítulo, apresentam-se as técnicas e métodos envolvidos numa aplicação de *DRTO*. Serão descritos alguns métodos básicos que são utilizados neste tipo de solução e outros que representam algumas soluções alternativas para tal sistema. É apresentada também uma análise crítica da utilização desses métodos em aplicações de *DRTO*. Aqui é descrita em detalhes as formas de construir um problema de otimização dinâmica. Em seguida, discutem-se os diferentes métodos de solução do problema. Finalmente, algumas estruturas de aplicações em tempo real e suas funções neste tipo de aplicação, bem como iniciativas de monitoração e diagnóstico da otimização.

3.1 Sistemas de otimização dinâmica

Para se resolver um problema de otimização dinâmica, primeiramente precisa-se formular o problema a ser resolvido. Neste tópico são descritas as diferentes partes do *DAOP* e algumas formas de uso.

3.1.1 Formulação do problema (*DAOP*)

Ao formular o problema de otimização dinâmica, deve-se analisar o sistema quanto ao seu objetivo, às variáveis de decisão, quanto ao modelo do processo e suas restrições. Na definição dos objetivos, devem ser definidos os benefícios a serem obtidos do sistema através de uma métrica de avaliação do desempenho. Além disso, devem-se definir quais as variáveis dependentes e de decisão (livres) envolvidas do problema. O modelo do processo descreve o comportamento dinâmico a ser otimizado. Na formulação do problema de otimização, definem-se as restrições que delimitam diretamente a região viável da solução. As restrições podem ser de caminho, pontuais (pontos interiores) e terminais (de tempo final). As restrições de caminho são aquelas a serem respeitadas ao longo das trajetórias das variáveis (dentro do horizonte de otimização). As restrições pontuais são imposições estabelecidas em instantes de tempo específicas. E as restrições de tempo final são aquelas que estabelecem as condições no final de uma operação.

As restrições podem ser de igualdade ou desigualdade. As restrições de igualdade são aquelas que descrevem o modelo do processo ou restrições adicionais impostas ao problema. Estas restrições adicionais são equações definidas pelo usuário de forma a estabelecer algumas regras ou caminhos a serem seguidos na solução do problema de otimização. Estas restrições podem ser, por exemplo, eficiência, recuperação ou rendimento, e até mesmo uma relação de vazões imposta ao processo. As restrições de desigualdade são aquelas que definem a janela de operação (a região viável) e suas fronteiras (os domínios das variáveis de estado, de saída e de controle).

Ao executar um determinado plano de produção, pode-se definir uma receita a ser otimizada em um simples estágio de operação ou em múltiplos estágios. As receitas de múltiplos estágios são aquelas que podem ter diferentes objetivos, restrições ou até mesmo modelos de processo. Neste caso, acrescentam-se algumas condições adicionais no problema de otimização que serão descritas mais a frente.

3.1.1.1 Problema de Simples Estágio

A formulação de um problema de otimização dinâmica pode ser descrita na Figura 3.2. O problema consiste de uma função objetivo ***J***, vetor de restrições de igualdade ***H***, vetor de restrições de desigualdade ***G***, condições iniciais $\boldsymbol{x_0}$, onde ***x***, ***y***, ***u*** e ***p*** são os vetores das variáveis diferenciais de estado, algébricas, de controle e parâmetros invariantes no tempo respectivamente.

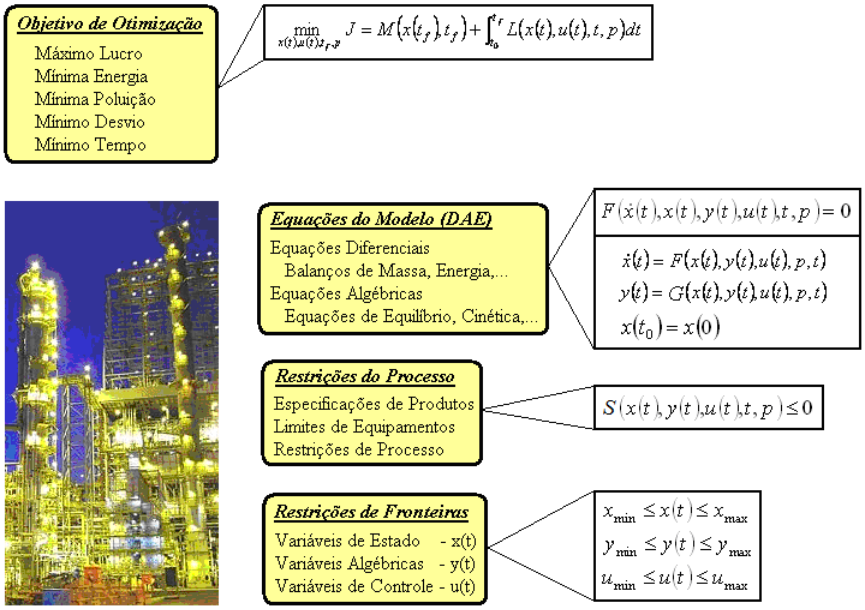


Figura 3.2 – Formulação do problema de otimização dinâmica.

Uma vez que a estrutura do problema e o modelo do processo estão definidos, deve-se determinar qual será o método de solução do problema. Ao definir estes pontos, é possível que haja a necessidade de reformular a apresentação do problema de otimização (ex.: discretizar o problema) para que se possa utilizar o algoritmo escolhido.

Para resolver um problema de otimização dinâmica, primeiramente deve-se construir o problema em questão. Na construção do problema, formula-se a função objetivo, descreve o modelo do processo, e define as restrições de caminho, pontuais e terminais. A seguir, são apresentadas as formas destas partes do problema de otimização dinâmica.

Formulação da Função Objetivo

A função objetivo é uma medida escalar do desempenho ou sucesso de um determinado processo a ser otimizado (maximizado ou minimizado). Os objetivos típicos de um processo a ser otimizado são normalmente ligados a critérios econômicos (Ex.: maximizar o lucro ou minimizar os custos; minimizar o consumo de insumos e energia), a critérios de segurança (Ex.: maximizar a eficiência de um ciclo de compressão), qualidade de produtos (Ex.: minimizar folga de especificação de produtos), flexibilidade (Ex.: minimizar o tempo de uma transição de processo ou de uma batelada), e critérios ambientais (Ex.: minimizar a produção de efluentes poluentes).

Há casos onde se deseja atender a várias especificações e requisitos, tais como máximo lucro, mínima folga de especificação de produtos e máxima vazão de carga. Há situações onde os objetivos são concorrentes, e deve-se obter uma solução compromisso. Neste caso, a formulação de uma única função objetivo com restrições pode não representar de forma adequada o problema, sendo necessária a definição de um conjunto de funções objetivo, que devem ser balanceadas de alguma maneira.

As variáveis de decisão são as variáveis livres que aparecem na função objetivo de forma explícita ou implícita através do modelo do problema. Elas representam os graus de liberdade a serem otimizados no sistema.

Genericamente, o índice de desempenho pode ter três formas possíveis. A formulação mais genérica é apresentada na forma do *Problema de Bolza* (Bliss, 1946 e Kamien e Schwartz, 1991), onde se tem uma parcela referente ao estado final do processo, chamado termo de *Mayer* (Kamien e Schwartz, 1991), e outra ligada ao desempenho ao longo da trajetória do sistema, chamado termo de *Lagrange* (Bliss, 1946 e Kamien e Schwartz, 1991). Desta forma, pode-se apresentar o **problema de *Bolza*** na seguinte forma:

$$J(u(t)) = \underbrace{\phi(x(t_f),t_f)}_{termo\ de\ Mayer} + \underbrace{\int_{t_0}^{t_f} \varphi(x(t),u(t),t)dt}_{termo\ de\ Lagrange} \tag{3.1}$$

Há dois casos especiais que são conseqüências de simplificações do problema de ***Bolza***, são elas as formas de *Lagrange* e de *Mayer*.

Na forma do **problema de *Lagrange***, só interessa a parte da função objetivo integral. Desta forma, pode-se dizer que o problema de *Lagrange* é o problema de *Bolza* onde $\phi(x(t_f),t_f) \equiv 0$. Portanto a função objetivo passa a ter a seguinte forma:

$$J(u(t)) = \int_{t_0}^{t_f} \varphi(x(t),u(t),t)dt \tag{3.2}$$

Por outro lado, na forma do **problema de *Mayer*** só interessa a parte da função objetivo relativo ao estado final do sistema. Desta forma, pode-se dizer que o problema de *Mayer* é o problema de *Bolza* onde $\varphi(x(t),u(t),t) \equiv 0$ e, portanto a função objetivo passa a ter a seguinte forma:

$$J(u(t)) = \phi(x(t_f), t_f) \tag{3.3}$$

onde:

J – é o critério de desempenho da otimização;
ϕ – é a parte que avalia a condição final da função objetivo;
φ – é a parte que avalia a função objetivo ao longo do horizonte de otimização;
x – é o vetor das variáveis de estado;
u – é o vetor das variáveis de controle;

Na realidade os problemas de *Mayer* e de *Lagrange* são tão gerais quanto o problema de *Bolza*. Se formular o problema de *Bolza* usando estados adicionais, pode-se definir um novo vetor de estados z onde:

$$z(t) = \begin{bmatrix} x^0(t) & x(t)^T \end{bmatrix}^T = \begin{bmatrix} x^0(t) & x^1(t) & \cdots & x^{nx}(t) \end{bmatrix}^T$$

sendo $z(t) \in \Re^{nx+1}, t \in [t_0, t_f]$ e $x_0(\cdot)$ é uma função contínua tal que:

$$\dot{x}^0(t) = \varphi(x(t), u(t), t), \qquad x^0(t_0) = 0 \tag{3.4}$$

Desta forma, pode-se dizer que:

$$x^0(t) = \int_{t_0}^{t} \varphi(x(\tau), u(\tau), \tau) d\tau$$

Pode-se reescrever o problema de *Bolza*, na seguinte forma de *Mayer*:

$$J(u(t)) = \phi(x(t_f), t_f) + x^0(t_f)$$

sendo que se deve acrescentar uma restrição de igualdade (Equação 3.4) e uma condição inicial $x^0(t_0)$ no problema de otimização.

Também se pode transformar um problema de *Bolza* como problema de *Lagrange*. Se formular o problema de *Bolza* usando estados adicionais, então se pode definir um novo vetor de estados z onde:

$$z(t) = \begin{bmatrix} x^0(t) & x(t)^T \end{bmatrix}^T = \begin{bmatrix} x^0(t) & x^1(t) & \cdots & x^{nx}(t) \end{bmatrix}^T$$

sendo $z(t) \in \Re^{nx+1}, t \in [t_0, t_f]$ e $x_0(\cdot)$ é uma função contínua tal que:

$$x^0(t) = \frac{d\phi(x(t), t)}{dt} \qquad \text{e} \qquad \dot{x}^0(t_0) = 0 \tag{3.5}$$

Desta forma, pode-se dizer que:

$$\phi\left(x\left(t_f\right),t_f\right)=\int_{t_0}^{t_f}x^0\left(t\right)dt$$

Pode-se reescrever o problema de ***Lagrange***, na seguinte forma:

$$J\left(u(t)\right)=\int_{t_0}^{t_f}\left\{\varphi\left(x(t),u(t),t\right)+x^0\left(t\right)\right\}dt$$

Formulação das Restrições

As restrições do problema de otimização são de duas naturezas. Elas são representadas pelo modelo do processo e restrições de processo. Nas restrições do modelo, suas equações devem ser respeitadas ao longo de todo o horizonte de otimização. Nas restrições de processo, elas podem ser descritas como: restrições de caminho, de pontos interiores e terminais.

Uma vez definida a estrutura básica do problema de otimização, deve-se construir o modelo do processo, onde se descreve o seu comportamento dinâmico. O modelo pode considerar os aspectos macroscópicos ou microscópicos do processo, ser estático ou dinâmico, conter equações de propriedades físicas e outras restrições de igualdade. O modelo também pode descrever os aspectos de segurança, qualidade de produto, eficiência de equipamento, controladores, dentre outros.

É muito importante que se mantenha o modelo o mais fiel possível à planta real. Além disso, deve-se ter em mente que o modelo irá exigir um esforço computacional por parte do otimizador e também deverão ser evitados problemas de convergência. Portanto, deve-se utilizar o modelo na forma mais simples possível, com qualidade necessária, para resolver o problema de otimização e que tenha o mínimo de problemas de estabilidade numérica. Em alguns casos, isto exige que se faça uma redução do modelo.

O **modelo dinâmico do processo** é descrito por um conjunto de equações e de condições de contorno que definem um estado do processo.

Processos químicos são geralmente modelados dinamicamente usando equações diferenciais originadas de leis de conservação (balanços materiais, de energia, de momento) e por equações algébricas que definem relações físicas e termodinâmicas, além de equações constitutivas do processo (ex.: equações de equilíbrio, cinéticas, etc.), formando um conjunto de equações algébrico-diferenciais (*DAE's – Differential-Algebraic Equations*). O modelo pode ser representado na forma de equações explícitas, semi-implícitas e implícitas. A forma na qual o modelo deve ser apresentada está diretamente ligado à maneira com a qual o algoritmo de otimização foi implementado.

Considerando a solução de problemas de otimização dinâmica, o modelo pode ser representado na forma de equações explícitas:

$$\dot{x}(t)=F\left(x(t),\,y(t),u(t),t,\,p\right) \qquad (3.6a)$$

$$y(t)=G\left(x(t),\,y(t),u(t),t,\,p\right) \qquad (3.6b)$$

ou também representado na forma de equações implícitas:

$$F(\dot{x}(t), x(t), y(t), u(t), t, p) = 0$$
$$G(x(t), y(t), u(t), t, p) = 0$$

ou genericamente na forma de *DAE* agregada como:

$$F(\dot{x}(t), x(t), y(t), u(t), t, p) = 0 \tag{3.7}$$

Caso o algoritmo de otimização utilize uma forma diferente do modelo disponível, deve-se transformá-lo para uma forma adequada ao tal algoritmo. Por exemplo, se o modelo estiver representado na forma explícita, transforma-se na forma implícita apenas colocando a equação diferencial na sua forma residual.

Normalmente os modelos dinâmicos resultam em sistemas *DAE* de índice 1. Caso o problema tenha índice superior, recomenda-se que o sistema *DAE* seja reformulado para o índice 1 (Ascher e Petzold, 1998). O sistema *DAE* pode conter equações diferenciais de ordem superior. Neste caso, as equações de ordem superior devem ser transformadas em equações diferenciais de primeira ordem, pois a maioria dos algoritmos de otimização dinâmica resolve apenas sistema de equações diferenciais de primeira ordem. A transformação do sistema de equações diferenciais de ordem superior pode ser transformada em primeira ordem introduzindo um vetor de variáveis de estado aumentado. Por exemplo:
Considere um sistema de segunda ordem na forma semi-implícita

$$\ddot{x}(t) = F(x(t), u(t), p, t)$$

Este sistema é equivalente ao sistema de primeira ordem na forma

$$\dot{x}(t) = v(t)$$
$$\dot{v}(t) = F(x(t), u(t), p, t)$$

Onde o vetor de variável de estado aumentado é $\tilde{x} = \begin{bmatrix} x^T & v^T \end{bmatrix}^T$.

As **condições iniciais** definem os estados a partir do qual o modelo deve ser integrado no tempo. Usualmente são representados juntamente com o modelo do processo. Note que, as variáveis de estado (diferenciais) devem ser contínuas, mas as variáveis de saída (algébricas) e de entrada (controle) podem não ser contínuas. Portanto, as condições de iniciais podem ser escritas como:

$$I(x(t_0), y(t_0), p) = I^0 \tag{3.8}$$

onde I^0 são as condições iniciais especificadas. Ou ainda de forma mais simplificada como:

$$x(t_0) = x_0$$

Há caso em que se deseja limitar o processo ao longo de todo horizonte de otimização. Muitas vezes o problema de otimização tem sua região viável de operação limitada em função de restrições operacionais ou segurança. As **restrições de caminho** podem ser referentes a variáveis ligadas diretamente ao objetivo de otimização (ex.: máximo investimento permitido), outras restrições ligadas a limitações de equipamentos (ex.: máxima vazão da bomba), a aspectos de segurança (ex.: máxima pressão no reator), ao meio ambiente (ex.: máxima carga de poluentes permitida). Estas restrições podem ser, por exemplo, limites de especificações do processo ou até mesmo alguma relação imposta ao processo. Estas restrições podem ser impostas às variáveis de estado ou de controle envolvidos no problema. Também podem estar distribuídas no tempo, podendo ser restrições ao longo do horizonte de otimização ou no instante final (restrições terminais). Além disso, têm-se restrições de domínios das variáveis de estado e controle bem como dos parâmetros do modelo.

As restrições podem ser apresentadas na formas de equações ou inequações do sistema, e de fronteiras ou domínios de variáveis e parâmetros do problema de otimização. As restrições de fronteiras são inequações definidas pelo usuário de forma a estabelecer os limites do processo impostos às variáveis e parâmetros do problema de otimização. Estes limites podem ser também, por exemplo, limites de especificações do processo e equipamentos ou até mesmo alguma relação imposta ao processo. Além disso, os limites podem ser de domínios das variáveis que definem as regiões viáveis das mesmas.

As restrições de igualdade têm uma formulação idêntica às equações do modelo do processo. Por este motivo, estas equações são comumente embutidas no modelo.

As restrições de desigualdade podem ser escritas como:

$$S^L \le S(x(t), y(t), u(t), p, t) \le S^U \qquad \forall t \in [t_0, t_f] \tag{3.9}$$

onde:

S – conjunto de restrições de desigualdade do modelo de otimização;

S^L, S^U – limites inferiores e superiores destas restrições.

É importante ressaltar que os limites superiores e inferiores podem ser variáveis ao longo do tempo, por isso os mesmos são funções do tempo. O usual é considerarmos que seja constante num determinado intervalo de tempo.

As restrições de desigualdade podem ainda ser escritas mais genericamente como:

$$S(x(t), y(t), u(t), p, t) \le 0 \qquad \forall t \in [t_0, t_f] \tag{3.10}$$

Dentre as mais variadas formas de restrições de desigualdade em um problema de otimização dinâmica, destacam-se as restrições de limites ou fronteiras. As restrições de fronteira são impostas tanto para as variáveis de controle, que estão relacionadas a atuadores (ex.: vazões, pressões, etc.), como para as variáveis de estado, que estão ligadas a restrições operacionais de segurança ou desempenho (ex.: temperatura, recuperação), bem como limites de especificação de produtos (ex.: pureza, concentração).

As restrições de limites podem ser escritas como:

$$x^L \le x(t) \le x^U \qquad \forall t \in [t_0, t_f] \tag{3.11a}$$

$$y^L \le y(t) \le y^U \qquad \forall t \in [t_0, t_f] \tag{3.11b}$$

$$u^L \le u(t) \le u^U \qquad \forall t \in [t_0, t_f] \tag{3.11c}$$

onde:

x^L, x^U – limites inferiores e superiores das variáveis diferenciais;

y^L, y^U – limites inferiores e superiores das variáveis algébricas;

u^L, u^U – limites inferiores e superiores das variáveis de controle;

É usual, na solução de problemas de otimização, a transformação de uma restrição de desigualdade *S* em uma variável de estado algébrica ν, onde:

$$\nu(t) = S(x(t), y(t), u(t), p, t) \qquad \forall t \in [t_0, t_f]$$

$$\nu^L \le \nu(t) \le \nu^U \qquad \forall t \in [t_0, t_f]$$

Portanto, o vetor de variáveis algébricas pode ser aumentado na forma $\tilde{y} = \begin{bmatrix} y^T & \nu^T \end{bmatrix}^T$.

Com o objetivo de simplificar a representação do problema, podem-se aglutinar todas as restrições de desigualdade das variáveis de estado na seguinte forma:

$$S^L \le S(x(t), y(t), u(t), p, t) \le S^U \quad \text{ou} \quad S(x(t), y(t), u(t), p, t) \le 0 \quad \forall t \in [t_0, t_f] \tag{3.12}$$

Nos problemas de otimização onde o tempo final está livre, a variável tempo passa a ser uma variável de estado e seus limites são estabelecidos como:

$$t_f^L \le t_f \le t_f^U, \quad \forall t \in [t_0, t_f] \tag{3.13}$$

onde:

t_f^L, t_f^U – limites inferior e superior do tempo final.

Há situações em que se deseja limitar algumas variáveis em alguns instantes de tempo específicos dentro do horizonte de otimização. Algumas receitas de processo podem exigir que se tenha um determinado estado ou que esteja num determinado intervalo em certos momentos do processo. Estas **restrições de pontos interiores** passam a ser apresentadas como:

$$v_{\min}(t_I) \le v(t_I) \le v_{\max}(t_I) \qquad \forall t_I \in I = \{t_1, t_2, \cdots, t_{NI}\}$$

onde t_I é um instante de tempo dentro do horizonte de otimização.

Com o objetivo de simplificar a representação do problema, podem-se aglutinar as restrições da mesma forma anteriormente apresentada:

$$S_I(t_I)^L \le S_I(x(t_I), y(t_I), u(t_I), p) \le S_I(t_I)^U \text{ ou } S_I(x(t_I), y(t_I), u(t_I), p) \le 0 \quad \forall t_I \in I \qquad (3.14)$$

É comum estabelecer **restrições terminais** no problema, pois normalmente se especificam os valores mínimos e máximos para as variáveis de controle e de estado no final da operação de um sistema. As situações mais comuns são os limites de especificações de produtos, inventários ou de condições finais de processo. Além disso, podem-se ter restrições terminais, ligadas a desempenho, tais como: conversão, volume total produzido, dentre outras. Há problemas de otimização onde se deseja que algumas variáveis de estado, de controle ou uma composição delas tenham uma determinada posição no tempo final ou que as mesmas estejam em determinados intervalos no instante final:

$$T^L \le T(x(t_f), y(t_f), u(t_f), p, t_f) \le T^U \text{ ou } \qquad T(x(t_f), y(t_f), u(t_f), p, t_f) \le 0 \qquad (3.15)$$

Há casos onde os parâmetros são definidos como variáveis de decisão. Neste caso, os parâmetros livres são limitados na forma:

$$p^L \le p \le p^U \qquad (3.16)$$

onde:

p^L, p^U – limites inferiores e superiores dos parâmetros;

Juntando a função objetivo, as variáveis de controle e as restrições, a **formulação completa do problema de otimização dinâmica de simples estágio** pode ser representada genericamente da seguinte forma:

$$\min_{x(t), y(t), u(t), p, t_f} \phi(x(t_f), y(t_f), p, t_f) + \int_{t_0}^{t_k} \varphi(x(t), y(t), u(t), p, t) dt \qquad (3.17)$$

s.a.

$$F(\dot{x}(t), x(t), y(t), u(t), p, t) = 0; \qquad x(t_0) = x_0 \qquad t \in \lfloor t_0, t_f \rfloor$$
$$S^L \le S(x(t), y(t), u(t), p, t) \le S^U$$
$$S_I(t_I)^L \le S_I(x(t_I), y(t_I), u(t_I), p) \le S_I(t_I)^U \quad \forall t_I \in I = \{t_1, t_2, \cdots, t_{NI}\}$$
$$T^L \le T(x(t_f), y(t_f), u(t_f), p, t_f) \le T^U$$
$$u^L \le u(t) \le u^U$$
$$p^L \le p \le p^U$$
$$t_f^L \le t_f \le t_f^U$$

3.1.1.2 Problema de Múltiplos Estágios

Existem alguns problemas de otimização dinâmica que podem ser otimizados apenas em uma etapa (simples estágio), mas há outros conjuntos de problemas cujo procedimento de solução depende de alguma receita. Um exemplo é a otimização de um problema com alguma transição de processo (Figura 3.3). Neste caso, podemos ter um objetivo no bloco inicial de produção, que maximiza a produção, uma fase de transição, que minimiza o tempo de transição e a quantidade de produto fora de especificação e um segundo bloco onde podemos especificar a pureza do produto. Nestes casos, temos mais de um estágio

para resolver o problema ao longo do horizonte de otimização. Esses fatos podem justificar a representação do problema de otimização dinâmica na forma de múltiplos estágios.

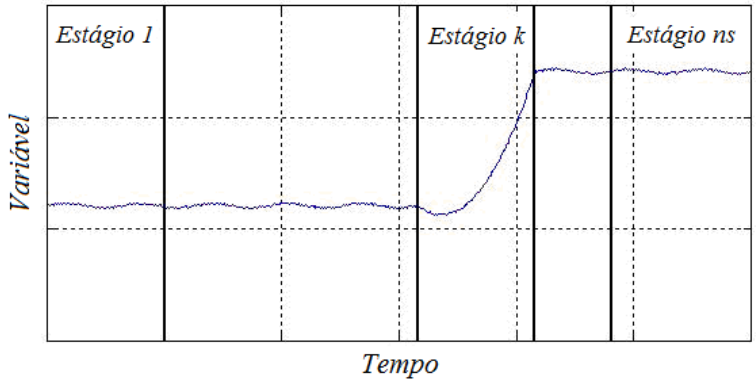


Figura 3.3 – estágios e transições de processo.

Pode-se definir o conjunto de estágios como:

$$\text{Estágio } k \in I = \{1, \ldots, ns\} \text{ onde } t \in [t_{k-1}, t_k), \quad k = 1,\ldots,ns-1 \text{ e } t \in [t_{ns-1}, t_f], \; k = ns$$

sendo:

ns – número de estágios (elementos);
t_0 – tempo no início do primeiro estágio;
t_k – tempo no final do estágio k ou início do estágio k+1;
t_f – tempo no final do último estágio.

Formulação da Função Objetivo

Para processos com múltiplos estágios (ex.: aquecimento, reação e resfriamento), a função objetivo é dividida em ***ns*** estágios. Com isso, podem-se ter objetivos no final de cada estágio ou ao longo de cada estágio, tempos finais livres ou fixos, pode envolver variáveis diferenciais (estados), algébricas (saídas) e de controles (entradas), além de parâmetros e tempo final.

Formulação das Restrições

As restrições envolvidas num problema de otimização dinâmica de múltiplos estágios também incluem o modelo dinâmico do processo, restrições de caminho, restrições de pontos interiores, restrições terminais e restrições nos parâmetros em cada estágio e restrições de junção entre estágios adjacentes.

Quando se tem um problema de múltiplos estágios, as condições de contorno se transformam em condições de continuidade das variáveis de estado. De forma mais simplificada, podemos escrever:

$$x_1(t_0) = x_0 \qquad \textit{Condições iniciais}$$

$$x_k(t_k) = x_{k+1}(t_k) \qquad k = 1,\cdots,ns-1 \qquad \textit{Condições de continuidade entre estágios}$$

Estas condições de continuidade, chamadas de **restrições de junção**, podem ser escritas genericamente da seguinte forma:

$$J_1\left[x_1(t_o), y_1(t_o), u_1(t_o), p_1, t_o\right] = 0 \tag{3.18a}$$

$$J_{k+1}\left[x_k(t_k), y_k(t_k), u_k(t_k), x_{k+1}(t_k), y_{k+1}(t_k), u_{k+1}(t_k), p_{k+1}, t_k\right] = 0 \quad k = 1, \cdots, ns-1 \tag{3.18b}$$

Em relação ao restante das restrições, ela tem o mesmo tratamento recebido no problema de simples estágio. Com isso, a **formulação completa do problema otimização dinâmica de múltiplos estágios** pode ser escrita na forma da Equação 3.19.

$$\min_{x_k(t), y_k(t), u_k(t), p_k, t_k} \sum_{k=1}^{ns}\left[\phi_k\left(x_k(t_k), y_k(t_k), p_k, t_k\right) + \int_{t_{k-1}}^{t_k} \varphi_k\left(x_k(t), y_k(t), u_k(t), p_k, t\right) dt\right] \tag{3.19}$$

s.a.

$$F_k\left(\dot{x}_k(t), x_k(t), y_k(t), u_k(t), p_k, t\right) = 0 \qquad t \in [t_{k-1}, t_k) \quad k = 1, \cdots, ns-1 \text{ e}$$
$$t \in [t_{k-1}, t_k] \quad k = ns$$

$$S_k{}^L \le S_k\left(x_k(t), y_k(t), u_k(t), p_k, t\right) \le S_k{}^U$$

$$S\left(t_I^{(k)}\right)^L \le S\left(x\left(t_I^{(k)}\right), y\left(t_I^{(k)}\right), u\left(t_I^{(k)}\right), p_k\right) \le S\left(t_I^{(k)}\right)^U$$

$$T_k^L \le T_k\left(x_k(t_k), y_k(t_k), u_k(t_k), p_k, t_k\right) \le T_k^U$$

$$J_1\left[x_1(t_o), y_1(t_o), u_1(t_o), p_1, t_o\right] = 0$$

$$J_{k+1}\left[x_k(t_k), y_k(t_k), u_k(t_k), x_{k+1}(t_k), y_{k+1}(t_k), u_{k+1}(t_k), p_{k+1}, t_k\right] = 0 \quad k = 1, \cdots, ns-1$$

$$u_k{}^L \le u_k(t) \le u_k{}^U$$

$$p_k{}^L \le p_k \le p_k{}^U$$

$$t_k{}^L \le t_k \le t_k{}^U$$

3.1.1.3 Representação Apropriada do Problema

Uma forma geral de representação do problema de otimização dinâmica deve ser comumente empregada pelos softwares de otimização dinâmica. A formulação do problema deve ter a flexibilidade para representar a maioria dos casos de otimização de processos e controle encontrados em plantas químicas. As seguintes alterações são normalmente aplicadas à formulação genérica.

Representação da Função Objetivo

Usualmente, o termo de Lagrange é transformado em termo *Mayer* definindo uma nova variável diferencial no modelo de otimização como sendo o integrando da parcela de Lagrange. Isto pode ser representado na seguinte forma:

$$\dot{z}_\varphi = \varphi(x(t), y(t), u(t), p, t); \qquad z_\varphi(0) = 0 \tag{3.20}$$

E a parcela Mayer é transformada em uma variável de estado, podendo ser uma variável diferencial ou algébrica no instante final. Usualmente é uma variável diferencial. Portanto, podemos fazer isso, acrescentando as seguintes equações no modelo de otimização:

$$\dot{z}_\phi = \dot{\phi}(x(t), y(t), p, t); \qquad z_\phi(0) = z_\phi^{\,0} \qquad \text{para variáveis diferenciais}$$

ou

$$w_\phi = \phi(x(t), y(t), p, t) \qquad \text{para variáveis algébricas} \qquad (3.21)$$

Note que, se ϕ for uma variável diferencial x ou algébrica y já existente, não há a necessidade de acrescentar quaisquer equações no modelo de otimização.

Convém acrescentar também que, se o tempo final (t_f) for uma variável livre na função objetivo do modelo de otimização ($\phi(t_f) = t_f$), uma nova variável diferencial deve ser acrescentada no modelo de otimização na forma:

$$\dot{z}_\phi = 1; \qquad z_\phi(0) = 0 \qquad (3.22)$$

Finalmente, pode-se dizer que o novo vetor aumentado de variáveis diferenciais toma a forma $z(t)^T = \left[x(t)^T \quad z_\varphi(t)\right]^T$ e de variáveis algébricas a forma $w(t)^T = \left[y(t)^T \quad w_\varphi(t)\right]^T$.

Representação das Restrições

As **equações do modelo** do processo não sofrem quaisquer alterações. A única conseqüência é que um conjunto de equações diferenciais e algébricas é acrescentado ao modelo de otimização quando a função objetivo ou alguma restrição sofre alterações na sua forma de apresentação. As **restrições de caminho, restrições de pontos interiores**, **restrições terminais**, **restrições de junção** e **restrições nos parâmetros** não sofrem alterações. A diferença principal, para as **restrições nos tempos finais** dos estágios, está na representação na restrição de limites do tempo final, quando a mesma for uma variável de decisão livre. Computacionalmente é preferível reformular um problema de tempo final livre para tempo final fixo. A transformação pode ser feita de forma que o tempo final t_f seja tratado como um parâmetro do sistema e o tempo do sistema é substituído por $t = t_f.\tau$, onde $\tau \in [0,1]$ é o tempo normalizado, reescrevendo as equações do modelo em função desta nova variável independente τ.

Representação Completa do Problema

De forma mais simples, podemos representar o problema de otimização dinâmica na seguinte forma:

Formulação do problema de simples estágio

$$\min_{z(t),u(t),p} \Phi\left(z(t_f), w(t_f)\right) \tag{3.23a}$$

s.a.

$$F\left(\dot{z}(t), z(t), w(t), u(t), p, t\right) = 0 \qquad t \in \left]t_0, t_f\right]$$
$$z(t_0) = z_0$$
$$S^L \le S\left(z(t), w(t), u(t), p, t\right) \le S^U \qquad t \in \left]t_0, t_f\right]$$
$$S_I(t_I)^L \le S_I\left(z(t_I), w(t_I), u(t_I), p\right) \le S_I(t_I)^U \qquad \forall t_I \in I = \{t_1, t_2, \cdots, t_{NI}\}$$
$$T^L \le T\left(z(t_f), w(t_f), u(t_f), p, t_f\right) \le T^U$$
$$u(t)^L \le u(t) \le u(t)^U$$
$$p^L \le p \le p^U$$
$$t_f^L \le t_f \le t_f^U$$

onde:

$$F^T = \left[F^T \quad F_\varphi \quad F_\phi\right]^T$$
$$z^T = \left[x^T \quad z_\varphi \quad z_\phi\right]^T$$
$$w^T = \left[y^T \quad w_\phi\right]^T$$

ou

Formulação do problema de múltiplos estágios

$$\min_{z_k(t),u_k(t),p_k} \sum_{k=1}^{ns} \Phi_k\left(z(t_k), w(t_k)\right) \tag{3.23b}$$

s.a.

$$F_k\left(\dot{z}_k(t), z_k(t), w_k(t), u_k(t), p_k, t\right) = 0 \qquad t \in \left[t_{k-1}, t_k\right) \quad k = 1, \cdots, ns-1 \text{ e}$$
$$t \in \left[t_{k-1}, t_k\right] \quad k = ns$$
$$S_k{}^L \le S_k\left(x_k(t), y_k(t), u_k(t), p_k, t\right) \le S_k{}^U \qquad t \in \left[t_{k-1}, t_k\right] \quad k = 1, \cdots, ns$$
$$S\left(t_I^{(k)}\right)^L \le S\left(z\left(t_I^{(k)}\right), w\left(t_I^{(k)}\right), u\left(t_I^{(k)}\right), p_k\right) \le S\left(t_I^{(k)}\right)^U$$
$$T_k^L \le T_k\left(z_k(t_k), w_k(t_k), u_k(t_k), p_k, t_k\right) \le T_k^U$$
$$J_1\left[z_1(t_o), w_1(t_o), u_1(t_o), p_1, t_o\right] = 0$$
$$J_{k+1}\left[z_k(t_k), w_k(t_k), u_k(t_k), z_{k+1}(t_k), w_{k+1}(t_k), u_{k+1}(t_k), p_{k+1}, t_k\right] = 0 \quad k = 1, \cdots, ns-1$$
$$u_k{}^L \le u_k(t) \le u_k{}^U$$
$$p_k{}^L \le p_k \le p_k{}^U \qquad k = 1, \cdots, ns$$
$$t_k{}^L \le t_k \le t_k{}^U$$

onde:

$$F_k{}^T = \left[F^{(k)T} \quad F_\varphi^{(k)} \quad F_\phi^{(k)}\right]^T,\ z_k{}^T = \left[x^{(k)T} \quad z_\varphi^{(k)} \quad z_\phi^{(k)}\right]^T \text{ e } w_k{}^T = \left[y^{(k)T} \quad w_\phi^{(k)}\right]^T.$$

3.1.1.4 Classes de Problemas

Um problema de otimização dinâmica tem algumas formulações diferentes em função de sua finalidade. Estas funções objetivo podem ser classificadas em quatro grandes categorias, em função de suas aplicações. São elas:

- Aplicações de controle ótimo ou *MPC* (*Model-based Predictive Control*);
- Aplicações de detecção de erros grosseiros e reconciliação de dados;
- Aplicações de ajustes de parâmetros do modelo do processo.
- Aplicações de otimização dinâmica ou *DRTO* (*Dynamic Real-Time Optimization*);

O ***problema de mínimo tempo*** (Kirk, 2004) consiste em transferir o sistema de um estado inicial para uma condição final no mínimo tempo. É um caso particular do controle ótimo, onde o problema tem a seguinte forma:

$$\min_{u(t)} \left\{ t_f = \int_0^{t_f} dt \right\} \tag{3.24}$$

$$\begin{aligned} s.a.\ \ & \dot{x}(t) = F\left(x(t), u(t), t\right) \\ & S(x(t), u(t), t) \le 0 \\ & T\left(x(t_f), u(t_f), t_f\right) \le 0 \end{aligned}$$

onde *x(t)* são as variáveis de estado e *u(t)* as variáveis de controle no instante de tempo t.

O ***problema de controle terminal*** (Kirk, 2004) consiste em minimizar o desvio do estado final de um valor de referência. Este é um caso típico de problema de balística, onde o objetivo é atingir o alvo.

Para um tempo final (t_f), levar o sistema para o alvo desejado, a função objetivo deve ser tal que:

$$\min_{u} \left\{ J = \left\| x\left(t_f\right) - x^{SP}\left(t_f\right) \right\|^2 \right\} \tag{3.25}$$

A forma quadrática do índice de desempenho se deve ao fato dos desvios poderem assumir valores positivos e negativos, sendo desejável punir quaisquer desvios.

O ***problema de mínimo consumo*** (Kirk, 2004) consiste em transferir o sistema de um estado inicial para uma condição final com o mínimo esforço das variáveis de controle. É um caso particular do controle ótimo, onde a função objetivo tem a seguinte forma:

$$\min_{u} \left\{ J = \int_0^{t_f} \left\| u(t) \right\| dt \right\} \tag{3.26}$$

O ***problema de rastreamento*** (Kirk, 2004) consiste em manter o sistema o mais próximo possível do valor de referência ao longo do horizonte de otimização. É um caso particular do controle ótimo, onde a função objetivo tem a seguinte forma:

$$\min_{u} \left\{ J = \int_0^{t_f} \left\| x(t) - x^{SP}(t) \right\|^2 dt \right\} \tag{3.27}$$

O ***problema de controle ótimo ou MPC*** (*Model Predictive Control*) pertence a uma classe de aplicações importantes de soluções de problemas de otimização dinâmica nas indústrias de processamento. Quando o modelo do processo for não-linear, se torna uma aplicação de *NMPC* (*Nonlinear Model Predictive Control*). Este tipo de aplicação tem recebido maior atenção por parte dos pesquisadores e da indústria.

Normalmente, estes tipos de controladores têm uma função objetivo semelhante aos encontrados nos controladores *LQ - Linear Quadratic* (Qin e Badgwell, 2003) da teoria de controle ótimo. A forma mais comumente encontrada é a do problema de mínimo erro nas variáveis de estado e mínimo movimento das variáveis de controle. Neste caso, pode-se escrever a função objetivo na seguinte forma:

$$J\left(x(t), u(t), t_f\right) = \int_0^{t_f} \left[\left(x - x_{SP}\right)^T Q \left(x - x_{SP}\right) + \left(u - u_{SS}\right)^T R \left(u - u_{SS}\right) \right] d\tau \tag{3.28}$$

onde: Q são os pesos de importâncias das variáveis de estado ou saída a serem controladas e R os fatores de supressão de movimentos das variáveis de controle.

Se colocar restrições de ponto final nas saídas, tem-se a estabilidade garantida (Mayne e Michalska, 1990):

$$J\left(x(t), u(t), t_f\right) = \int_0^{t_f} \left\{ \left\| x(\tau) - x_{SP} \right\|_Q^2 + \left\| u(\tau) - u_{SS} \right\|_R^2 \right\} d\tau + \left\| x(t_f) - x_{SP} \right\|_P^2 \tag{3.29}$$

Como dito anteriormente, as funções objetivo são semelhantes e podem ser usadas tanto para **reconciliação de dados** como para **ajuste de parâmetros**. Normalmente o índice de desempenho é a função de máxima verossimilhança. Porém se considerar que todas as medidas têm a mesma variância reduz-se a um problema de mínimos quadrados. Portanto, os casos mais comuns em otimização são os seguintes:

Na ***Reconciliação de Dados*** (Arora, 2003), pode-se formular o problema de otimização na seguinte forma:

$$\min_{z} \left\{ J = e^T \Sigma^{-1} e = \left(\hat{z} - z\right)^T \Sigma^{-1} \left(\hat{z} - z\right) \right\} \tag{3.30}$$

Sujeito a:

$$f\left(\dot{x}, x, z, p\right) = 0 \quad , \quad x(0) = x_0$$

$$z_{\min} \le z \le z_{\max}$$

onde x são as variáveis de estado, $\hat{z}$ são as variáveis medidas da planta, $z = [y^T \quad u^T]^T$ as variáveis reconciliadas, y as variáveis dependentes, u as variáveis independentes, p são os parâmetros do modelo do processo e $e^T\Sigma^{-1}e$ representa a função de máxima verossimilhança do modelo do processo.

$$\Sigma = diag\left(\left[\frac{1}{\sigma_{1,1}^2} \quad \cdots \quad \frac{1}{\sigma_{N,1}^2} \quad \cdots \quad \frac{1}{\sigma_{1,NE}^2} \quad \cdots \quad \frac{1}{\sigma_{N,NE}^2}\right]\right)$$

onde:

σ_i^2 – variâncias das variáveis medidas da planta;

NE – número de experimentos;

N – número de variáveis medidas do modelo.

Se utilizar o método dos mínimos quadrados, tem-se:

$$\min_{x,z} \left\{J = e^T e = \left(\hat{z} - z\right)^T \left(\hat{z} - z\right)\right\} \tag{3.31}$$

O problema de ***ajustes de parâmetros*** (Arora, 2003) consiste em determinar os valores dos parâmetros do modelo do processo de forma que o mesmo seja o mais fiel possível à planta, isto é, que os valores estimados no modelo estejam os mais próximos possíveis das medidas da planta. A formulação do problema de ajustes de parâmetros é semelhante ao de reconciliação de dados. Por esta razão, pode-se realizar a reconciliação de dados juntamente com o ajuste de parâmetros ou separadamente, pois as medidas do processo podem ser as mesmas para ambos os problemas.

Neste caso, o problema é o seguinte:

$$\min_{p} \left\{J = e^T\Sigma_y^{-1}e = \left(\hat{y} - y\right)^T \Sigma_y^{-1} \left(\hat{y} - y\right)\right\} \tag{3.32}$$

Sujeito a:

$$f\left(\dot{x}, x, y, u, p\right) = 0 \quad , \quad x(0) = x_0$$

$$p_{\min} \le p \le p_{\max}$$

onde x são as variáveis de estado, u são as variáveis independentes medidas na planta, $\hat{y}$ as variáveis dependentes medidas na planta, y as variáveis dependentes calculadas, p são os parâmetros do modelo do processo.

$$\Sigma_y = diag\left(\left[\frac{1}{\sigma_{1,1}^2} \quad \cdots \quad \frac{1}{\sigma_{ny,1}^2} \quad \cdots \quad \frac{1}{\sigma_{1,NE}^2} \quad \cdots \quad \frac{1}{\sigma_{ny,NE}^2}\right]\right)$$

Se utilizar o método dos mínimos quadrados, tem-se:

$$\min_{p}\left\{J = e^T e = \left(\hat{y} - y\right)^T \left(\hat{y} - y\right)\right\}$$

Na ***reconciliação de dados e ajuste de parâmetros simultâneos*** (Arora, 2003), o problema é o seguinte:

$$\underset{z,p}{Min} \ \left(\hat{z}-z\right)^T \Sigma^{-1} \left(\hat{z}-z\right) \tag{3.33}$$

Sujeito a:

$$f\left(\dot{x}, x, z, p\right) = 0 \quad , \quad x(0) = x_0$$
$$z_{\min} \le z \le z_{\max}$$
$$p_{\min} \le p \le p_{\max}$$

onde x são as variáveis de estado, $\hat{z}$ são as variáveis medidas da planta, $z = [y^T \ \ u^T]^T$ as variáveis reconciliadas, y as variáveis dependentes, u as variáveis independentes e p são os parâmetros do modelo do processo.

Nas **aplicações de otimização dinâmica ou *DRTO***, os problemas de otimização dinâmica têm uma grande diversidade de objetivos, em função da sua aplicação, ou seja, o tipo de sistema (ex.: reator em batelada, torre de destilação contínua, etc.) e sua finalidade (ex.: minimizar consumo de energia, minimizar o tempo de transição da operação). Há uma série de objetivos comuns em processos químicos:

- Alcançar o estado do sistema o mais próximo possível de um valor desejado no tempo final t_f;
- Minimizar o tempo para atingir um determinado estado desejado e mantê-lo na posição (minimiza o tempo de transição);
- Minimizar a energia para atingir um determinado estado desejado em t_f;
- Maximizar ou minimizar índices de desempenho específicos como lucro, recuperação de produto, eficiência, produção total, etc.

Em função desta grande diversidade de objetivos, não há formas típicas de função objetivo em problemas de *DRTO*, a não ser que se tenha apenas finalidade de controle ótimo. Portanto, deve-se utilizar a função objetivo na sua forma geral de *Bolza*. É freqüente encontrar a função de *Mayer* aplicada em problemas otimização dinâmica, sendo que o termo da função de *Lagrange* é normalmente convertido no termo de *Mayer*.

Além da otimização dinâmica contínua, um problema importante é a **otimização dinâmica inteira mista (*MIDO - Mixed-Integer Dynamic Optimization*)**. O mundo real de processamento químico tem freqüentemente **operações contínuas e discretas (hibridas ou combinadas)**. As transições de processos são comuns, onde podem ser alteradas restrições de processo, executadas manobras de válvulas (abrir/fechar) e bombas (ligar/ desligar), ou até mesmo utilizar um fluxograma de processo diferente para cada tipo de operação (desviar um equipamento). Também é comum observar que determinados eventos podem ocorrer ao longo da operação da planta. Um tanque pode ser esvaziado ou completamente cheio, um determinado prato de coluna de destilação pode secar, pode-se purgar ou aliviar correntes de tempos em tempos, etc. Também podem ocorrer decisões discretas a serem tomadas nas plantas pelos operadores e sistemas de controle durante partidas, paradas ou até mesmo em emergências. Além disso, é comum projetar processos com decisões discretas, utilizando superestruturas de processo para realizar a otimização de

projetos. No entanto, há uma maior oportunidade de otimização quando se realiza otimização de projeto e controle simultaneamente. Nos últimos anos, tem crescido o interesse pela otimização dinâmica de sistemas híbridos. Biegler e Grossmann (2004) e Allgor e Barton (1999) tem chamado a atenção para esta nova oportunidade na otimização dinâmica, porém ainda não tem sido observadas aplicações em tempo real.

Diante da crescente importância da otimização dinâmica de sistemas híbridos na indústria química, a comunidade cientifica tem se motivado a desenvolver simuladores dinâmicos adequados aos propósitos acima citados (Park e Barton, 1997). A otimização de sistemas dinâmicos híbridos nem sempre pode ser realizada usando formulações puramente contínuas (Barton *et al.*, 1998). Os problemas em engenharia química formulados como problemas *MIDO* envolvem variáveis contínuas e discretas. As variáveis contínuas correspondem às decisões de projeto e condições operacionais relacionados com uma determinada estrutura do sistema a ser otimizado (como vazões, pressões, temperaturas, etc.). As variáveis discretas geralmente são usadas para modelar a seleção da unidade de operações em um fluxograma, decisões de seqüenciamento temporal (variáveis binárias) ou números de equipamentos, de trens de bateladas, etc. (variáveis inteiras). Com base nestas características de processamento e projetos de sistemas de engenharia química, um problema *MIDO* por ser formulado da seguinte forma:

$$
\begin{aligned}
&\min_{u(t),y(t),t_f} \quad J = \phi\left(x\left(t_f\right)\right) \\
&s.a. \\
&\qquad \dot{x} = F\left(x, y, u, p, t\right); \quad x\left(t_0\right) = x_0 \\
&\qquad S\left(x, y, u, p, t\right) \le 0 \\
&\qquad T\left(x\left(t_f\right)\right) \le 0 \\
&\qquad x \in \Re^{nx},\ y \in \{0,1\}^{ny}
\end{aligned}
\tag{3.34}
$$

Allgor e Barton (1999) propõem formular o problema *MIDO,* transformado-o em um problema de programação não-linear inteira-mista (*MINLP - Mixed-Integer NonLinear Programming*) de dimensão finita através da discretização no domínio do tempo com colocação ortogonal em elementos finitos. Flores-Tlacuahuac e Biegler (2007) relatam duas abordagens para lidar com a solução do problema *MIDO*. Algumas abordagens (Park e Barton, 1997; Bansal *et al.*, 2003) resolvem o problema *MIDO* como uma seqüência de problemas interno e externo. O problema interno é formulado como um problema de otimização dinâmica contínuo, enquanto o problema externo é formulado como um problema de programação linear inteira-mista (*MILP - Mixed-Integer Linear Programming*).

O problema de otimização dinâmica no sistema *DRTO* deve ser formulado, com a linguagem própria de cada software (*IHM – Interface Homem-Máquina*), o passo seguinte é a interpretação da linguagem e formulação padrão do *DAOP*. A etapa seguinte é a escolha do método de solução do problema. A primeira fase é a discretização do *DAOP* e formular o problema *NLP* depois de discretizado (Figura 3.4).

Figura 3.4 – pontos de vista de formulação de problema no sistema de *DRTO*.

É importante destacar a conveniência da representação do *DAOP* na sua forma padrão, pois isto simplifica a integração do ambiente de modelagem com os algoritmos de otimização dinâmica, bem como a versatilidade do uso de diferentes tipos de algoritmos de solução de *DAOP*. A transformação do *DAOP* não padrão em seu formato padrão deve ser uma tarefa interna do sistema de otimização dinâmica, e não uma tarefa do usuário (na construção do modelo de *DAOP*). Há alguns softwares de otimização dinâmica que impõem a descrição do *DAOP* na sua forma padrão.

3.1.2 Solução do problema de otimização dinâmica

Problemas de otimização dinâmica de dimensão infinita (*DAOP*) podem ser resolvidos por duas classes de métodos, a dos métodos diretos e a dos indiretos. Esta classificação foi inicialmente atribuída a Edelbaum (1962).

Os métodos indiretos resolvem o problema de otimização dinâmica através da satisfação das condições necessárias de otimalidade. Estas condições são originárias da teoria de controle ótimo e aplicação do cálculo variacional. Estes métodos também são conhecidos como métodos analíticos, por utilizar as condições de otimalidade para transformar o *DAOP* em um problema de duplo valor de contorno (*TPBVP – Two Point Boundary Value Problem*).

Há basicamente dois métodos importantes usando este tipo de abordagem:

- ***Programação Dinâmica*** (Bellman, 1957) – Baseado no princípio da otimalidade de Bellman e na solução das equações de Hamilton-Jacobi-Bellman (*HJB*);
- ***Princípio do Máximo de Pontryagin*** (Pontryagin *et al.*, 1962) – Baseado na solução das condições de otimalidade da teoria de controle ótimo.

A essência do método indireto está na localização das raízes do *TPBVP* (Betts, 2001). Vários métodos podem resolver este tipo de problema, podendo ser utilizados até mesmo os métodos diretos. Os métodos numéricos para resolver o *TPBVP*, comumente encontrados na literatura, são: métodos de *shooting*[1] (*single shooting* ou *multiple shooting*), métodos de quadratura (diferenças finitas, métodos dos resíduos ponderados, colocação em elementos finitos), método *invariant embedding* ou programação dinâmica.

[1] O método *single shooting* surgiu da necessidade de se resolver os problemas de controle ótimo na área de balística, onde se desejava calcular a trajetória ótima de mísseis do ponto de lançamento para atingir um determinado alvo. Daí vem o jargão utilizado "*shooting*".

Os métodos indiretos de solução de problemas de otimização dinâmica não são adequados para resolver problemas de dimensão elevada. Por esta razão, estes métodos têm sido mais usados quando é necessário computar trajetórias ótimas muito precisas, como nas áreas de aeronáutica, aeroespacial, balística e robótica (que são problemas de menor porte) do que na indústria de processos.

Como alternativa aos métodos indiretos, tem-se os métodos diretos que aplicam algum nível de discretização no *DAOP*, transformando-o em um problema de programação não-linear (*NLP – Non-Linear Programming*) de dimensão finita. O problema de *NLP* então pode ser resolvido por uma infinidade de algoritmos disponíveis na literatura (Ex.: *SQP*, *rSQP*, dentre outros). A escolha do algoritmo apropriado dependerá da dimensão e da natureza do problema de *NLP* a ser resolvido. Os métodos diretos são mais simples de implantar e codificar, pois não precisam das equações adicionais de co-estado.

Há basicamente dois níveis de discretização de variáveis, que subdivide os métodos diretos em dois grupos: métodos de discretização parcial e total (vide Figura 3.5).

Nos métodos de discretização parcial, apenas as variáveis de controle são discretizadas, e as variáveis de estado são obtidas por integração do sistema de equações algébrico-diferencial (*DAE – Differential Algebraic Equations*). Neste caso, define-se uma forma funcional para as variáveis de controle, escolhem-se os pontos discretos no intervalo de integração e um perfil inicial para estas variáveis, integra-se o modelo neste intervalo e com as variáveis de estado obtidas e as sensibilidades da função objetivo, o algoritmo de *NLP* procura a solução ótima. Nesta abordagem, somente as variáveis de controle são tratadas como variáveis de decisão pelo algoritmo de *NLP*, pois as variáveis de estado são resolvidas por integração do sistema *DAE*. Por esta razão, estes métodos também são chamados de métodos seqüenciais, pois se resolve o problema de otimização em uma seqüência de integração e otimização. Dentre os métodos de discretização parcial encontrados na literatura, pode-se destacar a programação dinâmica e os métodos de *shooting* (*single shooting* e *multiple shooting*). Nos métodos de *multiple shooting* também é realizada uma partição do intervalo de integração em subintervalos menores para poder explorar propriedades de processamento paralelo e reduzir problemas numéricos em sistemas instáveis, gerando variáveis adicionais de otimização para atender às condições de continuidade entre intervalos adjacentes.

Nos métodos de discretização total, além de discretizar as variáveis de controle, discretiza-se também as variáveis de estado, resultando em um sistema de equações algébricas não-lineares (*NLA – Non-Linear Algebraic System*), em que são resolvidos simultaneamente a integração do sistema *DAE* e o problema de *NLP* de grande dimensão. Por esta razão, estes métodos também são chamados de métodos simultâneos. Dentre os métodos de discretização total, encontrados na literatura, podem-se destacar os métodos de colocação (métodos dos resíduos ponderados), diferenças finitas e colocação em elementos finitos.

Os problemas de *NLP* gerados pelos métodos diretos simultâneos são de dimensão elevada. Felizmente, uma série de algoritmos de *NLP* (ex.: *SQP*, *rSQP, IP – Interior Point*) tem a capacidade de resolver problemas de otimização dessas dimensões. Por isso, estes métodos têm sido objetos de interesse da indústria de processos, resolvendo problemas de otimização dinâmica via *DRTO* e *NMPC*.

Recentemente, uma abordagem híbrida dos métodos diretos e indiretos tem tido atenção por parte dos pesquisadores (Gath, 2002).

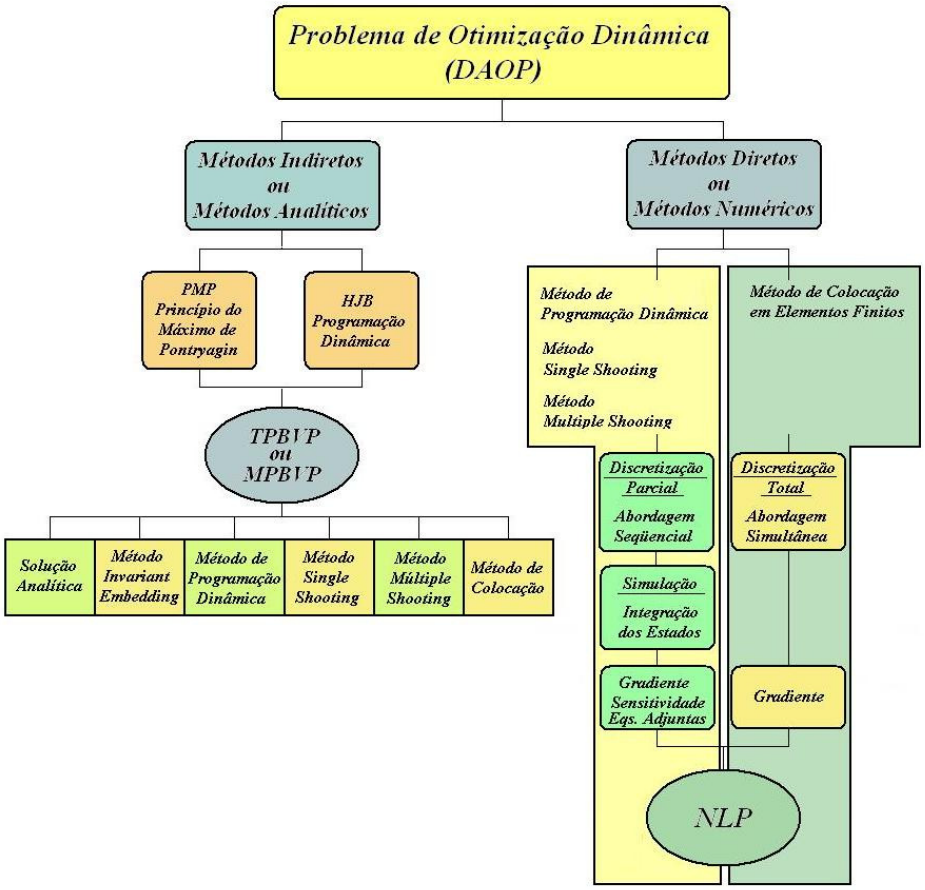


Figura 3.5 – Métodos de Solução de problemas de otimização dinâmica (*DAOP*).

3.1.2.1 Métodos indiretos

Estes métodos buscam o extremo de uma função objetivo pela aplicação das condições necessárias de otimalidade de 1ª ordem. Quando se tem um problema com restrições, utilizam-se os *multiplicadores de Lagrange* e variação restrita. Para certificar que a solução é um extremo, ela deve ser testada junto às condições necessária e suficiente de 2ª ordem.

Na abordagem clássica de solução de problemas de otimização dinâmica, eliminam-se as variáveis de controle expressando-as como função de variáveis de estado e co-estado (primal e dual). Para obter a solução do problema, normalmente as variáveis de estado são especificadas com condições iniciais e as variáveis adjuntas com condições finais e aplicam-se métodos iterativos que usam estimativas iniciais para encontrar a solução para o problema onde as condições necessárias de 1ª ordem devam ser satisfeitas. A solução obtida é usada para ajustar a estimativa inicial para tentar encontrar uma solução que esteja mais próxima de todas as condições necessárias.

Há duas categorias de métodos para resolver os *TPBVP's* gerados pelos métodos indiretos, são eles:

Iteração das Condições de Contorno (BCI – Boundary Conditions Iteration) procura encontrar as condições de contorno $\lambda(t_0)$ através da minimização do erro nas condições de contorno conhecidas $\lambda(t_f)$ por integração no tempo, para frente, das equações de estado e adjuntas. Nesta classe de métodos, podem-se incluir os métodos de programação dinâmica.

Iteração dos Vetores de Controle (CVI – Control Vector Iteration) onde as equações de estado são integradas no tempo para frente, usando perfis propostos para as variáveis de controles, e as equações adjuntas são então integradas no tempo para trás (mesma direção natural das condições de contorno de λ), para atualizar os perfis de controle propostos nos pontos discretos e atingir o critério de convergência. Nesta classe de métodos, pode-se incluir os métodos de *single shooting*, *multiple shooting* e colocação.

Os métodos de programação dinâmica transformam um problema de otimização dinâmica em um sistema de equações algébrico-diferenciais com valores de contorno, utilizando as equações de *Hamilton-Jacobi-Bellman* (*HJB*). A Formulação de *HJB* usa o principio da otimalidade de *Bellman* para transformar o problema de otimização de função objetivo escalar *J* em um sistema de equações diferenciais parciais (Kirk, 2004; Bryson & Ho, 1975). O princípio de otimalidade de *Bellman* pode ser formulado como: *"Se há um meio ótimo para ir de A a C, então há um meio parcial de B a C que também é ótimo"*.

Seja o problema geral de otimização dinâmica na forma de *Bolza* dado por:

$$\min_{x(t),u(t),t_f,p} J\left(x(t),u(t),t_f,p\right)=\phi\left(x\left(t_f\right),t_f\right)+\int_{t_0}^{t_f}\varphi\left(x(t),u(t),t,p\right)dt \tag{3.35}$$

sujeito a:

$$F\left(\dot{x}(t),x(t),u(t),t,p\right)=0 \qquad , \qquad \forall t\in T$$

Considere a seguinte função, também chamada de função de *Bellman*, definida como:

$$V\left(x(t),t,p\right)=\min_{u(t)}\left[\phi\left(x\left(t_f\right),t_f\right)+\int_{t}^{t_f}\varphi\left(x(t),u(t),\tau,p\right)d\tau\right] \tag{3.36}$$

onde ***V(x, t, p)*** é uma função equivalente ao mínimo custo se o sistema tem os estados *x* no tempo $t\le t_f$. Colocando os termos das restrições na formulação do Lagrangeano, tem-se:

$$\tilde{J}\left(x,u,t,p\right)=\phi\left(x\left(t_f\right),t_f\right)+\int_{t}^{t_f}\left\{\varphi\left(x,u,t,p\right)+\left(\frac{\partial V}{\partial x}\right)^{T}F\left(\dot{x},x,u,t,p\right)\right\}dt \tag{3.37}$$

O termo de Lagrange com o termo adjunto resulta na função escalar do Hamiltoniano *H* que é descrito na forma:

$$H\left(x,\dot{x},u,\left(\frac{\partial V}{\partial x}\right)^{T},t\right)=\varphi\left(x,u,t,p\right)+\left(\frac{\partial V}{\partial x}\right)^{T}F\left(\dot{x},x,u,t,p\right) \tag{3.38}$$

Colocando na forma diferencial em relação ao tempo no perfil ótimo u^*, obtém-se a equação diferencial parcial de *Hamilton-Jacobi-Bellman*, dada por:

$$-\frac{\partial V}{\partial t} = \min_{u(t)} H\left(\dot{x}, x, u, \left(\frac{\partial V}{\partial x}\right)^T, t\right)$$

ou

$$-\frac{\partial V}{\partial t} = \min_{u(t)}\left[\varphi(x,u,t,p) + \left(\frac{\partial V}{\partial x}\right)^T F(\dot{x},x,u,t,p)\right] \tag{3.39}$$

As condições de contorno definidas no tempo final t_f são:

$$-\left.\frac{\partial V}{\partial t}\right|_{t_f} = 0 \qquad \text{Condição de transversalidade}$$

e

$$V(x(t_f),t_f,p) = \phi(x(t_f),t_f)$$

Para resolver o problema, devem-se solucionar as seguintes equações diferenciais, correspondentes às condições necessárias do problema de otimização dinâmica.

- Equações algébrico-diferenciais dos estados:

$$F(\dot{x}(t),x(t),u(t),t,p) = 0 \qquad , \qquad \forall t \in T$$

- Condição estacionária:

$$0 = \frac{\partial \varphi(x,u,t,p)}{\partial u} + \left(\frac{\partial V}{\partial x}\right)^T \frac{\partial F(\dot{x},x,u,t,p)}{\partial u} \tag{3.40}$$

- Solução da equação diferencial parcial de *Hamilton-Jacobi-Bellman* – Cálculo do perfil de controle *u(t)*:

$$-\frac{\partial V}{\partial t} = \min_{u(t)} H\left(\dot{x}, x, u, \left(\frac{\partial V}{\partial x}\right)^T, t\right) \tag{3.41}$$

  Com isso, obtém-se o perfil ótimo u^*.

- Condições de Contorno:

$$-\left.\frac{\partial V}{\partial t}\right|_{t_f} = 0 \qquad \text{Condição de transversalidade}$$

  e

$$V(x(t_f),t_f,p) = \phi(x(t_f),t_f) \tag{3.42}$$

Este procedimento também resulta em um problema de valor de contorno duplo (*TPBVP).*

Princípio do Máximo de Pontryagin

O enunciado do principio do máximo de Pontryagin (*PMP*) deu origem a teoria de controle ótimo. Este princípio é decorrente da imposição de que o Hamiltoniano de um sistema contínuo sujeito a restrições de desigualdade nas variáveis de controle deve ser minimizado para qualquer conjunto possível destas variáveis de controle. O princípio do "máximo" é devido à diferença na convenção de sinal usada na definição do Hamiltoniano variacional. Há uma relação muito intima entre a Programação Dinâmica e o Princípio do Máximo de Pontryagin, que alguns consideram duas abordagens totalmente diferentes para a solução de controle ótimo.

Seja o problema geral de otimização dinâmica na forma de *Bolza* dado por:

$$\min_{x(t),u(t),t_f,p} J\left(x(t),u(t),t,p\right)=\phi\left(x\left(t_f\right),t_f\right)+\int_{t_0}^{t_f}\varphi\left(x(t),u(t),t,p\right)dt \qquad (3.43)$$

sujeito a:

$$F\left(\dot{x}(t),x(t),u(t),t,p\right)=0 \quad , \quad \forall t\in T$$

Definindo o Hamiltoniano:

$$H\left(\dot{x}(t),x(t),u(t),\lambda(t),t,p\right)=\varphi\left(x(t),u(t),t,p\right)+\lambda(t)^T F\left(\dot{x}(t),x(t),u(t),t,p\right) \qquad (3.44)$$

Para encontrar as trajetórias de *u(t)*, $\lambda(t)$ e *x(t)*, devem-se satisfazer as seguintes condições:

$$0=H_u(t)=\varphi_u(t)+\lambda(t)^T F_u\left(\dot{x},x,u,t,p\right) \quad \text{Condição de otimalidade} \qquad (3.45a)$$

$$-\dot{\lambda}(t)^T=H_x(t)=\varphi_x(t)+\lambda(t)^T F_x\left(\dot{x},x,u,t,p\right) \quad \text{Equação de Co-Estado} \qquad (3.45b)$$

$$\dot{x}=H_\lambda(t)=F\left(\dot{x},x,u,t,p\right) \quad \text{Equação de Estado} \qquad (3.45c)$$

onde $\lambda(t)\neq 0$ é o vetor de variáveis adjuntas de dimensão *nx* (multiplicadores de Lagrange do sistema de equações); $H_\lambda(t)$ é o gradiente de *H* em relação à $\lambda(t)$; $H_x(t)$, $\varphi_x(t)$ e $F_x(t)$ são os gradientes de *H,* φ e *F,* respectivamente, em relação à *x(t)*; $H_u(t)$, $\varphi_u(t)$ e $F_u(t)$ são os gradientes de *H,* φ e *F,* respectivamente, em relação a *u(t)*; e $H_u(t)=\partial H/\partial u=0$ é a condição necessária de 1ª ordem para otimalidade.

As condições de contorno são dadas por:

$$x(0)=x_0$$

$$\lambda\left(t_f\right)=\phi_x\left(t_f\right)^T$$

Este procedimento também resulta em um problema *TPBVP*. A relação entre as formulações de *PMP* e *HJB* é o fato que as variáveis adjuntas são as sensibilidades do custo ***V(x, t)*** em relação aos estados:

$$\lambda^T = \frac{\partial V}{\partial x}$$

Assim, o termo a ser minimizado é o Hamiltoniano e a equação diferencial parcial $\partial V/\partial x$ representa as dinâmicas das variáveis adjuntas (Equação 3.46).

$$\dot{\lambda}^T = \frac{d}{dt}\frac{\partial V}{\partial x} = \frac{d}{dx}\frac{\partial V}{\partial t} = -\frac{\partial H_{\min}}{\partial x} \tag{3.46}$$

onde $H_{\min}$ é o mínimo valor do Hamiltoniano.

Para resolver o *DAOP* pelos métodos indiretos é necessário primeiramente encontrar as expressões de $\dot{\lambda}$ e H_u, que não são fáceis de obter em alguns casos e também não são fáceis de resolver em muitos casos (Betts, 2001). Ao estabelecer as condições de otimalidade deste problema, obtém-se um sistema de equações de estado, que devem ser integrados em avanço, a partir das condições iniciais. E também um sistema de equações de co-estado (adjuntas), que é integrado de forma reversa a partir das condições de tempo final. Além de resolver o problema de controle singular (Hamiltoniano nulo). Na prática, é muito difícil de resolver este sistema de equações, principalmente quando são incluídas restrições de desigualdade. Este método deverá ter condições de identificar os arcos de restrições ativas e resolver cada parte do horizonte de otimização.

Estes métodos freqüentemente resultam em problemas numericamente mal condicionados e geralmente envolvem integrações iterativas dos sistemas de equações de estado e adjuntas, se tornando numericamente ineficientes (Renfro *et al.*, 1987). Além disso, a convergência do *TPBVP* pode ser lenta e computacionalmente proibitiva. Para minimizar o esforço computacional, devem ser fornecidas boas estimativas das condições iniciais das variáveis de estado e adjuntas para convergir eficientemente o método, o que não é uma tarefa fácil. Uma estimativa inadequada das variáveis adjuntas pode gerar um problema de estabilidade ao integrar as equações de co-estado, e também ser computacionalmente custoso na obtenção da solução ótima (Murthy *et al.*, 1980). Além disso, nem sempre as variáveis adjuntas têm significado físico, o que dificulta a definição de uma boa estimativa inicial de seus valores. O método indireto não funciona bem quando há descontinuidade nas variáveis adjuntas, o que ocorre freqüentemente na presença de restrições no estado. Uma abordagem analítica pode ser usada para contornar essa dificuldade, porém se torna uma tarefa árdua ao resolver problemas de grandes dimensões.

Os métodos indiretos baseados no *PMP* são mais frágeis que os diretos, pois necessitam de um estudo prévio do problema a ser resolvido. Por exemplo: as estruturas de chaveamentos dos arcos precisam ser conhecidas previamente, pois a solução freqüentemente passa pela divisão e seqüenciamento do horizonte de otimização em arcos de controle. Ou seja, é necessário identificar os arcos com restrições ativas, formular o *TPBVP* para cada arco, definir o intervalo de tempo de cada arco, definir as condições de junção dos arcos e

resolver o *TPVBP* (Betts, 2001). Devido a estas questões, os métodos indiretos são eficientes para problemas onde a formulação do *TPBVP* se torna simples de formular e de resolver. Tanto nos métodos indiretos quanto os diretos, o desconhecimento da estrutura dos arcos pode levar a uma solução oscilatória, exigindo algum tipo de recorrência na obtenção da solução ótima, tal como a detecção de estrutura e re-otimização.

Não é fácil introduzir restrições de estados nos métodos indiretos, porque é necessário aplicar o *PMP* com as restrições de estados nos *DAOP's* a serem resolvidos. E as presenças de restrições nos estados podem resultar em trajetórias ótimas de estruturas complicadas, em particular com muitos chaveamentos. Por outro lado, é mais fácil introduzir restrições de variáveis de estado nos métodos diretos, pois as mesmas são introduzidas diretamente no *NLP* a ser resolvido. Outra dificuldade se deve ao fato do método *PMP* exigir a diferenciação analítica das condições necessárias de otimalidade, tarefa que nem sempre é possível de ser feita.

Nos métodos indiretos (os baseados no *PMP*), normalmente é possível verificar a posteriori o status da trajetória ótima calculada. Já nos métodos diretos, isso é mais difícil de ser realizado. Uma forma possível de verificar as condições de otimalidade é através do exame dos multiplicadores de Lagrange do *NLP* resultante e analisar os arcos sugeridos pelo otimizador.

Nos últimos anos, a ***<u>análise das condições de otimalidade</u>*** (Srinivasan *et al.*, 2003) voltou a ganhar a atenção da comunidade cientifica com o objetivo de definir estratégias otimização com incertezas, baseado na análise e separação das partes das condições de otimalidade. Neste método, o instante de tempo onde as variáveis de controle se alternam de um arco para outro é chamado de instante de chaveamento. Para descrever esta metodologia, considere o problema de controle ótimo (Bryson e Ho, 1975) formulado na seguinte forma:

$$\begin{aligned} &\min_{u(t),t_f} \quad J = \phi\left(x\left(t_f\right)\right) \\ &s.a. \\ &\qquad \dot{x} = F\left(x,u,p,t\right); \quad x\left(t_0\right) = x_0 \\ &\qquad S\left(x,u,p,t\right) \le 0 \\ &\qquad T\left(x\left(t_f\right)\right) \le 0 \end{aligned} \tag{3.47}$$

O Hamiltoniano deste problema é dado por:

$$H\left(t\right) = \lambda\left(t\right)^T F\left(x,u,p,t\right) + \mu\left(t\right)^T S\left(x,u,p,t\right) \tag{3.48}$$

Usando o PMP, o problema de otimização original é então reformulado na forma:

$$\min_{u(t),t_f} \quad H(t) = \lambda(t)^T F(x,u,p,t) + \mu(t)^T S(x,u,p,t)$$

s.a.

$$\begin{aligned} &\dot{x} = F(x,u,p,t); \quad x(t_0) = x_0 \\ &\dot{\lambda}(t)^T = -\frac{\partial H}{\partial x}; \quad \lambda(t)^T = \left(\frac{\partial \phi}{\partial x}\right)\Big|_{t_f} + \nu^T \left(\frac{\partial T}{\partial x}\right)\Big|_{t_f} \\ &\mu(t)^T S(x,u,p,t) = 0 \\ &\nu^T T(x(t_f)) = 0 \end{aligned} \tag{3.49}$$

Os multiplicadores de Lagrange μ e ν são diferentes de zero quando as restrições estão ativas e zero caso contrário. As condições necessárias de otimalidade (*NCO - necessary conditions of optimality*) são satisfeitas quando $H_u(t) = \partial H/\partial u = 0$, e são dadas por:

$$H_u(t) = \frac{\partial H}{\partial u} = \lambda^T \frac{\partial F}{\partial u} + \mu^T \frac{\partial S}{\partial u} = 0$$

Para a variável de controle u_i, tem-se:

$$H_{u_i}(t) = \frac{\partial H}{\partial u_i} = \lambda^T \frac{\partial F}{\partial u_i} + \mu^T \frac{\partial S}{\partial u_i} = 0 \tag{3.50a}$$

ou

$$H_{u_i}(t) = \lambda^T F_{u_i} + \mu^T S_{u_i} = 0 \text{ , } i = 1,..., nu \tag{3.50b}$$

H_{ui} tem duas partes, a parte dependente do sistema $\lambda^T F_{ui}$ e a parte dependente de restrições $\mu^T S_{ui}$. O perfil ótimo da variável de controle u_i em um determinado intervalo de tempo (arco) é obtido em função da condição necessária de otimalidade. Dois cenários devem ser considerado, dependendo do valor de $\lambda^T F_{ui}$. Aqui, têm-se basicamente dois tipos de arcos, são eles: arcos não singulares ou bang-bang e arcos singulares. Arco não singulares são aqueles onde os perfis das variáveis de controle são determinados pelas restrições das próprias variáveis de controle ou das variáveis de estado ativas. Neste caso, $H_{ui} \neq 0;\ i = 1,..., nu$. Arcos singulares são aqueles onde os perfis das variáveis de controle não são determinados por quaisquer restrições ativas. São os arcos extremos, onde $H_{ui} = 0$. Esta condição não é suficiente para definir os perfis de controle ótimo. Neste caso, devem-se fazer testes adicionais para verificar se o arco singular é otimizante ou não.

A condição adicional é análoga à condição de convexidade ($H_{uu}(t) \geq 0$) é necessária para estabelecer os perfis de controle, sendo chamada de condição de Legendre-Clebsch generalizada. Para que se obtenha um perfil de controle otimizado, é necessário diferenciar sucessivamente a condição necessária de otimalidade (*NCO* - $H_u(t) = 0$) até que a variável de controle apareça explicitamente na condição de Legendre-Clebsch. Esta condição é representada por:

$$(-1)^k \frac{\partial}{\partial u}\left[\left(\frac{d}{dt}\right)^{2k} H_u(x,\lambda,u)\right] \geq 0 \qquad k = 0, 1, 2, ... \tag{3.51}$$

onde *k* é a ordem do problema singular. Desta forma, chamamos de problema singular de ordem *k* quando:

$$\left(\frac{d}{dt}\right)^i H_u = 0 \qquad i = 0, 1, .., k-1$$

e

$$\left(\frac{d}{dt}\right)^k H_u = a(x,\lambda) + b(x,\lambda)u = 0 \tag{3.52}$$

Na obtenção da solução de um problema de controle ótimo, deve-se identificar a estrutura da solução, ou seja, a seqüência de arcos singulares e não singulares, e os instantes de tempo das junções entre os arcos. Os instantes de tempo em que u(t) chaveia de arco singular para não singular e vice-versa são chamados de instantes de chaveamento. Nas soluções otimizadas, podem-se encontrar as seguintes situações:

Na situação onde as **restrições de caminho estão ativas**, têm-se as variáveis de estado e de controle ativos ao longo destes arcos, sendo que $\lambda^T F_{u_i} \neq 0$ e $\mu \neq 0$ no intervalo para satisfazer as *NCO's* ($H_{u_i}(t) = \lambda^T F_{u_i} + \mu^T S_{u_i} = 0$). Neste caso, obrigatoriamente uma das restrições de caminho deve estar ativa e o perfil de controle u_i pode ser calculado a partir desta restrição. Assim, podem-se ter as seguintes situações:

- *Restrições das variáveis de controle ativas*
  - São arcos não singulares nos perfis de controle – arcos determinados pelas restrições de controle ativas
    - No limite inferior (u_{min}), quando $H_u > 0$;
    - No limite inferior (u_{max}), quando $H_u < 0$.

- *Restrições das variáveis de controle inativas*
  - São arcos não singulares nos perfis de estado – arcos determinados pelas restrições de caminho dos estados ativas (u_{state}), onde $H_u \neq 0$.

Na situação onde a **solução está dentro da região viável**, tanto as variáveis de estado quanto de controle não estão ativos ao longo destes arcos, sendo que $\lambda^T F_{u_i} = 0$. Neste caso, os arcos de controle são computados de forma a satisfazer as *NCO's*. Desta forma, procuram-se perfis que buscam a otimalidade, ou seja, a ***sensibilidade nula do Hamiltoniano***. Estes são os arcos singulares (u_{sing}) determinados pelas transições, onde os perfis de controle não são determinados por quaisquer restrições ativas, onde $H_u = 0$.

Como a condição é $H_{u_i}(t) = 0$, deve-se diferenciá-la em relação ao tempo, resultando em:

$$\frac{dH_{u_i}}{dt} = \lambda^T \Delta F_{u_i} - \mu^T \frac{\partial S}{\partial x} F_{u_i}$$

onde o operador Δ é definido como:

$$\Delta \upsilon = \frac{\partial \upsilon}{\partial x} F - \frac{\partial F}{\partial x} \upsilon + \sum_{k=0}^{\infty} \frac{\partial \upsilon}{\partial u^{(k)}} u^{(k+1)}$$

Diferenciando sucessivamente H_{u_i} enquanto $\lambda^T \Delta^{k-1} F_{u_i} = 0$, tem-se:

$$\frac{d^k H_{u_i}}{dt^k} = \lambda^T \Delta^k F_{u_i} - \mu^T \frac{\partial S}{\partial x} \Delta^{k-1} F_{u_i} = 0 \qquad (3.53)$$

O processo de diferenciação é paralisado quando:

$$\lambda^T \Delta^k F_{u_i} \neq 0 \qquad (3.54a)$$

ou

$$\lambda^T \Delta^k F_{u_i} = a(x, \lambda, t) + b(x, \lambda, t) u_i = 0 \quad \text{(explícito em } u_i\text{)} \qquad (3.54b)$$

Definindo ξ_i o primeiro valor de k onde $\lambda^T \Delta^k F_{u_i} \neq 0$, então $\mu \neq 0$ e $\frac{d^{\xi_i} H_{u_i}}{dt^{\xi_i}} = 0$. Para poder resolver o problema, somente as restrições que tem grau relativo $r_{ij} = \xi_i$ podem estar ativas. Grau relativo r_{ij} da restrição ativa $S_j(x,\lambda)$ em relação a u_i é o número de diferenciações em S_j para u_i aparecer explicito em $\frac{d^{\xi_i} H_{u_i}}{dt^{\xi_i}} = 0$. Observe que somente alguns μ satisfazem $\frac{d^{\xi_i} H_{u_i}}{dt^{\xi_i}} = 0$ e $\mu \geq 0$ ao mesmo tempo. E só a restrição mais restritiva estará ativa e o μ indicará qual restrição permitirá que u_i seja determinado.

Só em algumas situações é possível determinar analiticamente o perfil de u_i. Na prática, estes princípios são utilizados em técnicas de otimização com incertezas (ex.: rastreamento de *NCO*) e detecção de estrutura em algumas técnicas de otimização dinâmica, que serão apresentados mais a diante.

3.1.2.2 Métodos diretos

São os métodos que transformam o *DAOP* em um problema de *NLP* através da parametrização de variáveis e são amplamente aplicados à solução de problemas de otimização dinâmica em engenharia química. Os métodos diretos podem ser classificados em seqüenciais e simultâneos.

Métodos Seqüenciais, também chamados de parametrização dos vetores de controle (*CVP – Control Vector Parameterization*) consistem na aproximação somente das trajetórias de controle por funções que envolvem poucos parâmetros, mantendo o sistema *DAE* na sua forma original. Desta forma, o *DAOP* é transformado em um *NLP* através da integração de *DAE's* e do gradiente da função objetivo, como um problema de valor inicial (*IVP – Initial Value Problem*) seguido da etapa de otimização, usualmente resolvido por um algoritmo de

SQP. Por isso, esta classe de métodos também é chamada de métodos com trajetórias realizáveis (*Feasible Path*). O *NLP* resultante da reformulação do problema de otimização é de pequena dimensão e se torna atrativo para aplicação em problemas de controle ótimo com sistemas *DAE*.

Uma limitação do método seqüencial está na dificuldade de tratar as restrições de caminho das variáveis de estado. Isso porque as variáveis de estado não estão diretamente incluídas no *NLP*.

Métodos Simultâneos, também chamados de discretização total, consistem na parametrização tanto das trajetórias das variáveis de controle como das variáveis de estado usando aproximações polinomiais onde os seus coeficientes se tornam as variáveis de decisão de um problema finito de *NLP*. Desta forma, o problema de otimização passa a ser a solução de um sistema de equações algébricas não-lineares de dimensão elevada que é resolvido simultaneamente com o problema de otimização. Por isso, esta classe de métodos também é chamada de métodos com trajetórias não realizáveis (*Infeasible Path*). Para resolver um problema com grande número de equações algébricas e de variáveis de decisão, necessita-se de algoritmos de *NLP* apropriados para lidar com estes tipos de problemas. Estes algoritmos de otimização para problemas de larga escala utilizam normalmente métodos de álgebra esparsa e usualmente resolvem o problema no espaço reduzido. Mesmo assim, não evitam a inicialização de um maior número de variáveis, pois todas as variáveis do sistema discretizado devem ter suas estimativas iniciais estabelecidas.

Devido às altas dimensões e complexidade dos modelos de processos químicos, nem sempre é fácil de otimizar usando a abordagem seqüencial. Os métodos simultâneos tendem a ser mais rápidos em termos de tempo computacional comparado à abordagem seqüencial, sendo mais eficientes para problemas de otimização de dimensões maiores.

Uma grande virtude do método simultâneo é a capacidade de lidar com as restrições nas variáveis de estado. Isso porque estas restrições são incluídas diretamente no *NLP* como restrições adicionais. O algoritmo de *NLP* força diretamente a obediência às restrições, prevenindo inviabilidade através da identificação e satisfação das restrições (os domínios e limites das variáveis são especificados, ex.: para evitar vazão negativa).

Na primeira abordagem, conhecida como métodos *shooting* diretos ou seqüenciais, onde se parametrizam as variáveis de controle *u(t)*, escolhe-se funções de controle *u(t)* finitas e utiliza-se diretamente na integração das equações algébrico-diferenciais do problema. Na segunda abordagem, as variáveis de controle *u(t)* e de estado *x(t)* são parametrizadas, gerando um sistema de equações algébricas que é resolvido de forma simultânea.

Para reformular o *DAOP* contínuo para um *NLP*, primeiramente os perfis de controle devem ser discretizados. Esta discretização é efetuada em três etapas:

- Dividir o horizonte de tempo em um número de subintervalos (ou elementos). Normalmente se divide o horizonte de tempo em intervalos iguais, onde

$$t_0 < t_1 < \cdots < t_{NE} = t_f$$

sendo *NE* o número de subintervalos de tempo.

- Normalizar o horizonte de tempo de cada subintervalo. Cada subintervalo pode ser adimensionalizado para que o tempo fique no intervalo [0, 1]. Não necessariamente se normaliza o intervalo de tempo, mas esta medida facilita a integração do *IVP*. A transformação empregada é a seguinte:

$$\theta_i(\tau) = t_i + \tau h_i\,,\quad t_i = t_0 + \sum_{k=0}^{i-1} h_k \;\text{ e }\; h_i = t_{i+1} - t_i\,,\quad \tau \in [0,1] \text{ e } \theta_i(\tau) \in [t_i, t_{i+1}] \qquad (3.55)$$

  sendo τ o tempo adimensional em cada intervalo, h_i o tamanho do subintervalo, t_i o tempo no inicio do subintervalo i e $\theta_i(\tau)$ é o tempo dimensional no subintervalo i referente ao tempo adimensional τ.

- Escolher a função de parametrização, $\psi_i(t)$, das variáveis de controle. As funções $\psi_i(t)$ são as funções básicas de parametrização, que podem ser tipicamente polinômios contínuos por partes. Normalmente se aplicam funções constantes por partes, lineares por partes, cúbicas por partes ou polinômios de Lagrange por partes.

Funções Constantes por Partes (*Piecewise Constant*) (vide Figura 3.6)

$$u_i(t) = \sum_{k=1}^{NE} \hat{u}_{i,k}\psi_k(t) \quad \text{onde} \quad \psi_k(t) = \begin{cases} 1 & \forall t \in [t_k, t_{k+1}] \\ 0 & \forall t \notin [t_k, t_{k+1}] \end{cases} \qquad (3.56)$$

onde $\hat{u}_{i,k}$ é o parâmetro da variável de controle i no subintervalo k.

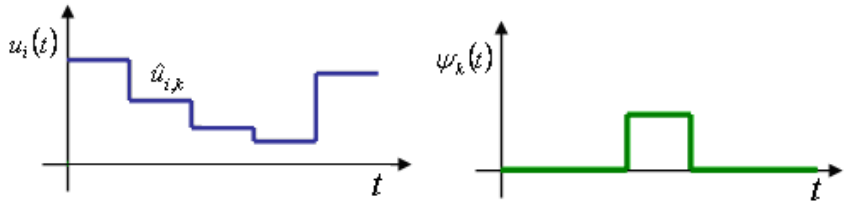


Figura 3.6 – Funções constantes por partes.

Funções Lineares por Partes (*Piecewise Linear*) (vide Figura 3.7)

$$u_i(t) = \sum_{k=1}^{NE} \hat{u}_{i,k-1}\left(1 - \psi_k(t)\right) + \psi_k(t)\,\hat{u}_{i,k} \qquad (3.57)$$

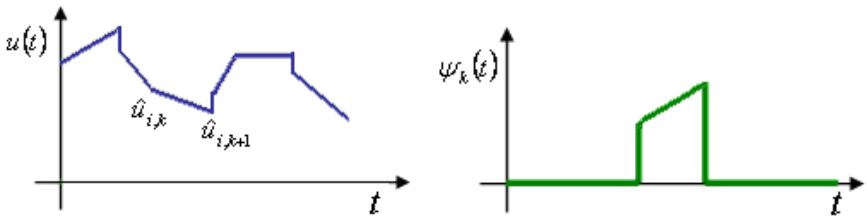


Figura 3.7 – Funções lineares por partes – descontínuas.

Caso o perfil do controle deva ser contínuo (vide Figura 3.8), devem-se adicionar as condições de continuidade dos controles no problema ou definir a função básica na forma:

$$\psi_k(t) = \frac{t - t_k}{t_{k+1} - t_k} \text{ para } t_k \le t \le t_{k+1} \text{ e } \psi_k(t) = 0 \text{ caso contrário.} \tag{3.58}$$

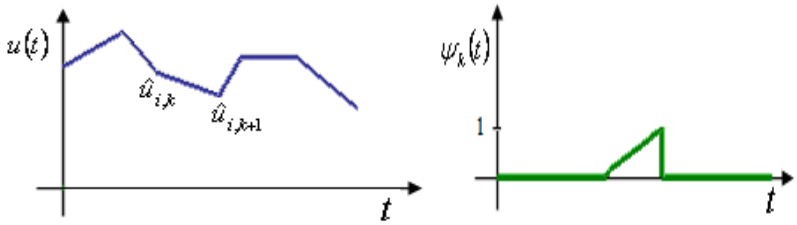


Figura 3.8 – Funções lineares por partes – contínuas.

Na aproximação polinomial (vide Figura 3.9) a variável de controle u_j no intervalo j usando as funções de polinômios de Lagrange por partes (*piecewise polynomial*), tem-se:

$$u_i(t) = \sum_{k=1}^{NE} \hat{u}_{i,k} \psi_k(t), \quad \psi_k(t) = \prod_{\substack{p=0 \\ p \neq k}}^{NE} \frac{(t - t_p)}{(t_k - t_p)}, \; t_0 \le t \le t_{NE} \tag{3.59}$$

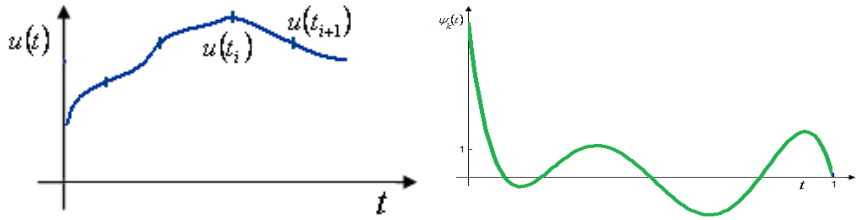


Figura 3.9 – Funções polinomiais por partes.

Neste caso, pode-se utilizar a aproximação polinomial que for mais conveniente. Desta forma, é usual aplicar colocação, *spline*, dentre outros.

A discretização do estado pode ser feita usando diversas estratégias, podendo ser *multiple shooting*, colocação global, colocação em elementos finitos ou diferenças finitas. Em todas elas, devem ser adicionadas restrições de continuidade nos estados. Esta continuidade é acrescentada na forma:

$$x(t_{j+1}) - s^x_{j+1} = 0 \qquad j = 1, \cdots, NE \tag{3.60}$$

onde s^x_j é o valor estimado da variável de estado no estágio j, *usado na etapa de correção (a cada iteração) do algoritmo de otimização* e NE é o número de estágio. Vide Figura 3.10.

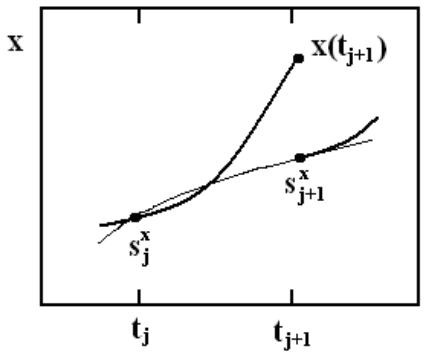


Figura 3.10 – Condição de Continuidade no estado

Discretização Parcial (Métodos Seqüenciais)

A abordagem seqüencial é um método do caminho viável (*feasible path*), onde a cada iteração o sistema *DAE* é resolvido por integração. Este método tem problemas de falta de estabilidade e de robustez (Gill *et al.*, 2000, Ascher e Spiteri, 1995), sendo robusto quando o sistema contém somente modos estáveis, caso contrário, a dificuldade de se encontrar uma solução viável é aumentada apreciavelmente. Assim, a solução de um sistema *DAE* para um dado conjunto de parâmetros de controle pode ser difícil ou até mesmo impossível. A simulação do processo pode falhar se a região viável não existir para certas variáveis de otimização (não fisicamente possíveis). Por isso, os métodos seqüenciais não são recomendáveis para sistemas instáveis ou de sensibilidade paramétrica muito alta, onde as variáveis de estado podem explodir nas iterações intermediárias do otimizador devido a variáveis de controle ou parâmetros inadequados. Este problema é acentuado pelo fato de não ser muito fácil definir em que nível os perfis de controle devem ser discretizados. O custo computacional principal da otimização está na solução das equações de estado e suas sensibilidades usando abordagem direta ou adjunta. Devido a estes custos, raramente as hessianas são calculadas e por isso, os métodos quasi-Newton *SQP* são os mais adequados para resolver estes tipos de problemas (Biegler e Wächter, 2003).

A ***programação dinâmica*** é baseada no princípio de otimalidade de Bellman (1957), que versa o seguinte:

> *"Se uma decisão forma uma solução ótima em um estágio de um processo, então qualquer decisão remanescente deve ser ótima com respeito ao resultado desta decisão tomada".*

Isto significa que, ao se dividir o horizonte de tempo em múltiplos estágios e decompor um problema de otimização dinâmica interconectado em uma seqüência de subproblemas, estes podem ser resolvidos serialmente. Cada subproblema contém um subconjunto de variáveis de decisão, que pode ser sequencialmente resolvido, de trás para frente, a partir do estágio final, por algum algoritmo de otimização apropriado. Infelizmente, não é possível obter a solução ótima analiticamente, sendo necessário utilizar algum método numérico. O esquema do algoritmo de programação dinâmica está descrito na Figura 3.11.

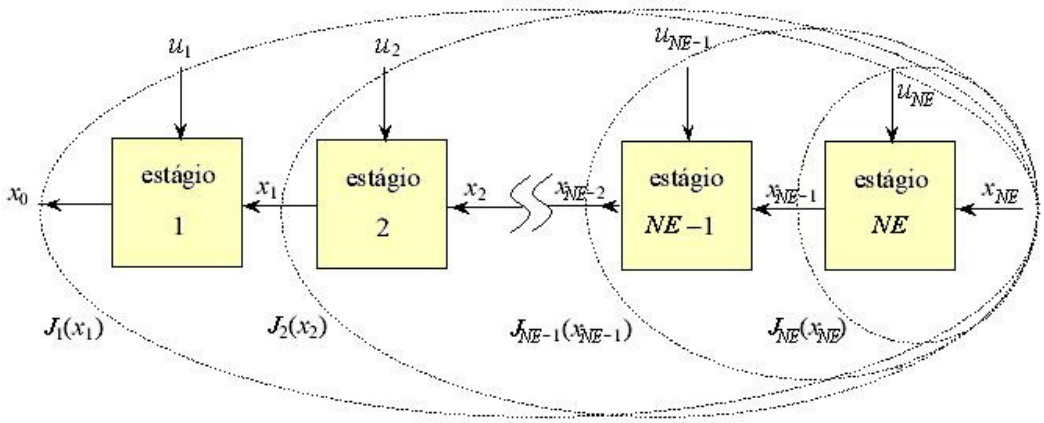


Figura 3.11 – Esquema da Programação Dinâmica de Bellman.

O ***método single shooting (Disparo Simples)*** (Pollard e Sargent, 1970; Sargent e Sullivan, 1977) consiste da aproximação da trajetória de controle por uma função de poucos parâmetros, deixando as equações de estado na forma original do sistema de *DAE*. Esta aproximação é feita dividindo o horizonte de tempo do problema de otimização em *NE* subintervalos onde as variáveis de controle são representadas por funções que podem ser constantes por partes, lineares por partes ou por uma aproximação polinomial. Com a escolha da forma dos perfis das variáveis de controle, a função original se transforma em $\tilde{F}(\dot{x}, x, \hat{u}) = 0$, onde $\hat{u}$ são os novos parâmetros de otimização invariantes no tempo. Neste caso, o problema passa a ter apenas os parâmetros de otimização ($\hat{u}$) e a função objetivo. Este método é do tipo caminho viável (*feasible path*), isto é, a solução é melhorada a cada iteração. Neste método, os valores da função objetivo e de sua matriz Jacobina são calculados para que a rotina de otimização seja usada iterativamente para encontrar os perfis ótimos das variáveis de controle. O procedimento se torna robusto quando é fornecida inicialmente uma solução viável.

O método *single shooting* necessita que as equações de estado e de sensibilidade sejam integradas simultaneamente para dados perfis de controle. Com isso, fornece os valores das funções e gradientes para o algoritmo de *NLP*. O algoritmo de otimização é aplicado no laço externo para atualizar as ações de controle.

O algoritmo de *NLP* requer avaliações da função objetivo e dos gradientes. Há três formas de obter o gradiente, são elas: diferenças finitas, equações de sensibilidade e equações adjuntas. As equações de sensibilidade são obtidas, de forma eficiente, por diferenciação das equações de processo, após a discretização das variáveis de controle, em relação ao conjunto de parâmetros. Esta é considerada uma característica interessante da abordagem seqüencial, ou seja, a matriz Jacobiana das equações de sensibilidade é reduzida a uma multiplicação de matriz por parâmetros em cada etapa de integração. Dada a sensibilidade dos estados em relação aos parâmetros, as derivadas da função objetivo e das restrições em relação aos parâmetros são facilmente calculadas.

A forma geral do *método single shooting* é a seguinte (vide a Figura 3.12):

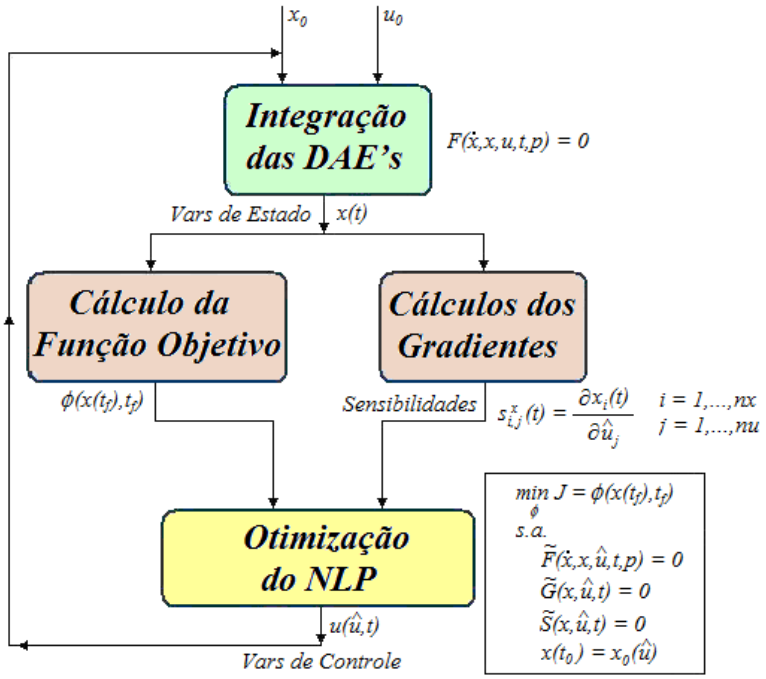


Figura 3.12 – Método do *Single-Shooting*.

Seja o problema de otimização dinâmica dado por:

$$\min_{x(t),u(t),t_f,p} \left\{ J = \phi\left(x\left(t_f\right),t_f\right) \right\} \tag{3.61}$$

sujeito a:

$$F\left(\dot{x}(t),x(t),u(t),t,p\right) = 0$$
$$S\left(x(t),u(t),t,p\right) \le 0, \quad \forall t \in T$$
$$I\left(x_0,u_0,t_0,p\right) = 0$$
$$x_{min} \le x(t) \le x_{max}$$
$$u_{min} \le u(t) \le u_{max}$$

Passo 1 – Definir uma função de parametrização das variáveis de controle (entradas do processo).

$$u_i(t) = \sum_{k=1}^{NE} \hat{u}_{i,k} \psi_k(t) \qquad i = 1,\cdots,nu$$

Passo 2 – Reformular o problema de otimização incluindo os parâmetros de controle, na forma:

$$\min_{x(t),\hat{u},t_f,p} \left\{ J = \phi\left(x\left(t_f\right),t_f\right) \right\} \tag{3.62}$$

sujeito a:

$$F\left(\dot{x}(t),x(t),\hat{u},t,p\right) = 0$$
$$S\left(x(t),\hat{u},t,p\right) \le 0$$

$$u_i(t) = \sum_{k=1}^{NE} \hat{u}_{i,k} \psi_k(t) \qquad i = 1, \cdots, nu$$
$$I(x_0, u_0, t_0, p) = 0$$
$$x_{\min} \le x(t) \le x_{\max}$$
$$u_{\min} \le u(t) \le u_{\max}$$

Passo 3 – Escolher valores iniciais dos estados x_0 e todos os parâmetros $\hat{u}$, e substituir estes valores das variáveis no modelo dinâmico do processo.

Passo 4 – Integrar o modelo dinâmico dado por $F(\dot{x}(t), x(t), \hat{u}, t, p) = 0$ para todas as entradas de t_0 a t_f, usando algum solver de *DAE* (ex.: *DASSL* ou *DASPK*).

Passo 5 – Avaliar a função objetivo *J* e as restrições $S(x(t), \hat{u}, t, p) \le 0$ usando os perfis de estado obtido no passo 4 e os parâmetros de controle $\hat{u}$ proposto.

Passo 6 – Atualizar os valores das variáveis de decisão $\hat{u}$ como algum método de *NLP*.

Passo 7 – Verificar a convergência de $\hat{u}$. Caso seja satisfeita, a otimização está concluída, caso contrário repetir os passos 4 a 7 até convergir.

Para usar um algoritmo de *NLP*, é necessário calcular as derivadas das restrições de desigualdade $g(x_k, \hat{u}, t_k) \ge 0$. Estes cálculos envolvem as derivadas parciais $\partial x_i / \partial \hat{u}_j$ que são soluções das equações diferenciais de sensibilidade. Este sistema deve ser resolvido junto com o conjunto original de *DAE's*.

Um atrativo do método *Single-Shooting* é que a necessidade de memória da máquina é consideravelmente pequena e tem matrizes Jacobiano de pequenas dimensões para serem invertidas.

O ***método do multiple shooting*** (Bock e Plitt, 1984) serve como uma ponte entre as abordagens seqüencial e simultânea que são baseadas na discretização total das variáveis de controle e de estado, podendo ser chamado de abordagem hibrida. O domínio do tempo é dividido em subintervalos de tempo menores e o sistema *DAE* é integrado em cada subintervalo $[t_i, t_{i+1}]$, introduzindo-se valores iniciais s_i^x (dos estados) nos tempos t_i como variáveis de otimização adicionais. As variáveis de controle são tratadas da mesma forma que os métodos seqüenciais, desta forma o *DAOP* é transformado em vários problemas de valor inicial local que podem ser resolvidos paralelamente. Este fato faz com que estes métodos sejam mais robustos que os *Single-Shooting*, pois são menos sensíveis a não linearidades e instabilidades nos sistemas *DAE*. Além disso, é possível paralelizar o processo de integração para cada elemento discreto no método *Multi-Shooting*.

Para obter os gradientes, as sensibilidades são obtidas tanto para variáveis de controle como das condições iniciais dos estados de cada elemento. Restrições de igualdade são acrescentadas ao problema de otimização para garantir a continuidade dos estados entre

cada elemento na solução final, satisfazendo o sistema *DAE*. Nesta abordagem, as restrições de desigualdade nos estados e nos controles podem ser impostas diretamente nos pontos da malha, mas as restrições de trajetória podem não ser satisfeitas entre os pontos da malha. Este problema pode ser evitado aplicando técnicas de penalidade para forçar a viabilidade.

O *NLP* resultante é resolvido usando métodos do tipo *SQP*. A cada iteração *SQP*, o sistema *DAE* é integrado em cada estágio, os gradientes da função objetivo e das restrições podem ser obtidos usando as equações de sensibilidade como no método *single shooting* (vide Figura 3.13). Como as integrações são desacopladas nos diferentes subintervalos $[t_i, t_{i+1}]$, o método é adequado para utilizar computação paralela na fase de integração.

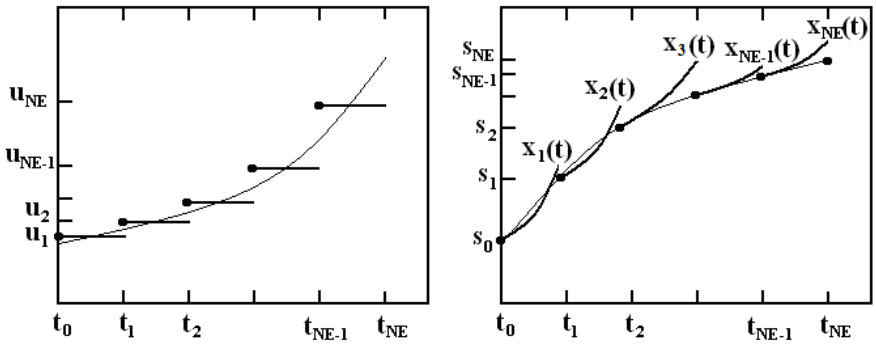


Figura 3.13 – Método *Multiple Shooting* (Bock e Plitt, 1984).

No *multiple shooting*, os valores dos estados não são propostos somente no instante inicial, mas também no início de cada subintervalo. Os desvios entre estimativas a os valores obtidos nestes pontos de tempo são então reduzidos pela nova estimativa com a ajuda de um método do tipo Newton, por exemplo. Intervalos menores no *multiple shooting* resultam em menor não linearidade e menor sensibilidade em relação aos valores propostos do que no *single shooting*.

Após a definição da função de aproximação das variáveis de controle, o sistema *DAE* é particionado em *NE* subintervalos $[t_i, t_{i+1}]$, com $i = 0, 1, ..., NE - 1$. Em cada ponto da grade os valores iniciais das variáveis de estado s_i são escolhidos como variáveis adicionais desconhecidas. Desta forma, em cada intervalo $[t_i, t_{i+1}]$, calculam-se as trajetórias de $x_i(t)$ através da solução de *IVP's* desacoplados. Note que as trajetórias $x_{i+1}\left(t; s_i^x, \hat{u}_i\right)$ dependem somente de $s_i^x, \hat{u}_i$ e t, sendo, portanto independentes em cada estágio e, como conseqüência, a função objetivo pode ser calculada para cada estágio independentemente.

$$J\left(s_i^x, \hat{u}_i\right) = \int_{t_i}^{t_{i+1}} \varphi\left(x_{i+1}\left(s_i^x, \hat{u}_i, t\right), \hat{u}_i\right) dt \qquad i = 0, 1, ..., NE - 1$$

A forma geral do *método multiple shooting* é a seguinte. Seja o problema de otimização dinâmica dado por:

$$\min_{x(t),u(t),t_f,p} \left\{ J = \phi\left(x\left(t_f\right), t_f\right) \right\} \tag{3.63}$$

sujeito a:

$$F\left(\dot{x}(t), x(t), u(t), t, p\right) = 0$$
$$S\left(x(t), u(t), t, p\right) \le 0, \quad \forall t \in T$$
$$I\left(x_0, u_0, t_0, p\right) = 0$$
$$x_{min} \le x(t) \le x_{max}$$
$$u_{min} \le u(t) \le u_{max}$$

Passo 1 – Definir uma função de parametrização das variáveis de controle (entradas do processo).

Passo 2 – Reformular o problema de otimização incluindo os parâmetros de controle:

$$\min_{s^x, \hat{u}, p} \left\{ J = \phi\left(s^x_{NE}, t_{NE}\right) \right\} \tag{3.64}$$

sujeito a:

$$F\left(s^x_i, \hat{u}_i, p\right) = 0$$
$$S\left(s^x_i, \hat{u}_i, p\right) \le 0 \; G\left(s^x_i, \hat{u}_i, p\right) \le 0, \qquad \forall t \in \left[t_i, t_{i+1}\right] \quad i = 0, 1, ..., NE - 1$$
$$x_{i+1}\left(t_{i+1}; s^x_i, \hat{u}_i\right) - s^x_{i+1} = 0$$
$$x_0 = s^x_0$$
$$x_{min} \le s^x_i \le x_{max}$$
$$u_{min} \le \hat{u}_i \le u_{max}$$

Passo 3 – Escolher valores iniciais dos estados x_0, todos os parâmetros $\hat{u}$, os valores iniciais intermediários s^x_i nos tempos t_i e substituir estes valores das variáveis no modelo dinâmico do processo.

Passo 4 – Integrar o modelo dinâmico em cada subintervalo $[t_i, t_{i+1}]$ (paralelamente) dado por $F\left(\dot{x}(t), x(t), \hat{u}, t, p\right)$ para todas as entradas de t_0 a t_f, usando algum solver de *DAE* (ex.: *DASSL* ou *DASPK*).

Passo 5 – Avaliar a função objetivo J e as restrições $S\left(x(t), \hat{u}, t, p\right) \le 0$ e $x_{i+1}\left(t_{i+1}; s_i, \hat{u}_i\right) - s^x_{i+1} = 0$ usando os perfis de estado obtido no passo 4 e os parâmetros de controle $\hat{u}$ proposto. Analogamente a parte Lagrange da função objetivo, $\varphi\left(s^x_i, \hat{u}_i, t, p\right)$, pode ser avaliada a cada intervalo independentemente.

Passo 6 – Atualizar os valores das variáveis de decisão $\hat{u}$ como algum método de *NLP*.

Passo 7 – Verificar a convergência de $\hat{u}$. Caso seja satisfeita, a otimização está concluída, caso contrário repetir os passos 4 a 7 até convergir.

A eficiência da abordagem tem sido observada em muitas aplicações práticas. Uma das razões mais importantes é a possibilidade de incorporar informação sobre o

comportamento da trajetória do estado na estimativa inicial para o procedimento de solução iterativa; isto pode reduzir a influência de estimativas iniciais ruins para os controles (que são normalmente muito menos conhecidos).

O método *Single-Shooting* é usado em problemas com muitas variáveis de estado, com poucas variáveis de controle e poucos intervalos de tempo. Este método usa discretização das variáveis de controle que são definidas como funções simples em certo número de intervalos de controle. Enquanto o método *Multiple-Shooting* é usado em problemas com poucas variáveis de estado, muitas variáveis de controle e muitos intervalos de tempo.

Note que se for usado um integrador Runge-Kutta implícito (*IRK*) e os subintervalos forem elementos finitos de tamanhos adequadamente escolhidos, então a abordagem *Multiple-Shooting* fica idêntica a formulação de colocação simultânea. Por esta razão o termo *Multiple-Shooting* pode ser usado como uma ponte entre ambos os métodos.

Discretização Total (Métodos Simultâneos)

Nos métodos de discretização total (Logsdon e Biegler, 1989; Cervantes e Biegler, 1998), se realiza a parametrização tanto da variável de controle quanto da variável de estado, evitando a integração das equações dinâmicas. Esta discretização é feita usando polinômios nos subintervalos (ou elementos finitos), onde os coeficientes e os tamanhos dos elementos se tornam variáveis de decisão em um problema de *NLP* muito maior. Uma vez discretizado o *DAOP*, resolve-se o problema através de um algoritmo de *NLP* (ex.: *SQP*, *rSQP* ou *IP*. Os problemas gerados são de dimensões elevadas e requerem técnicas especiais de solução.

O método de discretização total tem melhor estabilidade do que a discretização parcial, especialmente quando tem modos exponenciais crescentes (Biegler et al., 2002, Kameswaran e Biegler, 2006) , o que se torna uma grande virtude deste método. Eles convergem mais rapidamente, em problemas de larga escala, pois evita simulações (solução do modelo) repetitivas durante as iterações, tendo menos riscos de falhas de convergência na simulação do sistema de equações. O otimizador encontra soluções intermediárias que podem não ser fisicamente realizáveis. É um método onde é mais difícil identificar os efeitos das variáveis de otimização na função objetivo durante as iterações do otimizador e, portanto se tornando mais difícil de fazer um diagnóstico. Este método é do tipo caminho inviável (*infeasible path*), isto é, a solução fica disponível somente quando o processo iterativo convergir. Diferentemente dos métodos seqüenciais, o método simultâneo não requer solução do *IVP* a cada iteração do *NLP*. Pode ser mostrado que se a colocação é usada para discretização no método simultâneo, o problema é equivalente a realizar a integração Runge-Kutta totalmente implícita, que não é tão eficiente quanto o método *BDF* (*backward differentiation formulas*) para *DAE's*. Apesar dos métodos seqüenciais garantirem uma solução ótima seguindo um caminho viável, eles podem ser proibitivamente custoso porque tendem a convergir mais lentamente e requer a solução das equações diferenciais a cada iteração. A abordagem simultânea evita este cálculo através da convergência simultânea para o ótimo enquanto resolve as equações diferenciais.

Hertzberg e Asbjornsen (1977) foram os primeiros a introduzir a idéia de usar a colocação ortogonal junto com método quasi-Newton para executar a estimativa simultânea dos

parâmetros e integração não-linear do sistema dinâmico. As equações de sensibilidade das variáveis dependentes em relação aos parâmetros junto com o sistema de equações diferenciais são trocadas por um conjunto de equações algébricas aproximadas e a otimização é realizada no subespaço dos parâmetros.

O ***método de colocação ortogonal*** consiste na aproximação de *DAOP's* para equações algébricas usando polinômios ortogonais. Nesta abordagem, divide-se o horizonte de tempo em *NE* intervalos de tempo, onde os nós estão posicionados nas raízes do polinômio de *Legendre* (vide figura 3.14). A colocação ortogonal pode ser pensada como um procedimento de Galerkin com as integrais resultantes aproximadas por uma quadratura *NE* pontos. Note que os pontos desta quadratura correspondem às raízes do polinômio ortogonal. Em seguida, parametrizam-se as variáveis de controle e de estado no problema de otimização dinâmica.

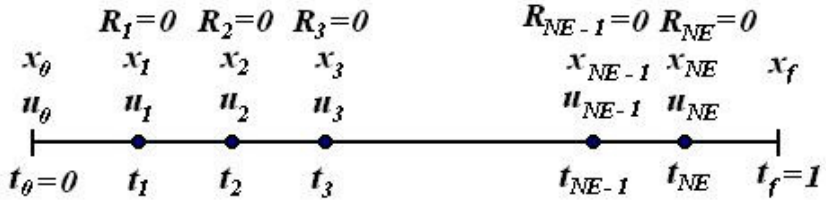


Figura 3.14 – Pontos de colocação global.

Seja o problema de otimização dinâmica definido por:

$$\min_{x,u} \ \phi\left(x\left(t_f\right), t_f\right) \tag{3.65}$$

s.a.:

$$F\left(\dot{x}(\tau), x(\tau), u(\tau), \tau\right) = 0 \qquad \tau \in [0,1]$$
$$S\left(x(\tau), u(\tau), \tau\right) \le 0$$
$$G\left(x(\tau), u(\tau), \tau\right) = 0$$
$$x(0) = x_0$$
$$x^L \le x(\tau) \le x^U$$
$$u^L \le u(\tau) \le u^U$$

Com a discretização total, o *DAOP* é transformado em um *NLP*, resultando em *NX*×*NE* equações residuais e *NU*×*NE* parâmetros desconhecidos.

O método de colocação global aproxima os estados usando polinômios de ordem *NE*+1. Para sistemas mal comportados (i.e., dinâmicas muito rápidas), a precisão da aproximação por colocação global pode exigir um *NE* muito grande.

Uma alternativa a colocação global é o uso do ***método de colocação em elementos finitos***, que consiste de aproximações polinomiais por partes (Cuthrell e Biegler, 1987). Um conjunto de polinômios de estado de ordem *NCOL*+1, $x^i_{(NCOL+1)}(\tau)$, e polinômios de controle de ordem *NCOL*, $u^i_{(NCOL)}(\tau)$, são definidos nos elementos finitos. Cada elemento finito h_i é limitado por dois nós, t_i e t_{i+1}, com $h_i = t_{i+1} - t_i$ (vide Figura 3.15). A distribuição dos

elementos pode ser escolhida tal que as aproximações sejam feitas de forma eficiente e precisa. O domínio $\tau \in [0,1]$ é mapeado a cada elemento finito através da fórmula (com $t_0 = 0$, $t_{NE+1} = t_f$) :

$$t = t_i + \tau h_i \qquad i = 1, \cdots, NE \qquad \text{para} \qquad \tau \in [t_i, t_{i+1}]$$

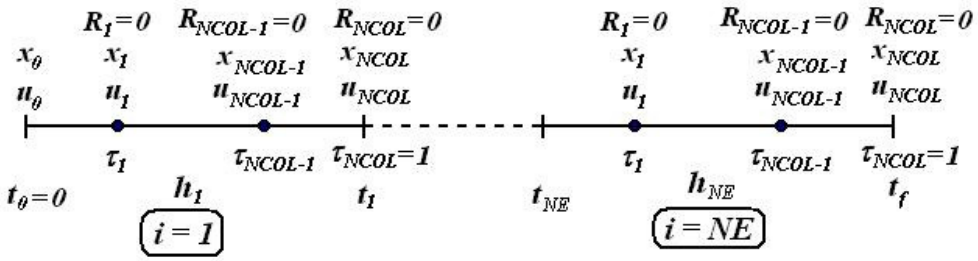


Figura 3.15 – Colocação ortogonal em elementos finitos.

As localizações das raízes do polinômio ortogonal de Legendre (com $\tau_0 = 0$) são mapeadas para os pontos:

$$t_{(i-1)(NCOL+1)+j} = t_i + \tau_j h_i \qquad i = 1, \cdots, NE\ ,\ \ j = 0, \cdots, NCOL \tag{3.66}$$

É conveniente definir a expressão *(i–1)(NCOL+1)+j* como a nova variável *[ij]*. Com esta convenção, pode-se reescreve *t* como:

$$t_{[ij]} = t_i + \tau_j h_i \quad i = 1, \cdots, NE\ ,\ \ j = 0, \cdots, NCOL \tag{3.67}$$

A forma geral do método é a seguinte. Seja o problema de otimização dinâmica definido por:

$$\min_{x,u} \phi\left(z\left(t_f\right), t_f\right) \tag{3.68}$$

s.a.:

$$\begin{aligned}
&F(\dot{x}(\tau), x(\tau), y(\tau), u(\tau), \tau) = 0 \qquad \tau \in [0,1] \\
&S(x(\tau), y(\tau), u(\tau), \tau) \le 0 \\
&G(x(\tau), y(\tau), u(\tau), \tau) = 0 \\
&x(0) = x_0 \\
&x^L \le x(\tau) \le x^U \\
&y^L \le y(\tau) \le y^U \\
&u^L \le u(\tau) \le u^U
\end{aligned}$$

onde $z = [x^T\ y^T]^T$, sendo *x* as variáveis diferenciais e *y* as variáveis algébricas.

Passo 1 – Discretizar as variáveis de estado e controle com o método de colocação em elementos finitos. Para calcular os valores de *x*, *y*, *u* em qualquer instante de tempo *t*, pode-se utilizar os polinômios interpoladores abaixo:

$$x(t) = x_{i-1} + h_i \sum_{q=0}^{ncol} \Omega_q \left(\frac{t - t_{i-1}}{h_i} \right) \frac{dx}{dt_{i,q}} \tag{3.69a}$$

$$y(t) = \sum_{q=1}^{ncol} \Psi_q \left(\frac{t - t_{i-1}}{h_i} \right) y_{i,q} \tag{3.69b}$$

$$u(t) = \sum_{q=1}^{ncol} \Psi_q \left(\frac{t - t_{i-1}}{h_i} \right) u_{i,q} \tag{3.69c}$$

onde:

$$\Omega_q(\tau) = \sum_{j=1}^{ncol} \frac{c_{ncol+1-j,q} \tau^j}{j!} \tag{3.69d}$$

$$\Psi_q(\tau) = \sum_{j=1}^{ncol} \frac{c_{ncol+1-j,q} \tau^{(j-1)}}{(j-1)!} \tag{3.69e}$$

Para $i = 1,...,NE$, sendo NE o número de elementos finitos, ψ_q a derivada de Ω_q, e $c_{ncol+1-j}$ são coeficientes dos polinômios interpoladores. Para as variáveis de controle e algébricas são usados polinômios interpoladores com uma ordem abaixo dos polinômios para as variáveis de estado para permitir as descontinuidades nos extremos dos elementos.

Passo 2 – Substituir as variáveis de estado e controle discretas no modelo dinâmico do processo e obter a expressão algébrica para os resíduos.

$$R(\tau_{[il]}, h_i) = f(X_{[il]}, Y_{[il]}, U_{[il]}); \quad X_0 = x_0 \quad i = 1, \cdots, NE \,, l = 1, \cdots, NCOL$$

As expressões de $\Omega_j(\tau_i)$ e $\psi_j(\tau_i)$ podem ser calculadas *offline*, pois dependem somente das localizações das raízes do polinômio de Legendre. Considerando, para resolver uma iteração do problema, que as variáveis $U_{[il]}$ são fixas, o sistema de equações do resíduo é composto de $NU[NE.NCOL+1]$ equações e $NU[NE.NCOL+1]$ coeficientes de estado. Para fazer o sistema bem posto, é necessário escrever um conjunto adicional de $NU(NE-1)$ equações para satisfazer as condições de continuidade dos estados nos nós interiores t_i, $i = 2, \cdots, NE$. Isto é feito forçando:

$$x^i_{(NCOL+1)}(t_i) = x^{i-1}_{(NCOL+1)}(t_i) \qquad i = 2, \cdots, NE$$

ou

$$X_{[i0]} - X_{[i-1,0]} - h_i \sum_{q=1}^{ncol} \Omega_q(1) \frac{dx}{dt_{i,q}} = 0 \tag{3.70}$$

Estas equações extrapolam o polinômio $x^{i-1}_{(NCOL+1)}(t)$ para o ponto final de seus elementos e fornece uma condição inicial para o próximo elemento e polinômio $x^1_{(NCOL+1)}(t)$.

Passo 3 – Substituir o modelo dinâmico discretizado no problema de otimização.

Incluindo o modelo *DAE* (discretizado em elementos finitos) e as condições de continuidade, a formulação do *NLP* fica:

$$\min_{X,U} \ \phi\left(X_{NE+1}\right) \tag{3.71}$$

s.a.:

$$R\left(\tau_{[il]}\right) = F\left(X_{[il]}, Y_{[il]}, U_{[il]}\right) = 0 \quad i = 1,\cdots,NE\ , l = 1,\cdots,NCOL$$

$$X_{[i0]} - X_{[i-1,0]} - h_i \sum_{q=0}^{ncol} \Omega_q(1) \frac{dx}{dt_{i,q}} = 0 \qquad i = 2,\cdots,NE$$

$$S\left(X_{[il]}, Y_{[il]}, U_{[il]}\right) \le 0$$

$$G\left(X_{[il]}, Y_{[il]}, U_{[il]}\right) = 0 \quad i = 1,\cdots,NE\ , l = 1,\cdots,NCOL$$

$$X_0 = x_0$$

$$x^L \le X_{[il]} \le x^U$$

$$y^L \le Y_{[il]} \le y^U$$

$$u^L \le U_{[il]} \le u^U$$

Com t_i, $i = 2,\cdots,NE$ fixos, o *NLP* pode ser usado para resolver a maioria dos tipos de *DAOP's*.

Passo 4 – Encontrar τ_{il} usando o método de colocação ortogonal em elementos finitos.

Passo 5 – Resolver o problema do passo 3 em todos os tempos τ_{il} escolhidos no passo 4 usando um algoritmo de programação não-linear como o *SQP*.

Este tipo de método pode encontrar uma solução ótima aproximada mesmo que se inicie de uma estimativa inicial ruim. A principal desvantagem do método é que o problema de otimização resultante tem um grande número de variáveis desconhecidas e a solução pode ser dificultada pela existência de mínimos locais. O grande tamanho do problema exige a utilização de algoritmos eficientes de otimização de larga escala. Com o desenvolvimento do *SQP*, *rSQP* e algoritmos de ponto interior, os *NLP's* resultantes dos métodos simultâneos podem ser resolvidos eficientemente (Biegler, 1984; Renfro *et al.*, 1987; Cervantes e Biegler, 1998; Biegler *et al.*, 2002).

Os métodos diretos são menos precisos que os indiretos e a qualidade da solução depende fortemente da parametrização (discretização) das variáveis de controle. Apesar disso, os métodos diretos não exigem precisão numérica muito grande para resolver o problema de otimização, mas podem resultar em soluções ótimas suspeitas. Uma grande desvantagem dos métodos diretos é que a parametrização de entrada é usualmente escolhida de forma arbitrária pelo usuário. Na realidade, devido à forma de discretização do problema, o *NLP* resultante pode apresentar mínimos locais, obtendo trajetórias ótimas relativamente diferentes das trajetórias ótimas reais.

O processo de discretização é um dos pontos fracos do uso desta abordagem. Para se aproximar da precisão dos métodos indiretos, é necessário construir uma malha muito refinada. Além disso, se tiver poucos elementos discretos, perdem-se graus de liberdade e

conseqüentemente oportunidades de otimização. Além disso, com uma malha de poucos elementos, pode-se não satisfazer as condições de otimalidade do *DAOP* contínuo (Hamiltoniano nulo). A eficiência do método e a precisão da solução dependem fundamentalmente da maneira como as entradas são parametrizadas (distribuição e quantidade de pontos discretos). Uma maneira de minimizar os efeitos desta escolha seria a utilização de estratégias de adaptação de malhas discretas (a ser apresentada mais adiante). Além disso, antes de resolver o problema de otimização dinâmica, não é possível saber os instantes em que as restrições se tornam ativas ou inativas, sendo necessário primeiramente resolver o problema de otimização e depois verificar a estrutura de arcos da solução.

Os métodos diretos exigem grande quantidade de memória para computar o controle ótimo, ao resolver problemas de grandes dimensões. Mas, mesmo assim, os métodos diretos seqüenciais e simultâneos são de longe os métodos de escolha. Eles não são muito sensíveis às estimativas iniciais dos perfis de controle. O problema se torna mais fácil de resolver quando ambos os estados e controles são parametrizados. Os métodos seqüenciais e simultâneos são os mais usados para resolver os problemas de controle ótimo em processos químicos.

Nos métodos seqüenciais, os erros controlados pelos integradores são usados para controlar sua precisão. De forma diferente, nos métodos simultâneos a precisão da quadratura está diretamente ligada à qualidade da discretização das variáveis. Na utilização de métodos simultâneos, deve-se ter um equilíbrio entre aproximação e otimização (Srinivasan *et al.*, 1995). Uma aproximação menos precisa na quadratura leva a um menor custo computacional. Por outro lado, uma melhor precisão na quadratura é obtida através do aumento do número de pontos de colocação, especialmente quando o sistema é rígido, onde se precisa discretizar o horizonte de otimização com uma grade muito fina, que se traduz em um grande número de variáveis de decisão, e um esforço computacional bem maior (Terwiesch et al., 1994).

Cabe ressaltar que os métodos indiretos e diretos são análogos para problemas de otimização dinâmica "*bem comportado*s", As abordagens diretas tendem a ser análogas às abordagens indiretas quando o número de subintervalos tende a infinito ou o maior subintervalo tende a zero (vide Figura 3.16). Tanartkit e Biegler, já em 1995 demonstraram a equivalência entre o método direto por colocação Radau e os métodos indiretos. Neste caso, as condições de *KKT* dos *NLP's* simultâneos são consistentes com as condições de otimalidade do problema variacional. Além disso, os multiplicadores de Lagrange fornecem uma boa estimativa dos valores das variáveis adjuntas correspondentes nos métodos indiretos. Porém, esta propriedade sozinha não implica que a seqüência de soluções aproximadas irá convergir como um todo. Vários estudos relatam problemas de estabilidade devido à discretização ruim, restrições de desigualdade de índice elevado e arcos singulares. Ambas as abordagens têm sido aplicadas com êxito para resolver problemas de controle por faixas em problemas de transferência de calor, o controle ótimo de bio-reatores em batelada e reatores químicos semi-contínuos.

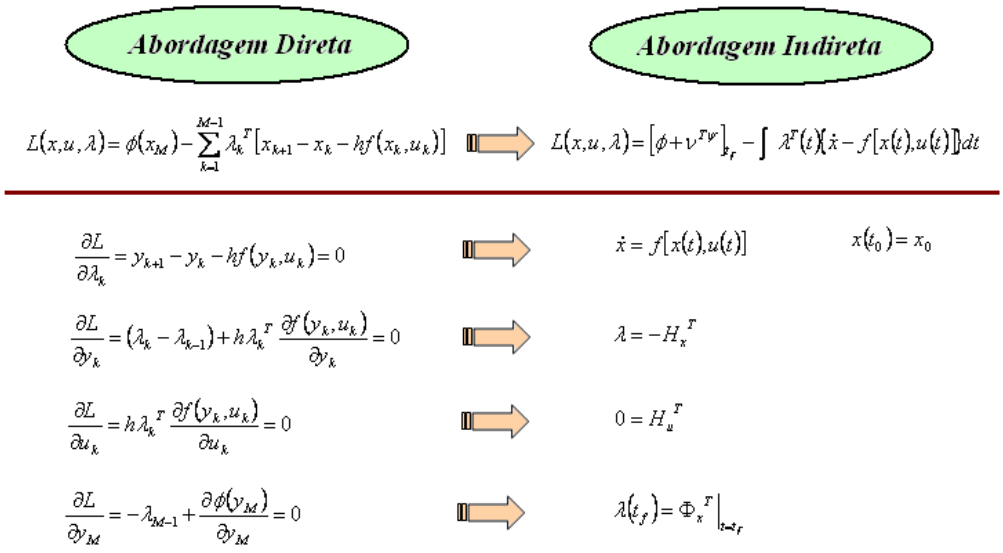


Figura 3.16 – Métodos Direto vs Indireto.

Outra abordagem possível é a utilização do método resultante da combinação de características dos métodos diretos e indiretos. Os métodos diretos geram *NLP's* de grande dimensão e a obtenção de gradientes da função objetivo resultante muitas vezes não é trivial. Por outro lado, há dificuldades de convergência dos métodos indiretos e a obtenção das equações de co-estado não é uma tarefa fácil. A existência de restrições pontuais ou de limites implica na introdução de multiplicadores e condições de complementaridade, que aumentam significativamente as dificuldades de solução dos *TPBVP's* pelos métodos indiretos. A utilização de métodos indiretos associado a métodos numéricos (ex.: *single-shooting*, *multiple-shooting*, etc.) podem apresentar dificuldades de convergência, devido ao fato de não se conhecer ou de não se ter qualquer idéia dos perfis das variáveis adjuntas e por conseqüência em fornecer uma estimativa inicial apropriada para essas variáveis. Métodos diretos, como colocação direta, não sofrem esse problema, mas geram resultados de menor precisão ou a iteração pode terminar com uma solução sub-ótima, pois o *NLP* resultante pode ter mais de um mínimo. Para superar essas dificuldades, a abordagem direta-indireta procura utilizar os méritos dos métodos indiretos e diretos de solução de DAOP (Stryk e Bulirsch, 1992), que procura reunir as vantagens de convergência do método direto, como colocação direta, com a precisão e confiabilidade do método indireto, com algoritmo *multiple-shooting*, através da utilização dos perfis de estado e co-estado obtidos pela abordagem direta onde se utiliza um problema de *NLP* simplificado, sem a inclusão das restrições de desigualdade, e aplicam-se estes perfis como estimativas iniciais para a abordagem indireta.

3.2 Softwares de otimização dinâmica

Há vários *softwares* de solução de *DAOP* disponíveis no mercado. Podem-se encontrar produtos comerciais (*COTS - Commercial Off-The-Shelf*) e softwares acadêmicos. Os *softwares* ofertados no mercado estão voltados para a solução de problemas de otimização dinâmica no modo *offline*. Alguns deles são oferecidos na forma de rotinas de programas,

outros com interfaces homem-máquina. Neste conjunto de softwares, há alguns que são abertos e livres para uso.

Estes *softwares* se diferenciam quanto aos aspectos dos métodos de solução de *DAOP* (vide Tabela 3.1), onde se destacam os métodos de discretização parcial (*DP*), com o método *single-shooting* (*SSH*), uma quantidade razoável de softwares que utilizam os métodos híbridos (*HD*), onde se tem o método *multi-shooting* (*MSH*), e os métodos de discretização total (*DT*) ou simultâneos. Neste caso, se destacam os métodos de colocação total (*C*), utilizando diferentes técnicas de quadratura, e colocação ortogonal em elementos finitos (*CEF*). Além disso, os softwares se diferenciam quanto ao tipo de problema *DAOP*, podendo ser de simples estágio (*SE*) ou de múltiplos estágios (*ME*). Há alguns softwares onde são ofertadas rotinas para problemas *SE*, que podem ser adaptados para resolver problemas *ME*.

Os diferentes softwares utilizam basicamente três tipos de ambientes de modelagem para construção do *DAOP*. Alguns pacotes utilizam intefaces com o *Matlab* (Mathworks, 2011), onde as rotinas e funções são escritas em linguagem *Matlab* (*ML*). Outros pacotes utilizam ambientes próprios de modelagem, tais como o *gPROMS* (*gP*). E um conjunto de softwares, geralmente acadêmicos, onde o usuário programa as funções do modelo *DAOP* em *FORTRAN* (*FOR*) ou *C/C++*, assim como também o programa principal de solução de otimização dinâmica (em alguns casos).

É mais apropriada a utilização de um ambiente de modelagem e simulação dinâmica orientada a objetos, onde podem ser destacados os ambientes *EMSO* (Soares et al., 2003; Soares, 2007), *gPROMS* (PSE, 2004), *Modelica* (Tiller, 2001), dentre outros. Estes ambientes têm linguagens próprias de modelagem de processos que se assemelham às memórias de cálculos em engenharia química. Isto faz com que o usuário tenha mais facilidades na modelagem de processos e no seu uso. O *software EMSO* atende a todos os requisitos de modelagem orientada a objetos e algoritmos confiáveis para solução de *DAE's*. A construção do *DAOP* pode ser efetuada pela utilização de arquivos de configuração ou até mesmo do próprio ambiente de modelagem de processos. Neste caso, o sistema deve possuir um interpretador de linguagem de otimização dinâmica. O modelo do *DAOP* deve ser formulado de forma a descrever a função objetivo, restrições, variáveis de controle, estágios do problema e definições dos algoritmos de soluções do problema. Estas definições devem ser tais que consiga representar todas as classes de problemas mencionadas anteriormente.

Alguns pacotes têm algumas funções adicionais para refinamento da solução ótima. Há *sotwares* que executam adaptações de malhas discretas (*MA*), que utilizam técnicas de elementos finitos móveis (*EFM*) ou *wavelts* (*WL*). Além disso, há softwares que realizam a detecção da estrutura (*SD*) da solução ótima, sendo utilizada a técnica de análise das condições necessárias de otimalidade (*NCO*). Praticamente, o *DyOS* é o único software que executa esta função embutida no pacote, apesar de haver a possibilidade de ser utilizada por outros softwares de forma customizada.

Os pacotes de solução de *DAOP*, baseados em métodos diretos, necessitam passar por um processo de discretização que resulta em um *NLP*. Os mesmos utilizam algoritmos de programação quadrática seqüencial (*SQP*), nas formas de espaço cheio e reduzido, e

algoritmos de pontos interiores (*IPT*). Os algoritmos de otimização *NLP* usualmente utilizados na solução de *DAOP* discretizados são *KNITRO* (*KN*), *IPOPT* (*IP*), *CONOPT* (*CN*), *SNOPT* (*SN*), *NPSOL* (*NP*), *SRQPD* (*SR*), *Matlab* (*fmincon*), *MSSQP* (*MQ*), *PRSQP* (*PR*), *FFSQP* (*FF*), *SPRNLP* (*SP*) e *BARNLP* (*BAR*).

Na realidade, há mais pacotes de otimização dinâmica disponíveis no mercado. Alguns de menor expressão e outros com enfoques e aplicabilidades bem diferentes dos tipos de problemas aqui discutidos. Há também alguns *softwares* que estão em estágios experimentais de desenvolvimento e não têm sido usados freqüentemente pela comunidade científica.

Tabela 3.1 – Softwares de otimização dinâmica.

Item	Software				Tipo de Problema	Modelagem	Funções		NLP		Referência
		DP	HD	DT			MA	SD	SQP	IPT	
1	SOCS	-	-	C	SE/ME	FOR	EFM	-	SP	BAR	Betts e Huffman, 1997
2	DIRCOL	-	-	C	SE	FOR	-	-	NP, SN	-	von Stryk, 1999
3	DIRMUS	-	MSH	-	SE	FOR	-	-	NP	-	Hinsberger, 1996
4	BNDSCO	-	MSH	-	SE	FOR	-	-	-	-	Oberle e Grimm, 1989
5	MUMUS	-	MSH	-	SE	FOR	-	-	-	-	Hiltmann et al., 1993
6	MUSCOD-II	-	MSH	-	SE/ME	gP,FOR/C	-	-	MQ, PR	-	Diehl et al. 2001
7	MISER3	-	MSH	-	SE	FOR	-	-	NP, FF	-	Fisher et al., 2004
8	RIOTS_95	SSH	-	-	SE	Matlab	EFM	-	NP	-	Schwartz et al., 1997
9	DYNOPT	-	-	CEF	SE	Matlab	-	-	ML	-	Cizniar et al. 2006
10	PSOPT	-	-	C	SE/ME	Matlab	EFM	-	SN	IP	Becerra 2009
11	GPOPS	-	-	C	SE/ME	Matlab	EFM	-	SN	IP	Rao et al - 2010
12	DyOS	SSH	MSH	-	SE/ME	gPROMS	WL	NCO	SN, NP	-	Brendel et al., 2008
13	CAMTOS		MSH	C	SE/ME	FOR	-	-	SN	-	P. F. Gath, 2002
14	DYNOPC	-	-	CEF	-	FOR/C	EFM	-	-	IP	Lang e Biegler, 2007
15	COOPT	-	MSH	-	SE	FOR	-	-	SN	-	Serban, 2002
16	PROPT	-	-	C	SE/ME	Matlab	-	-	CN,SN,CP	KN	Rutquist e Edvall, 2010
17	gOPT	SSH	MSH	-	SE	gPROMS	-	-	SR	-	Vassiliadis, 1992
18	ACADO	SSH	MSH	-	SE	Matlab	-	-	ML	-	Ariens, 2010
19	NUDOCCCS	-	MSH	C	SE	FOR	-	-	NP	-	Büskens, 1996

Tem-se observado é que os pacotes existentes no mercado individualmente, não possuem todas as facilidades e flexibilidades desejadas para um sistema de *DRTO*. Portanto, a opção por incorporar simplesmente um dos pacotes existentes no mercado representaria uma perda de funcionalidade e flexibilidade do sistema de *DRTO*, no sentido de ter opções de algoritmos para os pontos acima referidos. Com isso, sugere-se que se utilizem as melhores características de cada software.

A estratégia de utilização de *plug-ins* proporciona mais aplicabilidade e flexibilidade ao *software*. Lembre-se que esta não tem sido uma prática usual nos *softwares* conhecidos de otimização dinâmica.

3.3 Soluções em tempo real de otimização dinâmica

A otimização dinâmica em tempo real (*DRTO*) é uma aplicação *online*, onde se coletam informações de instrumentos da planta e se realiza o cálculo da trajetória ótima das variáveis de controle, com o auxílio de algoritmos de otimização dinâmica. Os valores das ações de controle calculadas, por sua vez, são implementados na planta, alterando suas condições de operação.

As aplicações de otimização dinâmica, em processos industriais, usualmente envolvem modelos não lineares de grandes dimensões. E um sistema de *DRTO* deve ser robusto, preciso, eficiente e aplicável em linha. Além disso, estes sistemas devem ter a habilidade de lidar com incertezas originadas dos erros de medições, nos parâmetros e até mesmo deficiências na representação do modelo, inclusive perturbações não medidas. No intuito de obter tal tipo de solução, a comunidade científica tem apresentado algumas opções para lidar com estes tipos de problemas.

A seguir, são apresentadas algumas das propostas mais relevantes de estruturas de aplicações em tempo real encontradas na literatura aberta. Vale lembrar que há pequenas mudanças de funcionalidades propostas para os sistemas de otimização em tempo real que não alteram as estruturas básicas deste tipo de sistema.

3.3.1 Estrutura geral do *DRTO*

Os sistemas de *DRTO* têm uma estrutura básica onde se estima o estado da planta, otimiza o modelo dinâmico do processo, e implementa as trajetórias ótimas de controle. A definição da estrutura de um sistema de *DRTO* depende de vários fatores operacionais. Isto ocorre devido a certas dificuldades encontradas em problemas industriais de engenharia química. Além disso, têm-se incertezas associadas aos recursos utilizados pelo sistema de *DRTO*. Em função desses fatores, escolhe-se a melhor estrutura de sistema de otimização em tempo real.

Fatores operacionais de otimização

Os fatores que levam a diferentes estruturas de otimização dinâmica são os seguintes: a natureza do processo a ser otimizado; o tipo de modelo; a política de operação; o nível de monitoração do processo; e o método de otimização utilizado;

Quanto à ***natureza do processo***, podem-se ter processos em batelada, semi-batelada e contínuos na indústria química. Em funções das características destes processos, a soluções de otimização poderão ter características diferentes em função das dificuldades de medição, modelagem e otimização de cada tipo de processo. A seguir serão apresentadas algumas características de operação e otimização destes processos.

Em **processos em batelada e semi-batelada**, a carga é processada segundo uma receita pré-estabelecida. Na elaboração da receita de produção, são estabelecidas as faixas de concentrações, vazões, temperaturas para uma determinada reação ou separação. Uma vez estabelecida a sua receita, é construído o plano de produção, onde é feita a alocação de recursos e estabelecida uma programação de produção para satisfazer as demandas dos produtos. Nesta etapa as operações unitárias são alocadas aos equipamentos e definida a seqüência de operações. Finalmente, é realizada a receita de produção de forma eficiente e segura, onde as variáveis de processo são ajustadas de acordo com o procedimento estabelecido. A produção pode então ser otimizada de forma a respeitar os critérios de segurança e as restrições operacionais.

Os processos em batelada são de natureza repetitiva, onde os resultados de bateladas anteriores podem ser usados para aprimorar o controle e otimização do processo. Este tipo de processos tem dinâmicas lentas e rápidas no mesmo sistema e não tem estado estacionário ao longo do processamento. Normalmente tem comportamento não linear, principalmente se envolver processo de reação. Nestes processos, o estado inicial é muito diferente do estado final. As concentrações, velocidades de reação, volatilidades relativas e outros variam muito durante a batelada. Os ganhos de processo, constantes de tempo ou tempo morto variam ao longo da batelada.

Muitas vezes estes processos têm comportamento de natureza irreversível. Há propriedades de produtos de processos de polimerização e cristalização, por exemplo, onde é impossível corrigir as características do produto quando o material fica fora de especificação. Como as bateladas têm tempo de operação finito, as ações corretivas são limitadas e às vezes uma batelada tem que ser descartada. Estes processos também estão sujeitos às perturbações do tipo: erros operacionais, problemas de processo (falhas de sensores), perturbações desconhecidas e ambientais. Isto faz com que tenhamos incertezas inerentes ao processo.

Os **processos contínuos**, diferentemente dos processos em batelada, são constituídos de número de modos de operação (estados estacionários ou quase-estacionários) e transições de processo (operações transientes). As mudanças de modos são provocadas usualmente por mudança na composição da carga e especificações de produtos, partidas e paradas de plantas, mudança de quantidade de carga e intervenções de manutenção. As refinarias de petróleo, por exemplo, operam com 10 a 30 tipos diferentes de petróleos e ocorrem 3 a 5 transições por semana. Durante as transições de processo, o operador deve fazer um número de intervenções complexas onde a planta perde rendimento e qualidade de produto (pode levar a produtos fora de especificação ou até mesmo causar acidentes) e opera em condições sub-ótimas. Estas operações de transições podem levar de 4 a 12 horas.

Quanto ao ***modelo do processo***, o desenvolvimento de modelos confiáveis usualmente consome muito tempo. Em processos de reações de química, bio-produtos ou polímeros, há um número significativo de reações que são freqüentemente desconhecidos na sua totalidade, sendo efetuada uma pequena quantidade de experimentos para modelar os processos de reação, de mistura e transferência de massa e calor. Essas dificuldades resultam em incertezas nos modelos de processo. Esses problemas fazem com que a atualização do modelo através do feedback da planta seja um dos fatores importantes para o sucesso da aplicação de otimização dinâmica de processos.

A ***política de operação*** otimizada consiste em encontrar o melhor ponto de operação ou perfis ótimos de operação da planta, na buscar da excelência operacional (Bonvin, 1998). Os seguintes aspectos são importantes na otimização de processos:

- **Segurança:** são restrições do sistema relacionadas aos limites de projeto dos equipamentos e emissões. Por exemplo: em reatores em batelada, deve ser evitado o disparo de reação, superaquecimento, polimerização indevida, reações paralelas desconhecidas;
- **Qualidade de produtos:** são restrições de especificação de qualidade de produtos, com baixa variabilidade nas suas características;
- **Escala de Produção:** Usualmente são ligados aos objetivos de produção, tais como a redução de estoques intermediários, menores tempo de entrega;

- **Produtividade:** são objetivos relacionados ao aumento de produção, redução de tempo ocioso, redução do tempo de batelada ou transição de processo;
- **Flexibilidade:** nas operações de processos em batelada é usual a construção de plantas de múltiplos propósitos, para aumentar a rentabilidade das empresas. Em plantas contínuas, é comum encontrar instalações onde são realizadas múltiplas campanhas. É o caso de unidades de destilação de petróleo e plantas de polimerização.

A otimização de processo em batelada leva em conta as características transientes do processo. Esta otimização tem aspectos inerentemente dinâmicos. Como conseqüência direta, os objetivos de otimização são estabelecidos para o tempo final da batelada. Os objetivos operacionais no fim da batelada normalmente são ligados aos indicadores de desempenho do tipo: seletividade, pureza, contaminação admissível, lucratividade. Os objetivos de otimização comuns são: maximização da concentração de produtos, minimização do tempo de batelada, máximo rendimento, mínimo custo operacional, dentre outros.

Os problemas de otimização podem ter restrições de caminho e restrições de tempo final. As restrições de caminho estão ligadas aos aspectos de operação, de segurança e variabilidade da batelada. As restrições terminais estão ligadas aos aspectos de produção, de qualidade de produto e impacto ambiental.

A otimização de processos contínuos é feita de forma semelhante aos aplicados em processos em batelada. Os objetivos de otimização são estabelecidos usualmente para o tempo final de um determinado horizonte de otimização. Os objetivos de otimização comuns são: maximização da concentração de produtos, minimização do tempo de transição, mínima integral do desvio quadrático em relação à referência, máximo rendimento, mínimo custo operacional, dentre outros.

Usualmente, o horizonte de otimização é divido em estágios de tempo, separando os diferentes modos e transições de operação. Da mesma forma que em processos em batelada, os problemas de otimização podem ter restrições de caminho e restrições de tempo final. As restrições de caminho estão ligadas aos aspectos de operação, de segurança e variabilidade da batelada.

Quanto o ***nível de monitoração***, em processos químicos, há variáveis de processo que são monitoradas online onde há medições contínuas de vazão, nível, temperatura, pressão e outras variáveis, como informações analíticas, que normalmente são disponíveis *offline* (usualmente obtidas de analisadores *offline* ou análises de laboratório). Alem disso, podem-se associar cálculos inferenciais integrados ou não aos analisadores de processo.

Em muitos processos em batelada ou semi-batelada há pouca disponibilidade de medições no processo. Na maior parte destes processos, as medidas de qualidade normalmente estão disponíveis só no final da batelada. Este fato faz com que a função objetivo relacionada à qualidade de produto seja avaliada somente depois de encerrada a batelada. Como os processos em batelada são de natureza repetitiva, pode-se adotar uma estratégia de otimização onde a operação ótima é atingida após algumas bateladas. Este tipo de otimização é chamado de otimização batelada a batelada (Srinivasan et al., 2001). Em processos contínuos as medições são disponíveis online, mas em quantidade limitada.

Fontes de Incertezas na otimização

A otimização do processo está sujeito a incertezas ligadas às propriedades da carga, aos sensores do processo, à dinâmica do processo e ao seu modelo. ***Incerteza na carga*** significa que a qualidade ou quantidade da carga varia com o tempo. Entende-se por qualidade de carga as grandezas intensivas ligadas à carga. São elas a composição, temperatura, pressão, fração vaporizada, dentre outras. Entende-se por quantidade de carga as grandezas extensivas ligadas à carga. São elas a vazão de carga ou a quantidade adicionada em um equipamento em batelada ou semi-batelada. As ***incertezas nos sensores*** são relativas aos erros de medição dos instrumentos de processo. Erros aleatórios, sistemáticos e até mesmo falhas de instrumentos ou de leitura de dados. As ***incertezas no processo***, correspondem às perturbações no processo devido a correntes não medidas ou não controladas, condições ambientais, manobras de processo (alinhamentos, bombas, ...), até mesmo intervenções de manutenção. E as ***incertezas no Modelo*** são aquelas relacionadas aos parâmetros do modelo, às hipóteses simplificadoras, à natureza do modelo (ex.: rigoroso, caixa preta, ...), à estrutura do modelo (representatividade do processo em questão)

Sistema de otimização a ser utilizado

A escolha do sistema de otimização a ser aplicado em um determinado processo depende das características do processo, do modelo do processo, da política de operação a ser otimizada, do nível de monitoração do processo, das incertezas envolvidas no problema a ser otimizado e das exigências associadas aos benefícios esperados da aplicação do sistema de otimização. Há processos contínuos onde não há mudanças na produção ou perturbações significativas, ou seja, não há mudanças no esquema de produção, e nem mudanças significativas na qualidade da carga e especificações de produtos. Neste caso a otimização estática *offline* pode ser executada de tempos em tempos, sendo suficiente para otimizar a planta. A mesma somente deve ser executada quando algum parâmetro do problema de otimização (preço ou parâmetro do modelo) ou a carga mudar significativamente. Quando se trata de processos em batelada ou contínuos onde há mudanças freqüentes de modo de produção ou perturbações significativas, o uso de otimização dinâmica proporciona maiores benefícios do que as aplicações de *RTO*.

A ***otimização estática*** é utilizada quando se configura certo cenário e o mesmo não muda ao longo de um determinado horizonte de tempo. Neste horizonte, não deve ocorrer mudanças de modo ou transição de processo. A otimização estática também se aplica a processos contínuos onde não há receitas de produção envolvida. De forma simplificada, pode-se representar um problema de otimização estática da seguinte forma:

$$\begin{aligned} &\min_{u(t)} \quad \phi(x,u) \\ &s.a. \\ &\qquad F(x,u,p)=0 \\ &\qquad S(x,u)\le 0 \end{aligned} \tag{3.72}$$

sendo o seu Lagrangeano e suas condições necessárias de otimalidade dadas por:

$$L(x,u,p,\lambda,\mu) = \phi(x,u) + \lambda^T F(x,u,p) + \mu^T S(x,u,p) \tag{3.73a}$$

e

$$L_u(x,u,p,\lambda,\mu) = \frac{\partial L}{\partial u} = \frac{\partial \phi}{\partial u} + \lambda^T \frac{\partial F}{\partial u} + \mu^T \frac{\partial S}{\partial u} = 0 \tag{3.73b}$$

Dado um cenário, ou seja, um determinado problema de otimização estática a ser resolvido sugere o ponto ótimo de controle. Este ponto ótimo pode ser implementado na forma de *setpoints* de controle regulatório, de valores alvo ou até mesmo faixas das variáveis controladas da camada de controle avançado. Ou até mesmo uma combinação dessas ações de controle do otimizador. O otimizador pode rodar em dois modos, são eles o modo malha aberta ou malha fechada. Em malha aberta o otimizador sugere a solução ótima, e o operador analisa e implementa toda ou parte da solução. Em malha fechada o otimizador implementa automaticamente a solução sugerida ou sugere ao operador que tem um tempo para criticar a sugestão do otimizador.

A ***otimização dinâmica*** é utilizada quando há mudanças de modos de operação na planta ou alguma receita de processamento muda ao longo de um determinado horizonte de tempo. Este tipo de situação é comumente encontrado em processos em batelada ou em transições em processos contínuos. Dado um determinado plano de transição ou uma receita, ou seja, um determinado problema de otimização dinâmica a ser resolvido, o otimizador sugere os perfis ótimos de controle. De forma semelhante à otimização estática, pode-se representar um problema de otimização dinâmica da seguinte forma:

$$\begin{aligned} &\min_{u(t)} \quad \phi(x(t_f)) \\ &s.a. \\ &\quad \dot{x} = F(x,u,p,t); \quad x(t_0) = x_0 \\ &\quad S(x,u,t) \le 0 \\ &\quad T(x(t_f)) \le 0 \end{aligned} \tag{3.74}$$

sendo o seu Hamiltoniano (Bonvin et al., 2005), sua função objetivo aumentada e suas condições necessárias de otimalidade dadas por:

$$H(x,u,p,\lambda,\mu t) = \lambda(t)^T F(x,u,p,t) + \mu(t)^T S(x,u,p,t), \tag{3.75a}$$

$$\varphi = \phi(t_f) + \nu^T T(x(t_f)) + \int_{t_0}^{t_f} \left[\lambda(t)^T F(x,u,p,t) + \mu(t)^T S(x,u,p,t)\right] dt \quad \text{e}$$

$$H_{u_i}(t) = 0 \qquad \therefore \qquad H_u(t) = \frac{\partial H}{\partial u} = \lambda^T \frac{\partial F}{\partial u} + \mu^T \frac{\partial S}{\partial u} = 0 \tag{3.75b}$$

Estes perfis ótimos podem ser implementados na forma de seqüências de *setpoints* de controle regulatório, de valores alvo ou de faixas das variáveis controladas da camada de controle avançado. Da mesma forma que no *RTO*, o otimizador pode rodar em dois modos, são eles o modo malha aberta ou malha fechada.

Otimização em tempo real são aquelas aplicações onde se monitora continuamente as mudanças do processo, que podem ser devido a alterações de modos de operação de processos contínuos (transições) ou devido à execução de uma receita em processos batelada ou semi-batelada. Uma aplicação em tempo real necessita de atualização do estado e modelo quando sujeito às incertezas devido às perturbações no processo (ex.: mudança de qualidade da carga), às mudanças de estrutura do problema de otimização a ser resolvido, aos erros de medição e incertezas no modelo do processo. Dependendo dos fatores citados acima, diferentes soluções podem ser adotadas. Em otimização dinâmica pode-se ter as seguintes opções: otimização nominal e otimização com incertezas.

Na **otimização nominal**, o estado da planta e o modelo do processo são fornecidos previamente (Figura 3.17). Neste caso, as incertezas na planta são desconsideradas. Normalmente são otimizações numéricas executadas de forma *offline* e os perfis ótimos não são alterados ao longo de um horizonte de operação. Esta solução é enviada para o nível de controle, onde o mesmo irá buscar o ponto ou trajetória de referência dada pelo otimizador.

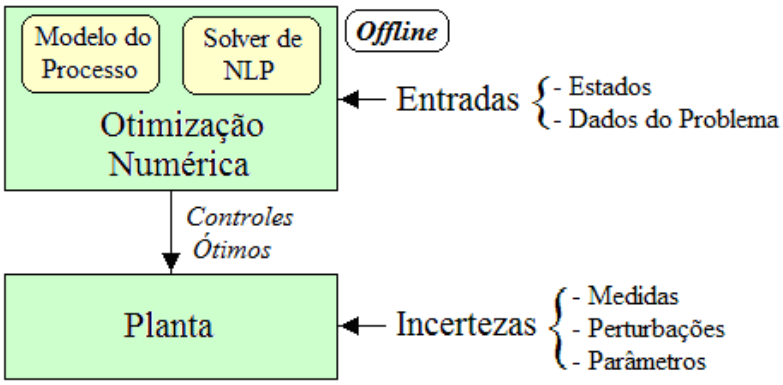


Figura 3.17 – Esquema da otimização nominal.

Na **otimização com incertezas**, os erros de medição, incertezas no modelo e as perturbações no processo passam a ser considerados. Dependendo da forma como os efeitos desses erros são considerados, pode-se classificar a solução de otimização em: otimização robusta e otimização baseada em medidas (Srinivasan *et al.*, 2002).

A ***otimização robusta*** é uma abordagem adotada quando as incertezas são consideradas no problema sem fazer atualização das medidas do processo. Normalmente, esta solução é mais conservativa que aquelas baseadas em medidas e procuram garantir a viabilidade da solução diante das incertezas esperadas. Uma opção seria levar em conta as incertezas de forma explícita (Terwiesch et al., 1994). O modelo nominal deve dar espaço a um modelo reformulado onde se leva em conta uma série de variações possíveis e realizações do sistema. Diferentes reformulações podem ser utilizadas, dependendo dos objetivos de produção e das informações disponíveis, As opções são: melhor valor esperado, variação mínima, limite absoluto, melhor pior caso e melhor caso, dentre outros (Terwiesch et al., 1994). As trajetórias das variáveis de controle são calculadas de forma *offline* uma vez, obtendo-se uma solução ótima, com incertezas, que são utilizadas em todas as bateladas subseqüentes ou para todas as transições em processos contínuos. Normalmente o problema de otimização se torna mais complexo do que a otimização nominal (Ruppen et. al., 1995; Diehl et al., 2005).

A ***otimização baseada em medidas*** consiste numa estratégia onde o estado da planta e os parâmetros do modelo são atualizados com as medidas do processo para compensar as incertezas associadas à otimização nominal. Isso geralmente é feito através de re-otimizações *online* (Agarwal, 1997; Kadam e Marquardt, 2004). Este tipo de problema ainda tem grandes desafios técnicos, porém as soluções tem tido progressos significativos (Binder et al., 2001a/b. Biegler et al., 2002). Tem-se observado duas fortes tendências nas soluções de otimização baseada em medidas, onde a otimização pode ser feita em um ou dois níveis, e podem ser classificados em: otimização explícita e otimização implícita.

Na **otimização explícita**, as medidas da planta são utilizadas para estimar o estado do processo e até mesmo ajustar os parâmetros do modelo (Figura 3.18). Após a atualização das informações da planta, é realizada a otimização numérica do problema em questão. Esta operação é feita toda vez que novos dados são obtidos da planta. Essa abordagem é muito utilizada em aplicações de otimização estática (*RTO – Real-Time Optimization*) e em controladores preditivos. Esta abordagem também é utilizada em otimização online de processos em batelada (Loeblein et al., 1997), soluções de *NMPC* (Biegler e Zavala, 2009; Zavala e Biegler, 2009) e em sistemas de *DRTO* propriamente dito (Kadam et al., 2004). Esta abordagem pode ter diferentes estruturas de software. Mais a frente, serão apresentadas algumas opções. Há uma diversidade de abordagens *NMPC* e propostas na literatura (Rotava, 1997 e Longhi, 2001). Este controlador usualmente recebe *setpoints* ou trajetórias ótimas de referência, originários de aplicações de *RTO* ou *DRTO*.

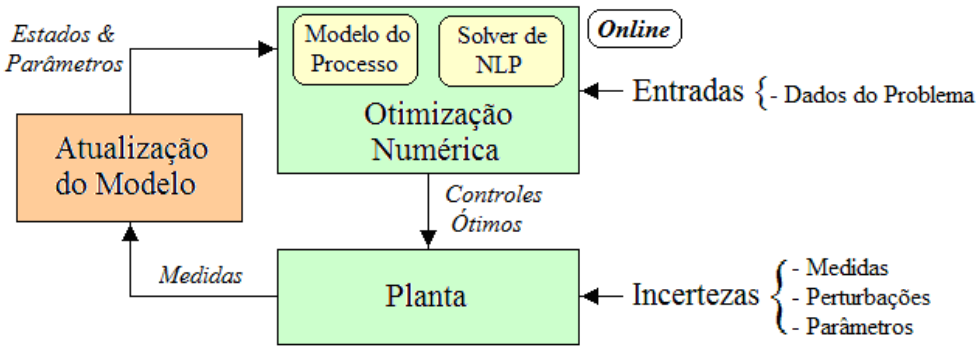


Figura 3.18 – Esquema da otimização explícita.

A **otimização implícita** (vide Figura 3.19) é uma metodologia onde se realiza a otimização numérica, normalmente *offline* e da mesma forma da otimização nominal. Após a realização da otimização nominal, analisam-se os perfis ótimos através da detecção da estrutura do controles (detecta partes fixas e livres dos perfis de controle). Esta estrutura da solução ótima é analisada através da dissecação das condições de otimalidade. Estas condições têm duas partes: uma referente à busca das restrições e outra referente à busca da otimalidade (sensibilidade nula do Hamiltoniano). Ambas as partes levam em conta as restrições de caminho e/ou de tempo final. Examinando a solução ótima, dissecam-se os perfis de controle e de estados separando-os nos seguintes arcos: controle no limite superior, controle no limite inferior, estado na restrição de caminho e estado dentro da região viável (arco de busca da sensibilidade zero). Supondo que as restrições ativas (de caminho e terminais) não se alteram com incertezas no modelo, a otimalidade pode ser alcançada através da sua perseguição dos arcos no nível de controle de processo, onde a

otimização é feita através da redução *online* dos desvios das restrições. Isto é feito com a escolha adequada da estrutura de controle que satisfaça as condições necessárias de otimalidade (*NCO*). Após a detecção da estrutura de arcos, o processo é controlado e otimizado utilizando as medidas da planta e perseguindo a estrutura de arcos de controle.

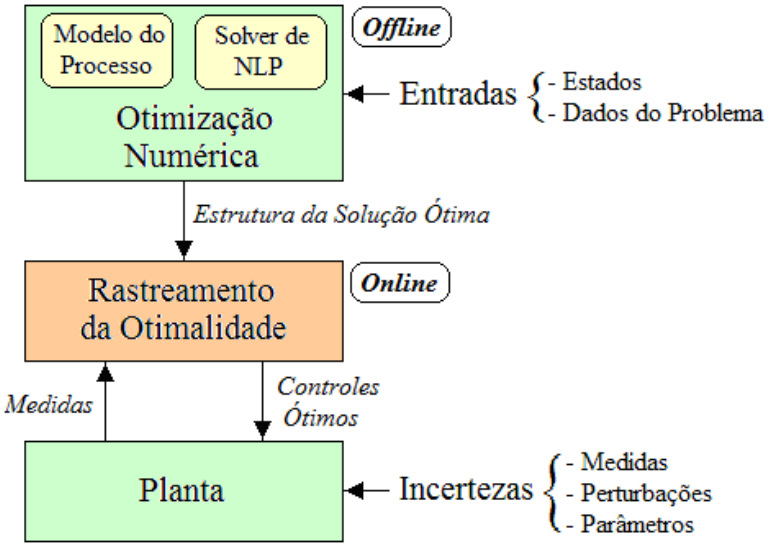


Figura 3.19 – Esquema da otimização implícita.

Ao realizar a análise da solução ótima, as variáveis de controle são particionadas em arcos η(t) que estão relacionados às restrições e objetivos no caminho e os escalares π que afetam as restrições e objetivos terminais (normalmente é o tempo final, quando o mesmo é livre).

Uma parte da solução ótima está ligada às restrições do problema de otimização e a outra está relacionada ao objetivo de otimização (vide Tabela 3.2). Esta forma de caracterização da solução ótima foi proposta por Srinivasan et al. (2002), que separa as variáveis de decisão em variáveis $\bar{\eta}(t)$ e $\bar{\pi}$ que buscam as restrições e variáveis de decisão $\tilde{\eta}(t)$ e $\tilde{\pi}$ que buscam a otimalidade (sensibilidade nula). Este particionamento, leva à seguinte separação das condições necessárias de otimalidade (*NCO*), como mostra a Tabela 3.3:

Tabela 3.2 – Condições de otimalidade da otimização dinâmica.

	Caminho	Tempo Final
Restrições	$\mu(t)^T S(x,u,p,t)=0$	$\nu^T T(x(t_f))=0$
Otimização (Sensibilidade nula)	$H_{u_i}(t)=0$	$H_{u_i}(t_f)=0$

Tabela 3.3 – Qualificação das restrições e otimalidade da otimização dinâmica.

	Caminho	Tempo Final
Restrições ativas	$\bar{S}(x,u,p,t)=0$	$\bar{T}(x(t_f))=0$
Otimização (Sensibilidade nula)	$\lambda^T F_{\tilde{\eta}} = \lambda^T \dfrac{\partial F}{\partial \tilde{\eta}} = 0$	$\varphi_{\tilde{\pi}} = \dfrac{\partial \varphi}{\partial \tilde{\pi}} = 0$

A primeira linha da tabela diz respeito às qualificações das restrições ativas, e a segunda se refere às condições de otimalidade.

Os arcos $\overline{\eta}(t)$, que buscam as restrições, são determinados de forma a manter as restrições de caminho ativas ($\overline{S}=0$) e os arcos $\tilde{\eta}(t)$ buscam as condições de otimalidade $\lambda^T F_{\tilde{\eta}}=0$. Por outro lado, as restrições terminais ativas ($\overline{T}=0$) determinam os parâmetros $\overline{\pi}$; enquanto os parâmetros $\tilde{\pi}$ de busca da sensibilidade nula são obtidos a partir das condições $\varphi_{\tilde{\pi}}=0$. Com essa dissecação da solução ótima, pode-se projetar o sistema de controle auto-otimizante (Skogestad, 2000; Govatsmark e Skogestad, 2005) que deverá rastrear as *NCO*, como o controlador de caminho e terminal.

Este projeto consiste em definir a estrutura de controle auto-otimizante, isto é, definir os pares de variáveis manipuladas e controladas que ligam as variáveis de decisão ao *NCO* (Bonvin *et al.*, 2005; François *et al.*, 2005; Bonvin e Srinivasan, 2003; Srinivasan e Bonvin, 2004; Srinivasan e Bonvin, 2007). E em seguida estabelecer as leis de controle regulatório ou avançado que fazem esta ligação (variáveis de decisão - *NCO*).

É interessante notar que se dispor apenas de um sistema de otimização estática, tem-se apenas a parcela das condições de otimalidade referente ao tempo final, como pode ser visto na Tabela 3.4 e 3.5 (Chachuat *et al.*, 2008a; Chachuat et al., 2009; François et al., 2005).

Tabela 3.4 – Condições de otimalidade da otimização estática.

	Ponto
Restrições	$\mu^T S(x,u,p)=0$
Otimização (Sensibilidade nula)	$L_{u_i}(x,u,p,\mu)=0$

Tabela 3.5 – Qualificação das restrições e otimalidade da otimização estática.

	Ponto
Restrições ativas	$\overline{S}(x,u,p)=0$
Otimização (Sensibilidade nula)	$L_{\tilde{\pi}}=\dfrac{\partial L}{\partial \tilde{\pi}}=0$

Desta forma, a sistemática semelhante à acima descrita também pode ser utilizada para rastrear as *NCO* dadas por uma otimização estática.

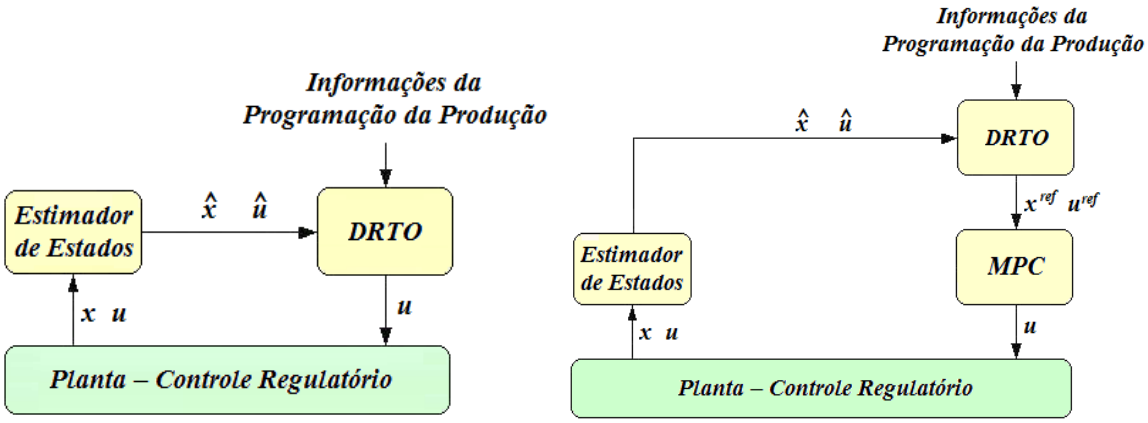
3.3.2 Estruturas alternativas do *DRTO*

Em função das funções desejadas e da infra-estrutura disponível na planta, há algumas variações de estrutura de *DRTO* em alguns aspectos específicos dos módulos do sistema ou na arquitetura do controle. As diferentes estratégias têm os seguintes módulos:

Separador de Escalas de Tempo – consiste de um módulo que separa as escalas de tempo com variações mais lentas, compatíveis com o horizonte de tempo do *DRTO*, ou com perturbações de constantes de tempo menor que o horizonte de predição do *MPC* (Kadam *et al.*, 2002 e Helbig et al., 2000). Desta forma, definem-se os tempos de amostragem para o ciclo do *DRTO* e para o ciclo do *MPC*.

Disparador do *DRTO* – consiste de um módulo que dispara a execução do *DRTO* com base na análise da sensibilidade do sistema às novas entradas do processo. Com base nesta sensibilidade, atualiza-se a predição e os multiplicadores de Lagrange de forma rápida e verifica se as variações relativas da função de Lagrange e das restrições de desigualdade foram maiores do que uma tolerância pré-estabelecida. Caso ambos os critérios sejam ultrapassados, dispara-se a execução do *DRTO*, caso contrário deve-se apenas executar o *MPC*.

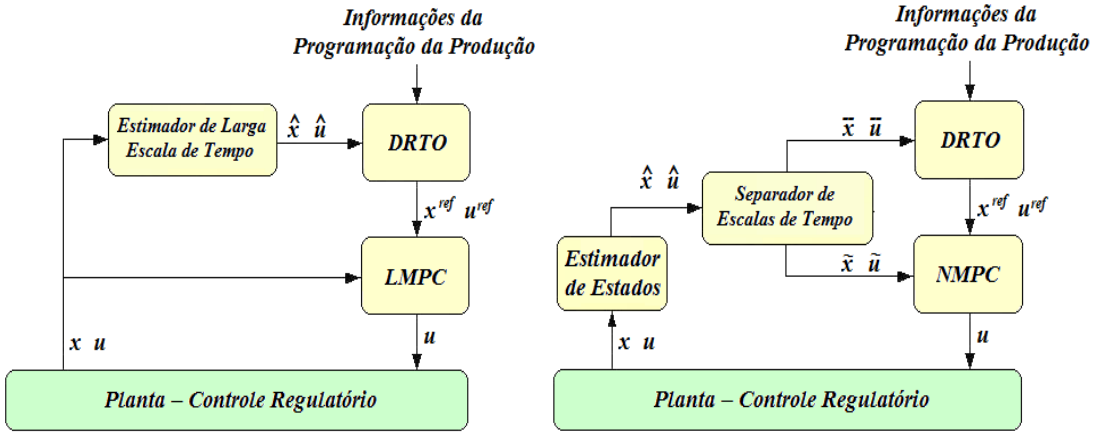
Estes blocos podem ser usados de diferentes formas e com graus de simplicidades diferentes. Os primeiros arranjos consistem em realizar a otimização dinâmica em uma camada, onde o controle e otimização são realizados juntos (Kadam e Marquardt, 2004), ou em duas camadas (Kadam e Marquardt, 2004), conforme a Figura 3.20. Estes arranjos são semelhantes ao utilizado em *RTO* (Zanin, 2001).



(a) (b)

Figura 3.20 – Esquema geral do *DRTO*, (a) uma camada, (b) duas camadas.

Em um segundo grau de complexidade, pode-se ter estimação dos estados para o *DRTO* e *MPC* ou somente para o *DRTO* (Backx *et al.*, 2000). No caso de um estimador para o *DRTO* e *MPC*, é necessário colocar um separador de escalas de tempo (vide Figura 3.21).



(a) (b)

Figura 3.21 – Estimador de estados.
(a) somente *DRTO*, (b) para o *DRTO* e *MPC*/*NMPC*.

O estimador de estados pode ser único para ambas as aplicações (Figura 3.21.b, Kadam *et al.*, 2002) ou diferentes (Backx *et al.*, 2000). Neste ultimo caso, o estimador do *DRTO* pode ter um modelo mais complexo (rigoroso) do que o modelo do *MPC*. Quando se utiliza estimadores distintos para as duas aplicações, o separador de escalas de tempo deve obrigatoriamente estar antes dos estimadores de estados, conforme mostra a Figura 3.22.

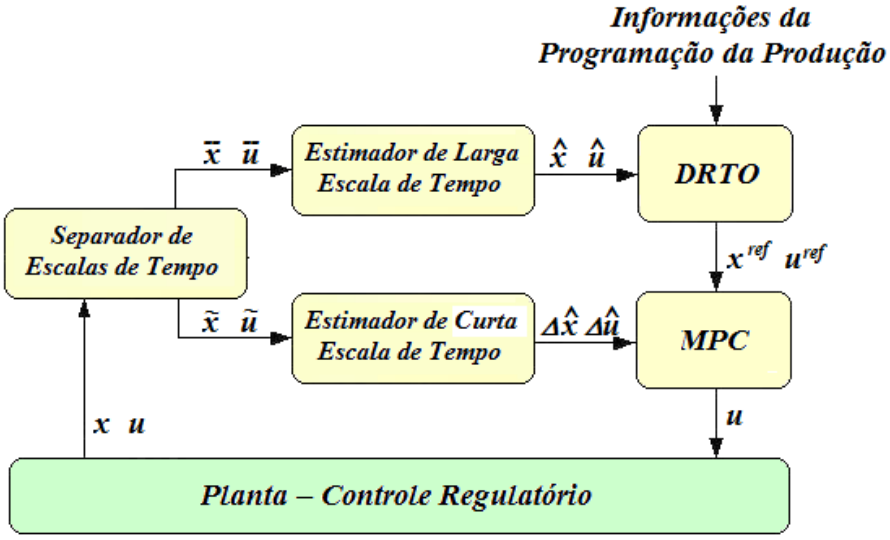


Figura 3.22 – Estimadores de estados distintos para *DRTO* e *MPC*.

O sistema de *DRTO* pode ser executado em um ciclo de execução pré-determinado ou ser executado somente quando há perturbação sensível no processo. Neste caso, é necessário utilizar um disparador de *DRTO* (Kadam *et al.*, 2003) para definir quando o *DRTO* deve ser executado (vide Figura 3.23).

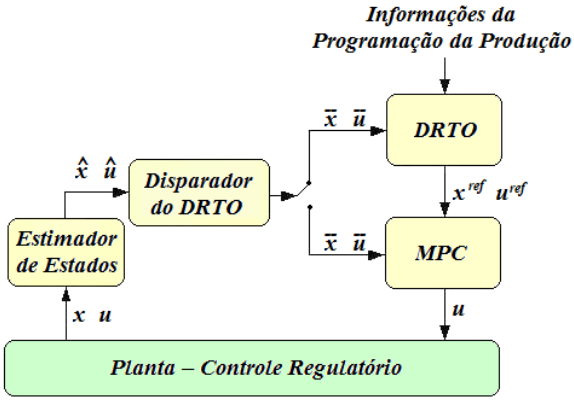


Figura 3.23 – Sistema *DRTO* com disparador.

Finalmente, o resultado do *DRTO* pode passar por uma validação final das ações de controle. As ações de controle do *DRTO* podem ser enviadas diretamente para o *MPC* ou para a os controladores da planta ou passam por um processo de análise dos resultados do otimizador. Neste caso último caso, as ações de controle devem passar por um teste de significância das movimentações de controle, isto é, verificar através de testes estatísticos se as ações de controle são realmente distintas das anteriores, conforme Figura 3.24 (Miletic e Marlin, 1998).

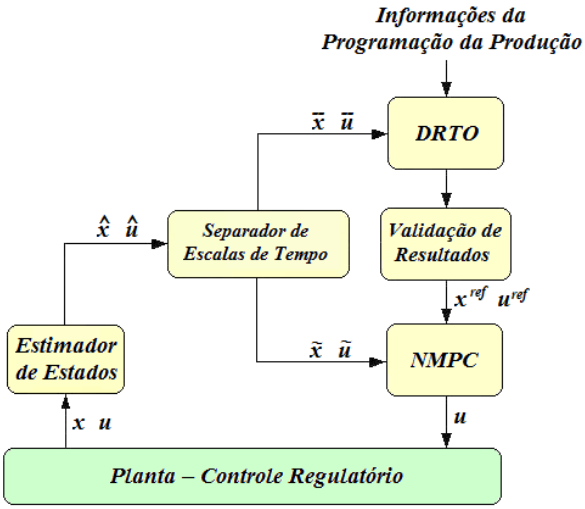


Figura 3.24 – Validação de Resultados do *DRTO*.

Em suma, há uma serie de opções de estrutura de sistema de *DRTO*, como conseqüência das funções necessárias do sistema. Esta decisão depende da natureza do problema a ser resolvido (complexidade e dinâmica) e da tecnologia disponível para a aplicação.

Uma classe diferente de aplicações é a ***otimização dinâmica de processos em batelada***. Esta otimização tem atraído o interesse de pesquisadores de longa data. Como exemplo, pode-se citar Robinson em 1969 e 1970, que apresentou estudos de otimização dinâmica em colunas de destilação em batelada, utilizando o principio de máximo de Pontryagin. Mujtaba (2004), em seu livro, relata uma série de estudos deste tipo de otimização de processos. Além desses processos, há uma série de estudos com sistemas reacionais e de cristalização. Em otimização dinâmica de reatores em batelada, há basicamente dois tipos de problemas, são eles: a maximização da conversão e minimização do tempo de batelada (Aziz e Mujtaba, 2002). Apesar desta quantidade de trabalhos, as soluções comerciais existentes no mercado consistem apenas de controladores regulatórios e sistemas de gerenciamento e operações em bateladas. Por outro lado, a comunidade científica tem apresentado e implementado industrialmente algumas soluções customizadas de otimização dinâmica, como apresentado por Sheikhzadeh *et al.* (2008) para controle e tempo real de processo de cristalização semi-batelada.

Têm sido propostas diferentes estratégias para otimizar processos em batelada, em função dos seus objetivos de produção e das características dos processos em batelada (Bonvin, 1998). Quando as medições do processo não estão disponíveis durante a batelada, é recomendável utilizar a estratégia de otimização batelada-a-batelada, também chamada de rodada-a-rodada (Srinivasan *et al.*, 2001, François *et al.*, 2002). Neste caso, destacam-se as dificuldades de modelagem e de medições de informações do processo, que resultam em incertezas no processo de otimização. A avaliação dos resultados da otimização pode ser efetuada somente ao final de cada batelada. Após cada batelada, as informações do processo são reconciliadas e usadas para ajustar o modelo de otimização e serem usadas na rodada seguinte. Desta forma, só se utiliza a otimização numérica *offline*, que define a estrutura da solução ótima. Esta estrutura é utilizada no projeto e ajuste do controle regulatório auto-otimizante que irá buscar as restrições e objetivos corretos. Esta estratégia é eficiente somente quando as bateladas são repetitivas.

Quando as medidas do processo estão disponíveis *online* e em tempo real, pode-se utilizar a otimização online direta (também chamada de otimização explícita). Esta é uma otimização baseada em medições (*MBO - Measurement Based Optimization*), onde se tem um horizonte de otimização definido, e restrições de caminho e tempo final. A freqüência de execução é bem menor do que o tempo de batelada, e as incertezas do problema são reconciliadas em tempo real. Normalmente a carga computacional é elevada e a aplicação tem características de controle preditivo não linear. Contando com as informações das medidas da planta e da freqüência de execução elevada, pode-se simplificar o modelo do processo para reduzir a carga computacional para resolver o problema mais rapidamente.

Quando a carga computacional é muito elevada, pode-se utilizar a estratégia de rastreamento das condições necessárias de otimalidade (também chamada de otimização implícita). Neste caso, pode-se executar a otimização nominal no modo *offline* e definir uma estratégia de controle regulatório. Esta solução é viável quando as bateladas têm natureza repetitiva. Recentemente, Srinivasan e Bonvin (2007) propuseram uma abordagem utilizando as condições necessárias de otimalidade (*NCO - Necessary Conditions for Optimality*) (Srinivasan et al., 2002 e Srinivasan et al., 2003) para projetar uma otimização dinâmica em duas camadas, onde no primeiro nível tem-se a otimização dinâmica baseado em modelos e no segundo nível um controle auto-otimizante livre de modelos e baseado na estrutura da solução ótima ou nas *NCO* (Chachuat *et al.*, 2008b e Chachuat *et al.*, 2009). Isto é efetuado com o rastreamento das *NCO*, realizado através da análise das condições de otimalidade obtidas da otimização offline, pela dissecação das *NCO* (detecção da estrutura da solução ótima). Com a estrutura de controle ótimo definida, projetam-se e implementam-se das estratégias de controle regulatório auto-otimizantes atualizadas que irão buscar as restrições e objetivos definidos pela otimização *offline*. Cabe lembrar que o modelo do processo é atualizado toda a vez que o problema de otimização dinâmica offline for resolvido.

Em alguns casos, tem se sugerido o uso de controladores preditivos não lineares - *NMPC* (Arellano-Garcia *et al.*, 2005) e otimização dinâmica em tempo real (*DRTO*), onde o *NMPC* minimiza o desvio de uma trajetória de referência e o *DRTO* procura encontrar a política ótima de operação do sistema (Würth *et al.*, 2009). O objetivo é atingir uma operação rentável e flexível, adaptada às novas condições de mercado e processar as incertezas, empregando um critério de otimização econômica. Recentemente a empresa *IPCOS NV* lançou um *NMPC* comercial (vide Figura 3.25) para processos em batelada (Landlust *et al.*, 2008 e Mesbah *et al.*, 2010). A arquitetura do sistema consiste em um controlador preditivo (*NMPC*), um observador (tipo filtro de Kalman estendido - *EKF*), um modelo hibrido (empírico e rigoroso) e uma interface de comunicação *OPC*. Este controlador foi inicialmente aplicado a um processo de cristalização em batelada industrial. Este sistema é composto de controle intra-batelada e outro inter-batelada (Vandecraen *et al.*, 2007). O objetivo do controle intra-batelada é a minimização no tempo de batelada e a redução da variabilidade do processo, enquanto que o controlador inter-batelada, a cada batelada, atualiza os parâmetros de operação do processo (*setpoints*, restrições) com o objetivo de obter a melhor qualidade do produto. Esta é uma arquitetura mais próxima das propostas apresentadas recentemente pela comunidade científica.

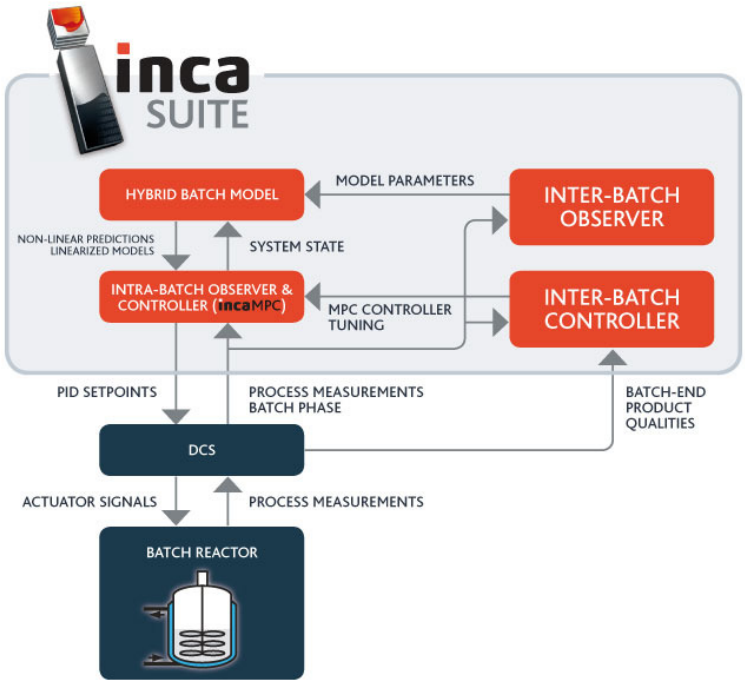


Figura 3.25 – Arquitetura do *MPC* para processos em batelada da *IPCOS* (Brochura do *INCA* e Vandecraen *et al.*, 2007).

3.4 Funções e métodos utilizados em sistemas de otimização dinâmica

Um sistema de *DRTO* utiliza uma série de métodos para executar as diferentes tarefas, e envolve a adaptação da malhas em otimização dinâmica e agrupamento de elementos para controle, a detecção da estrutura da solução do *DAOP*, o mecanismo de disparo do otimizador, os algoritmos de solução do *NLP* do problema discretizado.

Além da infra-estrutura de otimização, também há alguns métodos de monitoração e diagnóstico de sistemas de controle e otimização. São relatadas aqui algumas iniciativas de solução de problemas do otimizador, onde é descrita a metodologia *DMAIC* (abreviatura de *Define, Measure, Analyze, Improve, Control*) com foco em *DRTO*, e são apresentados os tópicos de avaliação de desempenho do otimizador, análise de sensibilidade da solução e alguns pontos na avaliação do erro de quadratura. A seguir são apresentados estes tópicos.

3.4.1 Agrupamento de Elementos para Controle

Para atingir a precisão desejada na discretização, muitas vezes é necessário aumentar o número de elementos finitos. Como conseqüência, os perfis das variáveis de controle podem apresentar muitas descontinuidades indesejadas. Isto pode ser inconveniente para aplicação em tempo real, pois se teriam intervalos muito curtos entre duas ações de controle. Nas plantas industriais, normalmente deseja-se manter as variáveis de controles constantes por um intervalo de tempo maior. Para resolver este problema, pode-se utilizar o recurso de agrupamento de elementos para controle, disponível no software *DynoPC*

(Lang e Biegler, 2005). Este recurso é similar ao agrupamento utilizado em controladores preditivos para distribuir melhor as ações de controle ao longo do horizonte de predição.

O agrupamento de elementos consiste em manter as variáveis de controle constantes dentro dos elementos contidos em cada agrupamento como podem ser observado na Figura 3.26. Os tamanhos dos grupos de elementos devem ser compatíveis com as ações da operação. Com isso, tem-se a liberdade para colocar quantos elementos finitos forem necessários para obter a precisão desejada para os perfis discretizados das variáveis de estado e algébricas do problema de otimização.

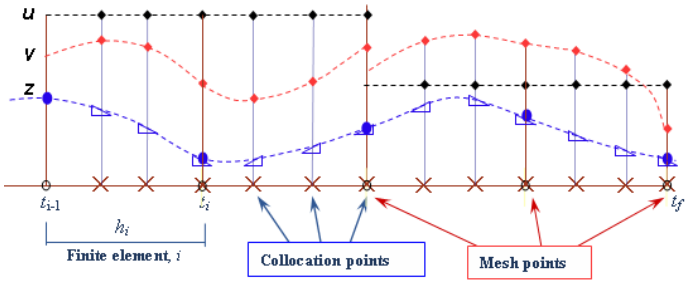


Figura 3.26 – Agrupamento dos elementos finitos (Lang e Biegler, 2005).

3.4.2 Adaptação da malhas em otimização dinâmica

A discretização é uma etapa importante na obtenção da solução ótima do *DAOP* pelos métodos diretos. Uma discretização inadequada leva às imprecisões nas soluções e perdas de oportunidades de otimização. As imprecisões ocorrem devido aos erros de quadratura e não atendimento das condições de otimalidade de forma precisa. E a perda de oportunidades está associada à redução de graus de liberdade para otimizar o processo e imprecisão na determinação dos instantes de chaveamento dos arcos de controle (apresentados no tópico a seguir). Por outro lado, o processo de discretização também afeta o desempenho e robustez do otimizador. Uma malha com mais elementos discretos resulta em um problema de *NLP* de dimensões maiores e conseqüentemente maior consumidor de recurso na máquina (mais iterações e mais tempo de *CPU*).

Devido a estes fatos, há um grande interesse na adaptação de malhas discretas na solução do *DAOP*. Este é um tópico que está em processo de maturação cientifica. Este assunto também é muito abordado na área de fluidodinâmica computacional (*CFD*), porém aqui serão apresentados alguns métodos voltados para adaptação de malhas na solução de *DAOP's*.

A precisão numérica de uma simulação dinâmica está diretamente ligada à ordem da aproximação e da distribuição dos pontos na malha discreta. Uma prática comum é avaliar e melhorar a qualidade da malha baseada em uma dada característica geométrica e então realizar a adaptação da malha. Uma malha deve ser adaptada para reduzir oscilações associadas à resolução inadequada de gradientes elevados, melhorando as formas para melhor representar o sistema. A adaptação dinâmica é uma área de fronteira na geração de grades numéricas e tem provado ser um dos pontos importantes na obtenção da boa qualidade da integração de sistemas.

Há três estratégias básicas de adaptação de malhas (vide a Figura 3.27), são elas:

- ***Redistribuição da malha (r-refinamento)*** – Neste caso os pontos se movimentam de uma região de menor erro para outra de maior erro ou gradientes elevados, melhorando a aproximação local. A movimentação deve satisfazer o princípio da equidistribuição dos pontos. Neste caso, o número de nós permanece o mesmo, não aumentando o esforço computacional e requisito de memória. Normalmente os critérios de adaptação são: a suavização e a ponderação do erro.

- ***Refinamento local da malha (h-refinamento)*** – Neste caso, adicionam-se ou removem-se pontos localmente para uma dada estrutura na região de erros relativamente grandes. Neste caso, a estrutura da malha é preservada, tendo como desvantagem o aumento do esforço computacional e a necessidade de memória. É importante que o critério de adaptação resolva descontinuidades e suavidades da solução.

- ***Alteração do método de solução*** – Neste caso, altera-se a ordem da aproximação nas regiões de maiores erros ou gradientes sem a mudança da distribuição dos pontos. Esta ação procura melhorar a precisão global na integração, porém aumenta a complexidade da implementação.

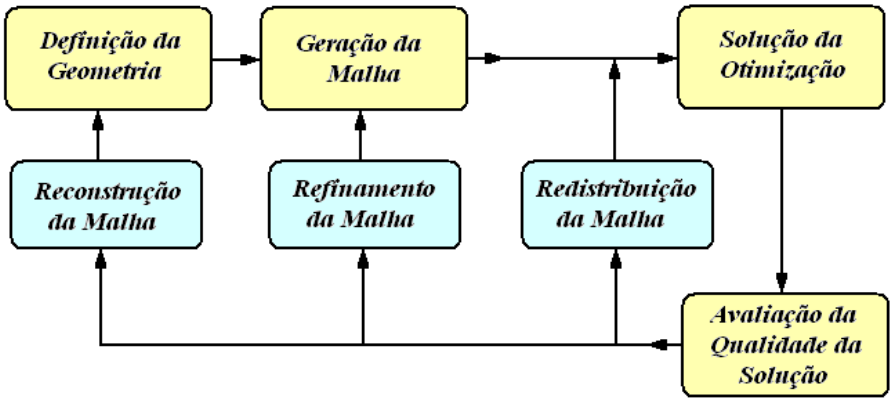


Figura 3.27 – Esquema geral de adaptação de malhas.

Às vezes é interessante combinar a redistribuição com o refinamento da malha numa mesma região ou em regiões diferentes do domínio discretizado.

Pelo fato da qualidade da discretização ter um reflexo direto sobre o resultado da otimização dinâmica, houve um grande interesse na otimização da adaptação de malhas. Há algumas abordagens diferentes na otimização da adaptação de malhas na solução de problemas de otimização dinâmica.

Aqui são considerados três métodos que mais se destacam na adaptação de malhas na solução da *DAOP*. Esta escolha levou em conta as diferentes abordagens e seus sucessos nas aplicações.

Adaptação pelo Método de Betts

Este é um método de refinamento de malha que parte de uma malha grosseira (ex.: 5 elementos) e aumenta o número de pontos discretos igualmente espaçados usando o critério de erro de discretização. Com o processo de refinamento da malha, o erro de discretização reduz até atingir um nível de tolerância aceitável. Quando o número de elementos da malha é aumentado, ou seja, comprimentos dos elementos finitos tendem a zero, as condições de *Karush-Kuhn-Tucker* (*KKT*) satisfeitas no problema discretizado (*NLP*) se equivalem às condições necessárias de otimalidade (*NCO – Necessary Conditions of Optimality*) do problema contínuo (*DAOP*). Esta é uma estratégia de obtenção de solução do problema de otimização dinâmica em duas camadas, onde na camada externa é realizado o refinamento da malha e na interna é executada a otimização dinâmica convencional com uma dada malha proposta pelo processo de refinamento externo.

O algoritmo de Betts (Betts e Huffman, 1998) tem basicamente cinco etapas: otimização dinâmica convencional; construção da representação continua aproximada da solução do *DAOP* (perfis de controle e estado); estimação do erro de discretização; detecção da estrutura da solução (arcos) e refinamento da malha discreta.

Na primeira etapa, é realizada a otimização dinâmica convencional, utilizando algum método direto. Na primeira iteração é proposta uma malha grosseira com os elementos eqüidistantes. Em seguida é construída uma representação contínua e uma malha muito fina com o objetivo de avaliar os erros em pontos intermediários dos elementos finitos da malha discreta. Para isso, pode-se aplicar uma aproximação por *spline* cúbica de ordem *2ne* nos perfis de estado e controle, onde *ne* é o número de elementos da malha. Ao analisar o erro do processo de integração (seja pelo uso de integradores convencionais ou por quadratura). Define-se como erro local ε_k a diferença entre o valor estimado e a solução do sistema algébrico-diferencial, obtida por integração. Com a informação dos erros locais, calcula-se o erro médio da malha atual e realiza-se o teste de precisão da discretização, usando a tolerância e o erro de discretização. Neste caso, deseja-se que a nova malha tenha um erro previsto abaixo tolerância $\kappa\delta$. Em seguida, calcula-se o erro máximo ε_{max} ao longo de todos os intervalos e avalia a ordem *p* da precisão do método de integração da nova malha. Se este erro for inaceitável, então a malha discreta deve ser refinada.

Na etapa seguinte, a estrutura da solução ótima é detectada, onde os arcos identificados de forma a estimar os pontos de transição onde as restrições de estado se tornam ativas ou inativas, isto é, quando o arco entra e sai da restrição da restrição de caminho. Nestes pontos de transição, devem-se adicionar estes pontos na nova malha, onde o número de pontos de transição é m_t.

O objetivo é construir uma nova malha usando informações do erro de discretização da malha anterior. Suponha que se deseja subdividir a malha atual, em uma nova malha que irá conter os pontos da malha anterior. Considere o número de pontos (I_k) a serem adicionados no intervalo *k*, e calcule o erro local relativo estimado para a nova malha ($1+I_k$) com um ponto a mais. Com isso, um novo erro local ε_k estimado pode ser calculado sem resolver o *DAOP* novamente. Em seguida, a nova malha pode ser construída,

escolhendo o conjunto I_k através da solução do problema de otimização de programação inteira (Equação 3.76).

$$\begin{aligned}
&\min_{I_k} \left\{ \phi(I_k) = \max_k \varepsilon_k \right\} \quad k = 1, 2, ..., ne \\
&s.a. \\
&\sum_{k=1}^{ne} I_k = ne_1 \\
&I_k \leq ne_2
\end{aligned} \tag{3.76}$$

Ou seja, minimizar o máximo erro em todos os elementos na malha atual, adicionando no máximo $ne - 1$ pontos. Este caso deseja-se minimizar o erro em todos os intervalos de malha atual, adicionando no máximo ne_1 pontos, sendo que ne_2 é o número máximo de pontos a serem adicionados em um único elemento da malha anterior para evitar um número excessivo de subdivisões em um único intervalo. Se a malha atual contém ne pontos, o número de pontos adicionados $ne_1 \leq ne - 1$ para que o número total de pontos da nova malha seja no máximo de $2ne - 1$.

Adaptação por Métodos Simultâneos

Na abordagem simultânea, o problema de otimização é resolvido simultaneamente, com a adaptação da malha através de um *NLP* obtido por discretização total (Tanartkit e Biegler, 1997). Esta abordagem é baseada na adequação a dois critérios são eles: o erro de discretização adequado e da condição de otimalidade satisfeita na solução final. Nesta abordagem, acrescentam-se elementos finitos até que seja garantido que as aproximações dos perfis das variáveis de estado sejam adequadas, onde os erros de discretização sejam limitados e satisfazem o critério de precisão desejado. Além disso, as condições de otimalidade do problema de controle ótimo devem ser satisfeitas. Da teoria de controle ótimo, as condições de otimalidade são satisfeitas quando a função Hamiltoniana permanece constante ao longo de todo o horizonte de otimização. Este dois testes são efetuados somente depois que o problema de otimização dinâmica seja resolvido, para determinar o número de elementos.

Considere o seguinte problema de otimização dinâmica discretizado (*NLP*):

$$\begin{aligned}
&\min \varphi(z_{ne}, y_{ne,ncol}, u_{ne,ncol}, t_f) \\
&s.a. \\
&\frac{dz}{d\tau_{i,j}} = F(z_{i,j}, y_{i,j}, u_{i,j}, p)\alpha_i \quad i = 1, \ldots, ne-1;\ j = 1, \ldots, ncol \\
&G(z_{i,j}, y_{i,j}, u_{i,j}, p) = 0 \\
&S(z_{i,j}, y_{i,j}, u_{i,j}, p) \leq 0 \\
&T(z_{ne,ncol}, y_{ne,ncol}) \leq 0
\end{aligned} \tag{3.77}$$

$$z_0 = z(0)$$
$$z_i = z_{i-1} + h_i^0 \sum_{j=1}^{ncol} \Omega_j(1) \frac{dz}{d\tau_{i-1,j}} \qquad (3.77)$$

onde $\alpha_i = h_i / h_i^0$ é o comprimento do elemento *i* normalizado e $\tau = (t - t_{i-1})/\alpha_i$ o tempo normalizado, sendo $h_i = t_i - t_{i-1}$ o comprimento do elemento, h_i^0 a malha inicial, e $z_{i,j} = z(t_{i,j})$.

O erro do processo de discretização normalmente é avaliado pelos resíduos da aproximação polinomial, na forma:

$$R_{i,j} = F(z_{i,j}, y_{i,j}, u_{i,j}, p)\alpha_i - \frac{dz}{d\tau_{i,j}} \qquad (3.78)$$

De forma análoga da teoria de controle ótimo, o Hamiltoniano do problema discreto pode ser definido como:

$$H_{i,j} = \lambda_{i,j}^T F(z_{i,j}, y_{i,j}, u_{i,j}, p) + \mu_{i,j}^T G(z_{i,j}, y_{i,j}, u_{i,j}, p) + \nu_{i,j}^T S(z_{i,j}, y_{i,j}, u_{i,j}, p) + \upsilon^T T(z_{ne,ncol}, y_{ne,ncol}) \qquad (3.79)$$

onde λ, μ, ν e υ são os multiplicadores de Lagrange do *NLP* e equivalentes às variáveis adjuntas do *DAOP*. Na solução do *NLP*, os três últimos termos são *nulos*, pois as condições de viabilidade e de complementaridade têm que ser satisfeitas na solução ótima. Portanto, pode-se escrever (Equação 3.80):

$$H_{i,j} = \lambda_{i,j}^T F(z_{i,j}, y_{i,j}, u_{i,j}, p) \qquad (3.80)$$

No processo de adaptação de malhas, o erro de discretização deve ser restrito a ε_{tol}. Tanartkit e Biegler (1997) apresentaram tipos de restrições de erro, que podem ser usados no teste de precisão dos perfis de estado. Uma maneira de avaliação do erro de discretização é a realização de rodadas do otimizador com malhas diferentes. Na primeira rodada, o problema é resolvido com um dado tamanho da malha e numa segunda, com uma malha mais fina, muitas vezes a metade da original. Em seguida, o erro é calculado comparando as duas soluções. Essa técnica é mais precisa, mas também é computacionalmente mais caro. Outra maneira de avaliar os erros de discretização é feita através da avaliação dos resíduos da aproximação polinomial. Usualmente são impostas restrições nos resíduos das equações do sistema *DAE* nos pontos de não colocação t_{nonc} (limites residuais), ou seja:

$$\|R(t_{nonc})\| = \|F(z(t_{nonc}), y(t_{nonc}), u(t_{nonc}), p)\alpha_i - \dot{z}(t_{nonc})\| \le \varepsilon_{tol} \qquad (3.81)$$

Esta condição não é imposta diretamente ao *NLP*. Em vez disso, os resíduos das equações de estado, nos pontos de não-colocação ($t_{i,nonc}$) são avaliados de limitados para cada

elemento. Se essa restrição não for satisfeita para um determinado elemento, o mesmo pode ser dividido em dois e o problema resolvido novamente, usando a solução anterior como a estimativa inicial do novo problema.

O problema de adaptação e otimização dinâmica é considerado resolvido quando as condições de otimalidade (Hamiltoniano) são satisfeitas. Considerando o uso da função Hamiltoniano como descrito anteriormente, pode-se considerar que o Hamiltoniano deva permanecer constante ao longo de todo o horizonte de otimização, para que a solução de um problema de controle ótimo seja uma solução ótima, ou seja:

$$\forall_{i,j}: \quad H_{i,j} = CTE \tag{3.82}$$

Portanto, se esta função não for constante para todos os elementos finitos e pontos de colocação, então a solução não é ótima, e a malha deve ser refinada, adicionando novos elementos. Tanartkit e Biegler (1997) propuseram a verificação desta condição na seguinte forma:

$$\left\| \frac{dH\left(t_{i,j}\right)}{dt} \right\| \leq \eta_{tol} \tag{3.83}$$

onde η_{tol} é a tolerância desejada definida como ($max_{i,j}[0{,}001;\ 0{,}001\|H_{i,j}\|]$). A malha deve ser refinada dividindo o elemento em duas partes se o critério for violado.

A primeira abordagem proposta consistiu na solução do problema de otimização e de refinamento da malha em uma única etapa (Vasantharajan & Biegler, 1990). Neste caso, os comprimentos dos elementos são considerados como variáveis de decisão, sendo impostas restrições ao erro de discretização para cada elemento finito. Estas restrições garantem a precisão da discretização. Porém, este método resulta em um problema *NLP* de maiores dimensões e de comportamento mais não linear do que a solução do *NLP* com os elementos fixos. Outro problema desta estratégia é a necessidade de se inicializar adequadamente o problema para se obter uma solução bem sucedida, pois caso contrário, causam inconsistências nas restrições de erros. A formulação geral desta abordagem é dada pela Equação 3.84, onde são impostas as restrições nos comprimentos dos elementos finitos $\Sigma h_i = 1$, $h_L \leq h_i \leq h_U$ e $h_i \left\| F\left(\dot{x}_i^*, x_i^*, u_i^*, p, h_i\right) \right\| \leq \varepsilon_{tol}$.

$$\begin{aligned}
&\min_{u_{i,j}} \phi\left(x(1)\right) \\
&s.a. \\
&\quad F\left(\dot{x}_{i,j}, x_{i,j}, u_{i,j}, p, h_i\right) = 0, \qquad x_{1,0} = x_0 \\
&\quad x_{i+1,0} = x_{i,0} + h_i \sum_{j=1}^{ncol} \dot{x}_{i,j} \Omega_j(1) \quad i = 1,\ldots,ne-1;\ j = 1,\ldots,ncol \\
&\quad S\left(x_{i,j}, u_{i,j}, p\right) \leq 0 \qquad\qquad i = 1,\ldots,ne;\ j = 1,\ldots,ncol \\
&\quad T\left(x_{ne,ncol}\right) \leq 0
\end{aligned} \tag{3.84}$$

$$\sum_{i=1}^{ne} h_i = 1$$
$$h_i \left\| F\left(\dot{x}_i^*, x_i^*, u_i^*, p, h_i\right) \right\| \le \varepsilon_{tol} \qquad i = 1, \ldots, ne$$
$$h_L \le h_i \le h_U$$

onde x^*, u^* são os perfis ótimos obtidos no problema interno para um vetor h definido pela otimização.

Em função dos problemas do método de uma camada de otimização, Tanartkit e Biegler (1997) propuseram uma estratégia de duas camadas em que a solução de um problema externo determina os comprimentos de elemento. E na segunda camada, tem-se a solução de um problema interno (malha fixa) de otimização dinâmica, onde se determinam os perfis ótimos das variáveis de controle e estado. A Figura 3.28 mostra uma representação esquemática desta abordagem.

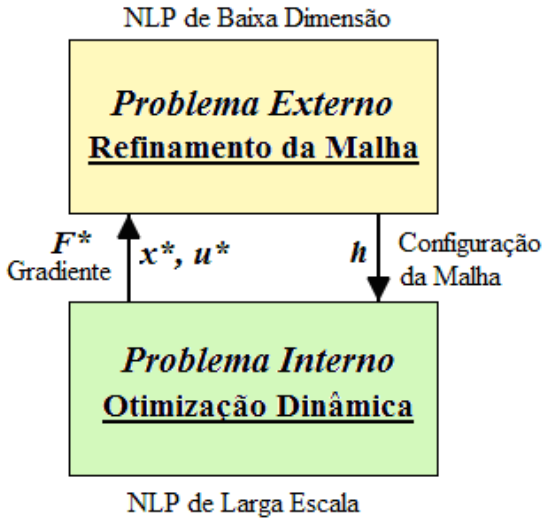


Figura 3.28 – Esquema de otimização dinâmica em dois níveis (Tanartkit, 1996).

Na formulação geral desta abordagem, resolve-se um problema de otimização dinâmica convencional na camada interna do problema, cuja formulação do problema interno pode ser escrita como:

$$\min_{u_{i,j}} \phi\left(x(1)\right)$$
s.a.
$$\begin{aligned} & F\left(\dot{x}_{i,j}, x_{i,j}, u_{i,j}, p, h_i\right) = 0, \qquad x_{1,0} = x_0 \\ & x_{i+1,0} = x_{i,0} + h_i \sum_{j=1}^{ncol} \dot{x}_{i,j} \Omega_j(1) \quad i = 1, \ldots, ne-1;\ j = 1, \ldots, ncol \\ & S\left(x_{i,j}, u_{i,j}, p\right) \le 0 \qquad i = 1, \ldots, ne;\ j = 1, \ldots, ncol \\ & T\left(x_{ne,ncol}\right) \le 0 \end{aligned} \tag{3.85}$$

Já, na camada externa, tem-se um problema de refinamento da malha discreta, onde são impostas as restrições nos comprimentos dos elementos finitos $\Sigma h_i = 1$, $h_L \leq h_i \leq h_U$ e $h_i \left\| F\left(\dot{x}_i^*, x_i^*, u_i^*, p, h_i\right) \right\| \leq \varepsilon_{tol}$, e pode ser escrito como:

$$\begin{aligned} &\min_{h_j} \phi\left(x^*\right) \\ &s.a. \\ &\quad \sum_{i=1}^{ne} h_i = 1 \\ &\quad h_i \left\| F\left(\dot{x}_i^*, x_i^*, u_i^*, p, h_i\right) \right\| \leq \varepsilon_{tol} \qquad i = 1, \ldots, ne \\ &\quad h_L \leq h_i \leq h_U \end{aligned} \tag{3.86}$$

onde x^*, u^* são os perfis ótimos obtidos no problema interno para um vetor h definido pela otimização do problema externo.

Com essa abordagem, evita-se a solução de um problema não-linear não suave porque os conjuntos de restrições ativas obtidos da solução do problema interno podem mudar de uma malha para outra. Isto acontece devido ao fato do processo de otimização da malha discreta na camada externa do algoritmo ser incapaz de detectar os pontos de chaveamento da estrutura da solução do controle ótimo (obtida na camada interna) fazendo com que o processo de busca de conjuntos ativos constantes (as variáveis ativas não se alteram ao longo do elemento) seja exaustivo.

Biegler, Cervantes e Wächter (2002), desenvolveram uma nova abordagem que permite superar estes problemas sem suavidades, incorporando o refinamento de malha no algoritmo de solução de problemas de *NLP* de ponto interior, como visto na Figura 3.29. Note que o algoritmo de *NLP* de ponto interior original tem como um ponto importante, a solução de um problema de barreira para encontrar a direção de busca no processo de otimização. Este algoritmo foi alterado neste ponto para poder lidar com a redução do parâmetro de barreira (μ) e o erro de discretização ao mesmo tempo.

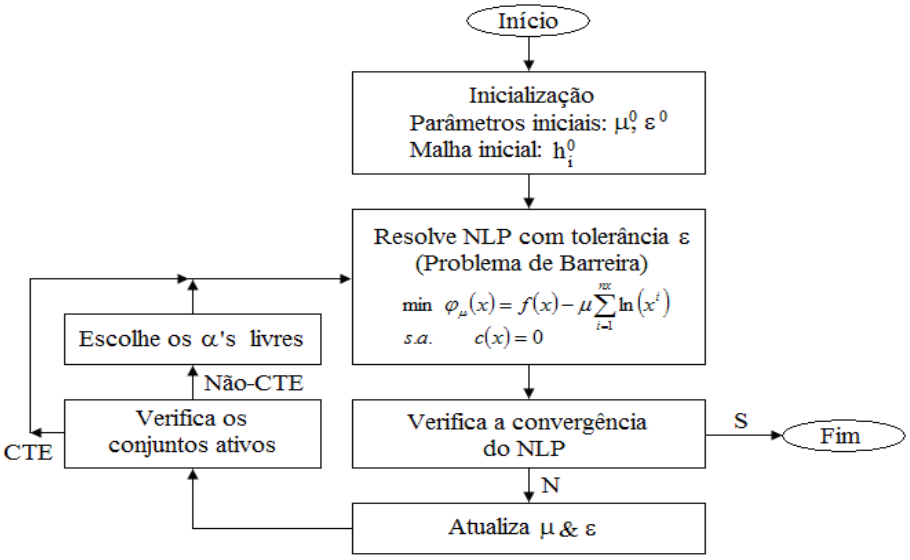


Figura 3.29 – Esquema do algoritmo de elemento finito móvel.

A idéia principal do algoritmo de refinamento de malha é detectar restrições ativas durante o processo de solução e obter uma resposta onde o conjunto de restrições ativas não altera em ao longo de qualquer elemento finito. Com um número suficiente de elementos finitos para garantir a precisão dos perfis de estado, é possível obter um conjunto ativo constante em cada um dos elementos finitos. E com isso, a determinação dos perfis de controle ótimo é precisa.

Como pode ser visto na Figura 3.29, pode-se resolver o problema de barreira para um determinado μ e uma dada tolerância ε, proporcional a μ. Nestas soluções intermediárias do problema de barreira, verificam-se as alterações no conjunto de restrições de ativas dentro de cada elemento. Se houver alterações nos conjuntos ativos entre dois elementos consecutivos, o comprimento do elemento α_i se torna uma nova variável livre do problema da barreira. A partir daí, o algoritmo de ponto interior retoma o seu curso normal, onde μ e $\varepsilon \propto \mu$ diminuem de acordo com o algoritmo de barreira (Cervantes et al., 2000). Este processo é repetido até que o algoritmo atinja seu ponto de convergência.

Após a convergência, é necessário verificar se número de elementos finitos usados no processo de otimização foi suficiente e se a solução do problema de controle ótimo foi obtida. No primeiro teste, verifica-se a satisfação dos critérios de restrições nos erros de aproximação para os perfis de estado. Num segundo teste, é verificado se a função Hamiltoniana permanece constante ao de todo o horizonte de otimização.

Adaptação Multi-escalas

Uma abordagem diferente é adotada por Binder et al. (1997), onde utiliza a técnica de *wavelet* (obs.: no espaço *wavelet*, o comportamento do processo é separado em suas freqüências características, simplificando o sinal mantendo somente aquelas freqüências mais significativas - Vide Apêndice A) para adaptar a malha através da redistribuição dos nós e do refinamento da malha. A adaptação de malhas é feito através de um ciclo que envolve a otimização do processo, análise do sinal, proposta de nova malha usando *wavelets*, reconstrução e re-otimização do processo, como pode ser visto na Figura 3.30. O problema de otimização é resolvido repetidamente com diferentes malhas discretizadas, melhorando a qualidade da sua solução. O refinamento é baseado nas informações das soluções anteriores, como as condições iniciais da solução anterior.

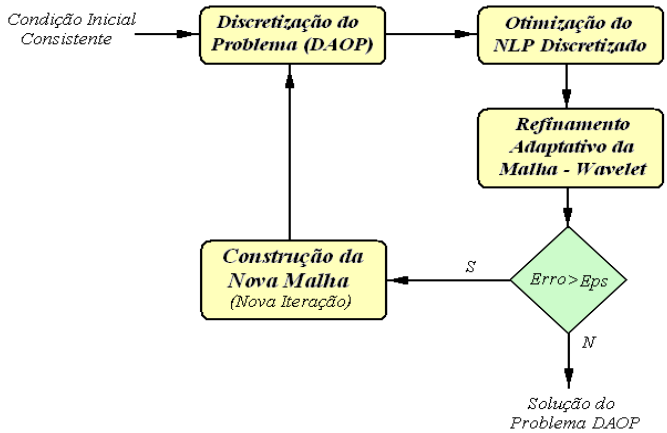


Figura 3.30 – Estratégia de refinamento adaptativo por *wavelet*.

Os conceitos de *wavelets* estão explicados no apêndice A, e são utilizados para a análise a posteriori da solução ótima u* encontrado na etapa de otimização. Nesta análise, uma nova malha de discretização é proposta, sendo utilizada na re-otimização subseqüente. Esta análise envolve duas etapas, uma de eliminação e outra inserção de pontos da malha discreta. Este processo de adaptação é baseado em representações de *wavelet* dos perfis de controle, convertido a partir da sua representação *spline*.

Para executar a adaptação de malhas discretas das variáveis de controle, inicialmente é necessário representar as ações de controle na sua forma de uma função discreta de simples escala de ordem $m = 1$ (perfis em degrau). Podem-se apresentar as parametrizações do controle, de acordo com a Equação 3.87, na seguinte maneira:

$$u(t) = \sum_{j=1}^{n} \hat{u}_{j,k} \varphi_{j,k}(t) \qquad \text{ou} \qquad u(t) = \hat{u}_k^T \varphi_k(t) \tag{3.87}$$

onde os valores de $\hat{u}_k$ são os parâmetros de controle obtidos pelo otimizador para determinada malha discreta k.

Em seguida, deve-se escolher um nível de resolução multi-escalas (escala J) e interpolar as ações de controle para obter os valores de $\hat{u}_J$ para os pontos discretos propostos. Com isso, convertem-se os perfis de controle para a representação, na resolução proposta, em simples escala nos pontos $k = 1,..., 2^J$. A transformação é feita utilizando $\hat{u}_{j,k}$ da seguinte forma, para encontrar os seus coeficientes na resolução J (Equação 3.88).

$$c_{J,k} = 2^{-J/2} \hat{u}_{J,k} \tag{3.88}$$

Com a representação das ações de controle em simples escala concluída, realiza-se a transformação destes perfis para o domínio *wavelet* através do operador T_J, onde $d_{\Lambda_J} = T_J c_{I_J}$, sendo $\Lambda_J := \{(j,k) | j_0 - 1 \le j \le J-1, k \in I_j\}$ o conjunto de índice I_J. Desta forma, os perfis de controle podem ser representados no domínio *wavelets* como $u(t) = d_{\Lambda_J}^T \Psi_{\Lambda_J}$. Com isso, os perfis de controle apresentados desta forma podem ser analisados e proposta uma adaptação para a malha, conforme recomendam Binder (2002) e Schlegel (2005).

O **refinamento de malha** é um processo que executa a análise dos perfis de controle e consiste de duas partes. Em primeiro lugar é realizada a eliminação de pontos da malha discreta e num segundo momento é efetuada uma etapa de inserção de pontos nesta malha, assegurando que nenhum coeficiente *wavelet* seja adicionado e que já tenha sido previamente removido na etapa de eliminação.

Uma propriedade importante da representação em função *wavelet* é o fato da mesma ser construída de tal forma que a norma euclidiana dos seus coeficientes *wavelet* ($d_{j,k}$) seja equivalente à norma L_2 da função original. Assim, descartando coeficientes pequeno $d_{j,k}$ fará com que ocorram apenas pequenas mudanças nas representações aproximadas da

função (no caso, perfis de controle). Desta forma, os perfis de controle podem ser aproximados por combinações lineares de poucas funções base *wavelet* apenas com os coeficientes $d_{j,k}$ significativos.

A **eliminação de pontos da malha** discreta tem o objetivo de remover pontos desnecessários dos perfis de controle ótimo $\hat{u}_i^{*l}$ obtido da solução do otimizador na etapa de refinamento *l*. O objetivo é obter uma representação mínima de u_i^l que satisfaça uma determinada tolerância ε_e'. Neste caso, o processo de eliminação termina quando:

$$\frac{\left\| u_i^l - \hat{u}_i^{*l} \right\|_2}{\left\| \hat{u}_i^{*l} \right\|_2} \leq \varepsilon_e' \tag{3.89}$$

Os coeficientes de pequena magnitude da transformada *wavelet* $d_{\Lambda_J}^T$ podem ser descartados (vide Figura 3.31 (b)), pois tem uma contribuição insignificante no sinal do perfil de controle. Pode-se definir um valor limítrofe ε_e dos coeficientes *wavelets*, onde os mesmos passam a ser significativos. Desta forma, se $d_{\Lambda_i}^{*l} \leq \varepsilon_e$, então o ponto discreto da malha correspondente a este coeficiente pode ser eliminado sem alterar significativamente o perfil original. A tolerância de eliminação ε_e deve ser definido como um número pequeno e pode ser definido, por exemplo, como a tolerância de integração ε_I do modelo do processo.

A **inserção de pontos da malha** procura obter uma aproximação melhor de $\hat{u}_i^{*l}$. O problema consiste na identificação das regiões dos perfis de controle que exigem uma resolução mais alta. A análise de *wavelet* da solução ótima pode ser usada para determinar as regiões do perfil de controle, onde um refinamento é recomendável. Pode-se observar que os coeficientes *wavelet* de grandes magnitudes são significativos para uma representação precisa do perfil de controle. Os pontos vizinhos a estes elementos com coeficientes de grandes magnitudes são bons candidatos para refinamento da malha, e devem ser marcados. Desta maneira, as funções *wavelet* $\psi_{j,k-1}$ e $\psi_{j,k+1}$, bem como as funções de maior escala, $\psi_{j+1,2k}$ e $\psi_{j+1,2k+1}$, são definidas como vizinhas de ψ_j de função *wavelet*, k.

Os pontos que podem se refinados são aqueles que estão nas fronteiras para serem selecionados para inserção, como pode ser visto na Figura 3.31 (a). Considerando $\breve{d}_{\Lambda_i}^{*l}$ os coeficientes *wavelets* limítrofes e $\bar{d}_{\Lambda_i}^{*l}$ os coeficientes *wavelets* limites selecionados para inserção, devem-se adicionar detalhes na malha discreta quando $\left\| \bar{d}_{\Lambda_i}^{*l} \right\|_2 \geq \varepsilon_i \left\| \breve{d}_{\Lambda_i}^{*l} \right\|_2$. Isto é, são escolhido para refinamento aquelas funções *wavelets* candidatas cujo valor seja superior a um porcentual ε_i da norma L_2 dos coeficientes *wavelets* selecionados na vizinhança. Sendo ε_i a tolerância de refinamento. Um bom valor é 0.9, que implica que as *wavelets* limite que representam 90% da norma dos coeficientes limítrofes devem ser selecionados para refinamento (Schlegel et al., 2005).

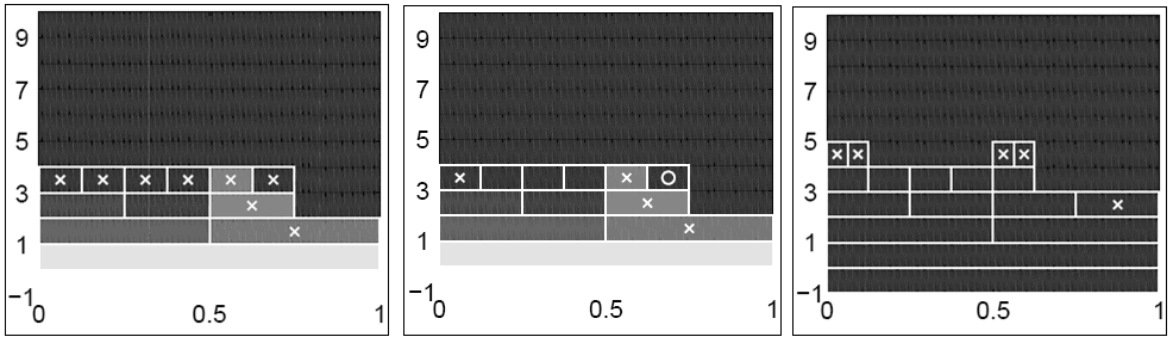


Figura 3.31 – Representação das escalas em *wavelets* e análise da malha. (a) Representação das escalas em *wavelets* limites, $\breve{\Psi}_{\hat{\Lambda}}$ (marcado com x); (b) Análise da malha, *wavelets* limites selecionadas para inserção, $\overline{\Psi}_{\hat{\Lambda}}$ (x) a para deleção $\tilde{\Psi}_{\hat{\Lambda}}$ (o); (c) malha refinada com *wavelets* adicionados (x). (Schlegel, 2005)

Como o processo de adaptação de malhas passa por uma análise do sinal de controle, eliminação e inserção de pontos da grade e re-otimização do processo, uma forma de definir quando a adaptação deve ser concluída é através do critério de desvio relativo da função objetivo do otimizador referente à etapa de adaptação (Ω^l) em relação ao ótimo verdadeiro (Ω^*), que pode ser escrito como $\delta_l = \left|\Omega^* - \Omega^l\right| \Big/ \left|\Omega^*\right|$. Como o ótimo verdadeiro não é conhecido, então uma alternativa é a substituição de δ_l por uma variação relativa na função objetivo em duas etapas de otimizações sucessivas. Schlegel (2005) então propôs um critério de parada heurística com base na verificação da melhoria relativa da função objetivo em cada etapa de refinamento. Neste caso, o processo de adaptação será encerrado quando (Equação 3.90):

$$\left|\frac{\Omega^l - \Omega^{l-1}}{\Omega^l}\right| \leq \varepsilon_\Omega \tag{3.90}$$

onde ε_Ω é uma tolerância especificada pelo usuário, em seguida, o algoritmo pára. A tolerância de parada ε_Ω pode ser definida como a tolerância de otimização ε_O.

Resumidamente, pode-se esquematizar o processo de adaptação de malhas da seguinte forma:

- Inicializar os parâmetros de adaptação
- Fazer enquanto $\left|\frac{\Omega^l - \Omega^{l-1}}{\Omega^l}\right| > \varepsilon_\Omega$ ou $l < l_{max}$
  - Discretizar os perfis de controle por *B-spline*: $u(t) = \hat{u}^T \Phi(t)$
  - Resolver o problema de otimização dinâmica
  - Conversão para o domínio *wavelets*: u^*, $\Delta_u^l \rightarrow d_\Lambda^{*l}$, Λ^l
  - Adaptar a malha discreta: Δ_u^{l+1}
    - Eliminação de pontos da malha

- Inserção de pontos da malha
  - Interpolar os perfis de controle para a nova malha: $u^{l+1} = \text{interp}(u^l, \Delta_u^{l+1})$
- Estabelecer a solução ótima com adaptação: $u^* = u^l$, $\Omega^* = \Omega^l$

3.4.3 Detecção da estrutura da solução do *DAOP*

Ao resolver o problema de otimização dinâmica, o algoritmo deve respeitar as condições necessárias de otimalidade (*NCO*) conforme a seção "*Métodos indiretos - Princípio do Máximo de Pontryagin*" do tópico 3.1.2.1 - "*Solução do problema de otimização dinâmica*" deste capítulo (Srinivasan et al., 2003). Quando se utilizam os métodos diretos, os multiplicadores de Lagrange do *NLP* fornecem informações sobre a estrutura da solução ótima. E cada multiplicador está associado a uma determinada restrição (de x_i ou u_i) em um determinado instante de tempo discreto (t_j).

Nas condições ótimas, todas as restrições de estado devem satisfazer as condições de complementaridade do problema ($\mu(t_j)S(t_j) = 0$), e o multiplicador de Lagrange μ_i indica se a restrição está ativa ou não. Se $\mu_i(t_j) = 0$, então a restrição i não está ativa no tempo t_j; Se $\mu_i(t_j) > 0$, significa que restrição está ativa no seu limite superior; Se $\mu_i(t_j) < 0$, então a restrição está ativa no seu limite inferior. Da mesma forma, para as variáveis de controle, as condições de complementaridade ($\eta(t_j)S_u(t_j) = 0$) devem ser satisfeitas, e as interpretações dos multiplicadores são as mesmas apresentadas para as variáveis de estado.

Quando o otimizador encontra uma solução bem sucedida, a cada instante de tempo (t_j), cada ação de controle $u_i(t_j)$ busca as condições de otimalidade. Existem 4 situações possíveis para os perfis de controle, são elas: $u_i(t_j) = u_{i,min}$, onde $u_i(t_j)$ está no limite inferior (representado pelo tipo u_{min}); $u_i(t_j) = u_{i,max}$, onde $u_i(t_j)$ está no limite superior (representado pelo tipo u_{max}); $u_{i,min} < u_i(t_j) < u_{i,max}$, onde $u_i(t_j)$ está livre e é determinando pelas restrições de caminho (representado pelo tipo u_{state}) ou está buscando a otimalidade ($H_{u_i}(t_j) = 0$), que se caracteriza pela sensibilidade nula do Hamiltoniano em relação ao controle (representado pelo tipo u_{sing}). Neste último caso, as variáveis de estado e de controle então dentro da região viável do problema. Resumindo, as variáveis do problema devem estar numa das seguintes condições: variável de controle ou de estado ativa, ou dentro da região viável com transição ótima (Vide Figura 3.32). Cada trecho do perfil de define um tipo de arco de controle e os instantes onde ocorrem as mudanças de tipo de arcos são chamados de instantes de chaveamento de estrutura.

A detecção automática da estrutura foi proposta por Schlegel e Marquardt (2004 e 2006) e consiste em analisar as *NCO* que caracterizam a estrutura da solução ótima. Depois de encontrar os tipos de arcos e os pontos de chaveamento, reformula-se o problema de otimização (neste caso, como multi-estágios) e resolve o novo problema de otimização dinâmica reformulado. Cada ponto de chaveamento define um estágio no novo problema de otimização, onde não se sabe previamente o momento correto deste chaveamento. Este momento é definido pelo tempo final de cada estágio, que passa a ser livre, se tornando uma nova variável de decisão no problema reformulado.

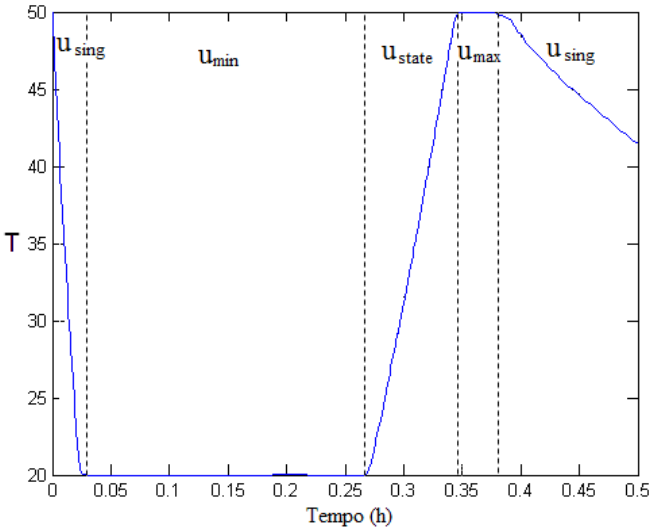
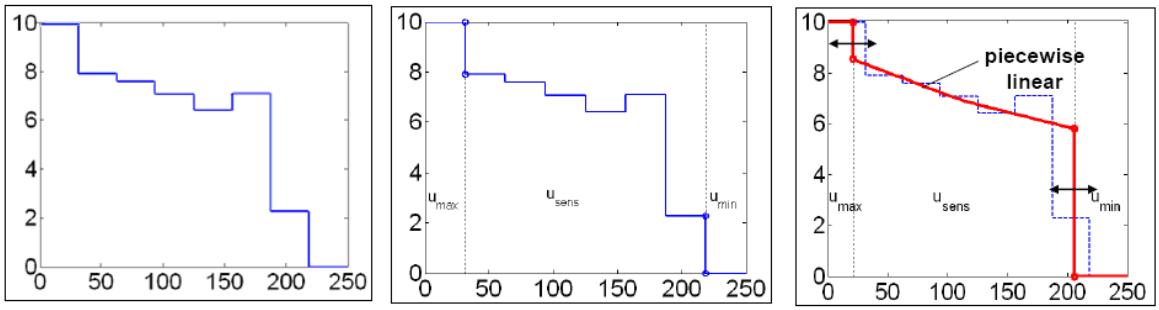


Figura 3.32 – tipos de arcos de controle.

As variáveis de controle definidas como u_{min} e u_{max} deixarão de serem variáveis livres nos respectivos estágios do novo problema de otimização dinâmica e a solução deste novo problema deverá ter uma qualidade melhor do que a anterior. Desta forma, o processo de detecção de estrutura pode ser descrito como:

- Resolver o *DAOP* de simples ou multi-estágios;
- Analisar os resultados do *NLP* e detectar os arcos de controle (Bryson e Ho, 1975; Srinivasan et al., 2003);
- Reformular o problema multi-estágios de acordo com o chaveamento de estrutura encontrado;
- Resolver novamente o *DAOP*.

A Figura 3.33 mostra resumidamente este processo. Na Figura 3.33 (a), tem-se a primeira solução ótima obtida; em (b), a detecção da estrutura de controle ótimo; e em (c), a solução refinada.



(a) (b) (c)

Figura 3.33 – Processo de detecção de estrutura.
(a) solução original; (b) detecção de estrutura; (c) re-otimização (refinamento da solução)
(Fonte: Schlegel e Marquardt, 2004).

Este procedimento pode ser repetido até encontrar uma solução satisfatória. È bom relembrar que ao reformular o problema com uma estrutura de controle definida, os tempos de duração dos estágios se tornam variáveis livres (t_f^k) e, portanto passam a ser variáveis de decisão adicionais do novo problema de otimização. Cabe também lembrar que este procedimento consome mais tempo computacional, pois repete todo o processo de otimização por um número de vezes pré-definido. A decisão de realizar esta tarefa passa pela avaliação da necessidade de refinar a solução, em contrapartida de um maior tempo de CPU gasto para obter esta solução refinada.

Ao analisar a estrutura da solução ótima, devem-se decompor os arcos em seus intervalos de tempos discretos pequenos, analisar cada trecho da solução ótima $(t_{j-1},t_j]$ individualmente, caracterizar cada tipo de arco (baseado nos multiplicadores de Lagrange do *NLP*), e agregar os arcos de controle adjacentes de mesmo tipo, como mostra a Figura 3.34.

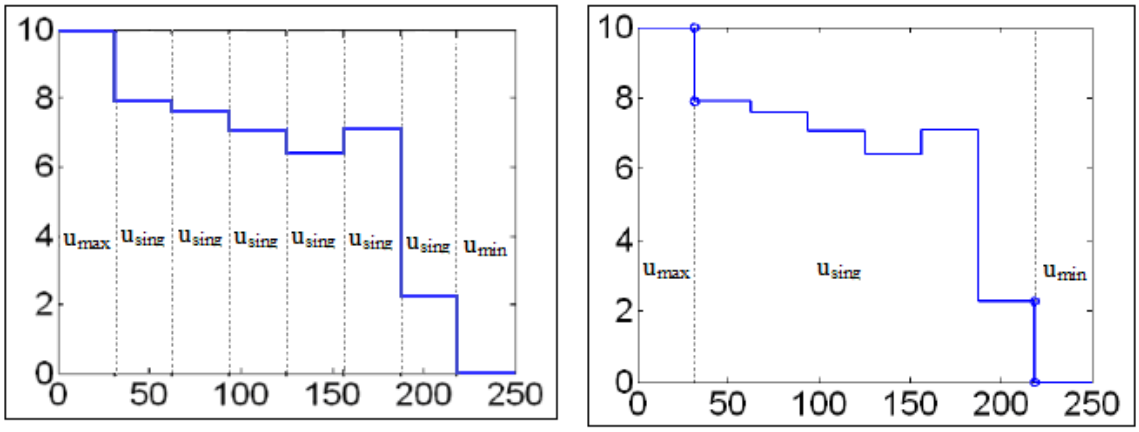


Figura 3.34 – Agregação dos arcos de controle

No processo de detecção da estrutura, devem-se identificar primeiramente as restrições das variáveis de estado que estão ativas. Analisando os multiplicadores de Lagrange do *NLP* para as variáveis de estado em cada intervalo de tempo discreto (t_j), pode-se identificar quando a restrição está *ATIVA* ou *INATIVA*. Quando $\mu_i(t_j) \neq 0$, a restrição de estado está *ATIVA*. Desta forma, a restrição é considerada ativa no intervalo de tempo, se a mesma atinge a restrição de pelo menos uma vez no intervalo $t_{j-1} < t \leq t_j$.

Para as variáveis de controle, a análise é mais complicada. Primeiramente, verificam-se os sinais dos multiplicadores de Lagrange $\eta(t_j)$. Se $\eta_k(t_j) < 0$, significa que a variável de controle k está no seu limite mínimo no tempo t_j, portanto é classificada como $U_{k,j} = MIN$. Da mesma forma, se $\eta_k(t_j) > 0$, então $U_{k,j} = MAX$. Por outro lado, se $\eta_k(t_j) = 0$, então o controle pode ser $U_{k,j} = STATE$ (busca a restrição de estado) ou $U_{k,j} = SING$ (controle singular).

A distinção entre arcos *STATE* e *SING*, depende do número de variáveis de controle inativas $n_U(t_j)$ e do número restrições de caminho de estado ativo $n_X(t_j)$ no intervalo j. Os seguintes casos podem ser identificados quando a variável de controle $u_k(t_j)$ está inativa:

- Se $n_X(t_j) = 0$, então $U_{k,j} = SING$. Significa que não há restrições de estado ativas, e as variáveis de controle estão buscando a otimalidade.
- Se $n_X(t_j) = n_{\bar{U}}(t_j)$, então $U_{k,j} = STATE$. Tem-se um emparelhamento das variáveis de estado com as de controle, ou seja, todos os perfis das variáveis de controle inativas são determinados pelas restrições de caminho de estado.
- Se $n_X(t_j) < n_{\bar{U}}(t_j)$, então $U_{k,j} = STATE$ para as $n_X(t_j)$ variáveis de controle mais sensíveis com os estados ativos. Neste caso tem-se novamente um emparelhamento das variáveis. Por outro lado, $U_{k,j} = SING$ para as $n_{\bar{U}}(t_j) - n_X(t_j)$ variáveis de controle inativas restantes. Estas variáveis de controle estão buscando a otimalidade, ou seja, os perfis das variáveis de controle inativas são determinados pelo controle singular.
- Se $n_X(t_j) > n_{\bar{U}}(t_j)$, um problema de detecção de estrutura, pois não pode haver mais restrições de caminho de estado ativas do que variáveis de controle inativo.

A questão de emparelhamento das variáveis de estado e controle foi abordada por vários pesquisadores, podem-se destacar os trabalhos de Vassiliadis et al. (1994) e de Feehery e Barton (1998 e 1999). Eles mostraram que, em problemas com restrições de caminho de estado de igualdade, nem todas as variáveis de controle são verdadeiramente independentes. Um determinado subconjunto de variáveis de controle é determinado diretamente pela solução de um subsistema de *DAE's* quadrado, ou seja, definida pelo emparelhamento de variáveis. Esta mesma técnica foi aplicada a problemas com restrições de caminho de desigualdade por Feehery e Barton (1998).

Quando os arcos de restrições de caminho se tornam ativos, estes arcos podem ser tratados como restrições de igualdade dos estados. Neste caso, na análise de emparelhamento, é necessário definir quais variáveis de controle perderão liberdade para o controle singular. Lembre que, para o otimizador, todas as variáveis de controle são tratadas como graus de liberdade, mesmo as que buscam as restrições de caminho de estado.

A escolha do emparelhamento é feita baseada nas sensibilidades das variáveis de estado ativas com as de controle inativas ($s_{i,k}(t_j) = \partial x_i / \partial \bar{u}_{k,j}$). Os emparelhamentos são feitos nas ordens decrescentes das sensibilidades (primeiro o par mais sensível) até que todas as restrições ativas tenham seus pares. Isso significa $n_X(t_j)$ emparelhamentos por intervalo j. As variáveis de controle restantes $n_u(t_j) - n_{\bar{U}}(t_j)$, serão graus de liberdade para realizar o controle singular (a otimização propriamente dita).

Note que, nem sempre há unicidade nas correspondências dos controles. Unicidade depende da estrutura do sistema *DAE* aumentado com as restrições de estado ativas. Em alguns casos, não é possível associar somente uma variável de controle a um estado. Muitas vezes as ações de controle são resultados de combinações dos controles para manter uma ou mais restrições ativas (Srinivasan et al., 2003).

Há situações onde a escolha do emparelhamento não é a mais adequada, mas esta escolha não afeta a rodada seguinte do otimizador. A única conseqüência pode ser a colocação de um estágio a mais ou a menos no problema de otimização dinâmica.

O algoritmo de detecção da estrutura de arcos da solução ótima tem duas partes, primeiro se realiza a análise dos intervalos de discretização e depois a agregação dos intervalos para

arcos. O algoritmo de análise dos intervalos de discretização pode ser resumido da seguinte forma:

- Escolha uma malha com *ne* elementos discretos;
- Faça análise da estrutura para cada intervalo $(t_{j-1}, t_j]$ da malha discreta; $(j = 1:ne)$
  - Seja $n_{\bar{U}}(t_j)$ o número de variáveis de controle livres e $n_X(t_j)$ o número de restrições de estado ativas no elemento finito *j*;
  - Inicializa o número de restrições ativas de estado $n_X(t_j)$ e de controle $n_U(t_j)$;
    - $n_X(t_j) = n_U(t_j) = 0$;
  - Verificar quais restrições de caminho de estado ($S_i(t_j)$) estão ativas, para um determinado intervalo de tempo $(i = 1:n_S)$
    - Se $\mu_i(t_j) \neq 0$,
      - então $X_{i,j} = ATIVA$;
      - senão $X_{i,j} = INATIVA$;
  - Verificar o tipo de arco para cada variável de controle ($u_k(t_j)$), para um determinado intervalo de tempo $(k = 1:n_u)$
    - Quais controles estão no limite mínimo ($\eta_k(t_j) < 0$);
      - $U_{k,j} = MIN$;
    - Quais controles estão no limite máximo ($\eta_k(t_j) > 0$);
      - $U_{k,j} = MAX$;
    - Se o número de restrições de estado ativas é menor do que o número de variáveis de controle livres ($n_X(t_j) \leq n_{\bar{U}}(t_j)$);
      - Define-se o emparelhamento dos controles;
      - Escolha dos controles que está buscando ativar as restrições
        - $U_{k,j} = STATE$; $\quad \forall k = 1 : n_X(t_j); k \notin U(t_j)$
      - Escolha dos controles que controles está buscando a otimalidade
        - $U_{k,j} = SING$; $\quad \forall k = 1 : (n_{\bar{U}}(t_j) - n_X(t_j)); k \notin U(t_j)$;
    - Quando há erro na definição da estrutura de arcos ($n_X(t_j) > n_{\bar{U}}(t_j)$);

Após esta análise, com as informações do tipo de ações de controle U_j (*MIN*, *MAX*, *STATE* ou *SING*) e das situações das restrições de caminho dos estados X_j (*INATIVA* ou *ATIVA*) a cada intervalo *j*, pode-se agregar os trechos de mesmas características e construir os arcos de controle, e assim definir a estrutura da solução ótima obtida. O algoritmo de agregação dos intervalos para arcos pode ser descrito da seguinte forma:

- Inicializa a estrutura de arcos de controles com a estrutura no instante inicial $A = U_1$;
- Analisar mudanças na estrutura dos arcos de controle ou restrições ativas para cada intervalo $(t_{j-1}, t_j]$ da malha discreta; $(j = 2:ne)$
  - Se a estrutura de arcos de controle no elemento *j* for diferente da estrutura em *j-1* (pelo menos uma variável de controle de U_j diferente) $U_j \neq U_{j-1}$; ***OU***
  - Se a estrutura de restrições ativas de estados no elemento *j* for diferente da estrutura em *j-1* (pelo menos uma variável de estado de X_j diferente) $X_j \neq X_{j-1}$;
    - Então acrescenta um estágio na estrutura de arcos de controle $A = [A\ X_j]$;

A detecção da estrutura de controle é baseada na distribuição da malha discreta da solução do *DAOP*. Usualmente, os pontos discretos não coincidem com os tempos de chaveamentos da solução ótima se fosse resolvido sem discretização. Com isso, um dos pontos mais importantes do método de detecção da estrutura e reformulação do *DAOP* multi-estágios consiste na determinação destes instantes de chaveamento. Isto pode ser feito com a reformulação de um *DAOP* multi-estágios com tempos finais livres em cada estágio. A implementação desta estratégia completa consiste no processo de formulação do *DAOP*, discretização da malha, adaptação da malha e detecção da estrutura da solução ótima. Com isso, o *DAOP* pode ser reformulado e repetido o mesmo processo de obtenção da solução. Isto permite o refinamento da solução ótima. Schlegel (2005) relata melhorias na função objetivo da ordem de 3% a 7%, porém com um custo computacional 7 vezes maior.

A implementação desta estratégia combinada de adaptação de malha e refinamento da solução pela detecção de estrutura da solução no *DyOS* está esquematizada na Figura 3.35. Neste esquema, *MX* corresponde ao número máximo de adaptações de malhas e *SD* é o número de refinamentos por detecção de estrutura de arcos. A definição destes parâmetros estabelece a aplicabilidade do algoritmo.

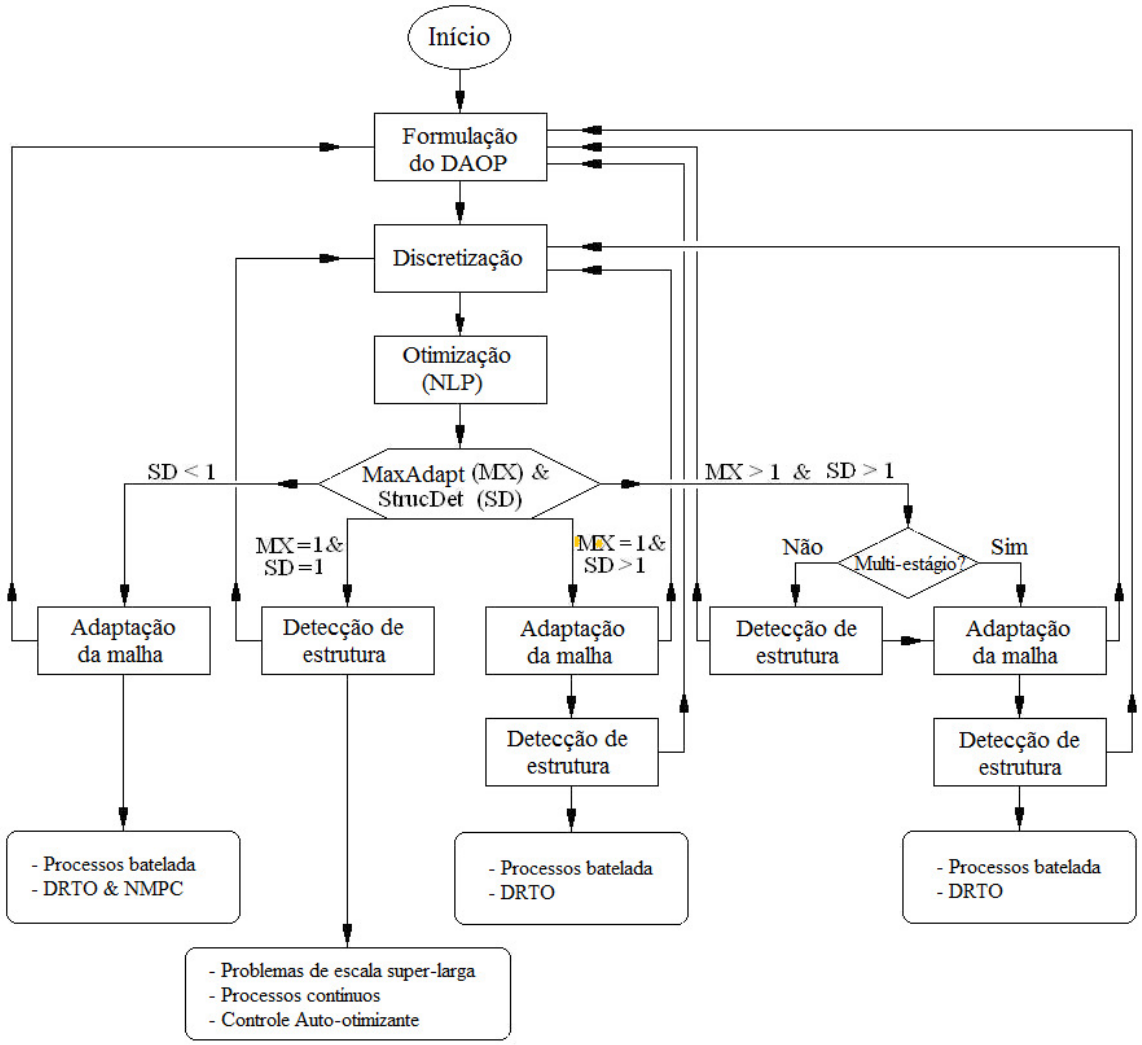


Figura 3.35 – Esquema de adaptação de malhas e detecção de estruturas no *DyOS*.

3.4.4 Mecanismo de disparo do otimizador

Durante a obtenção da solução de problemas de otimização dinâmica, pode-se levar muito tempo para encontrar a solução, principalmente quando se trata de problemas de larga escala e com modelos complexos. A otimização dinâmica de um conversor de *FCC*, por exemplo, pode levar de 35 a 60 min (num Intel Pentium 4) para obter a solução ótima (Almeida e Secchi, 2006a e 2011). Por esta razão, é conveniente o uso de um disparador do *DRTO*, pois não é recomendável executá-lo de forma cíclica com freqüência constante, mas sim sob demanda, ou seja, sempre que ocorrer um evento perturbador da solução (ex.: mudança na estrutura do *DAOP*, perturbação no processo, não cumprimento da receita ótima, dentre outros).

O uso do disparador é uma estratégia que foi proposta por Kadam et al. (2002) para decidir quando o otimizador dinâmico deve ser executado. Esta estratégia consiste de um sistema em dois níveis (Kadam et al., 2002), onde o *DRTO* é executado em um nível superior e um controle preditivo (*MPC*) que executa a recita ótima num nível inferior. Estas duas aplicações são executadas em escalas de tempos diferentes. O *DRTO* pode rodar em escalas de horas e o *MPC* em minutos. O controlador preditivo é normalmente linear e precisa de correções periódicas das predições dos estados e correções no modelo do processo de tempos em tempos. Usualmente, o *MPC* recebe trajetórias de referências ou valores ótimos para controle (y_{ref}, u_{ref}).

O disparador do *DRTO* é um dispositivo que analisa e valida a atualização da otimização em tempo real baseado na análise de sensibilidade. A sensibilidade paramétrica (Fiacco, 1983) é uma ferramenta útil para analisar perturbações na solução ótima. A aplicação desta técnica em otimização dinâmica também é conhecida como controle extremo da vizinhança (Bryson e Ho, 1975), e parte do pressuposto de que o conjunto de restrições ativas não se altera com o aparecimento de perturbações na planta. Porém, esta hipótese é relativamente forte, pois há vários casos em que isso não pode ser garantido. Isto faz com que esta técnica seja restrita às situações onde se tenha pequenas perturbações no processo.

Um método alternativo foi proposto por Kadam et al. (2002) e se baseia na atualização da sensibilidade na obtenção de uma solução rápida através da solução de um problema de *QP* (programação quadrática). Diferentemente do controle da vizinhança, este método permite a mudança do conjunto de restrições ativas do problema. Com esta solução, define-se um mecanismo de disparo de *DRTO* que decide o momento de se fazer a re-otimização do *DAOP*.

Ao atualizar o estado da planta, o estimador de estado encontra novos valores para os estados, controles e parâmetros. Ao obter a solução do *DAOP* por métodos diretos, passa-se por um processo de discretização que resulta num problema de *NLP* que pode ser representado na seguinte forma:

$$\begin{aligned} &\min_{z} f(z, p) \\ &s.a. \\ &\quad g(z, p) \geq 0 \end{aligned} \tag{3.91}$$

Ao resolver o *NLP*, é necessário satisfazer as condições de otimalidade (*NCO*). Considerando que a solução ótima nominal do *DRTO* seja z^*, λ^*, as *NCO* são dadas por:

$$\begin{aligned} &L\left(z^*, p, \lambda^*\right) = 0 \\ &\lambda_i^{*A} > 0 \text{ para a restrição ativa } g^A\left(z^*, p\right) = 0 \\ &\lambda_i^{*I} = 0 \text{ para a restrição inativa } g^I\left(z^*, p\right) > 0 \end{aligned} \tag{3.92}$$

sendo que o Lagrangeano pode ser escrito como:

$$L(z, p, \lambda) = f(z, p) - \lambda^T g(z, p) \tag{3.93}$$

A solução ótima pode ser obtida por diferenciação do *NCO* em relação à z e p, cujo sistema de equações algébricas a ser resolvido é escrito como:

$$\begin{bmatrix} L_{zz} & -\left(g_z^A\right)^T \\ g_z^A & 0 \end{bmatrix} \begin{bmatrix} z_p \\ \lambda_p^A \end{bmatrix} = -\begin{bmatrix} L_{zp} \\ g_p^A \end{bmatrix} \tag{3.94}$$

onde $z_p = \partial z/\partial p$, $\lambda_p = \partial \lambda /\partial p$, $L_{zz} = \partial^2 L/\partial z^2$ e $L_{zp} = \partial^2 L/\partial z \partial p$. Sendo as estimativas dos efeitos das perturbações em p dadas por $\Delta z = z_p(p)\Delta p$ e $\Delta \lambda^A = \lambda^A{}_p(p)\Delta p$.

Caso o conjunto de restrições ativa seja alterado com a perturbação Δp, o novo conjunto ativo passa a ser desconhecido, e a solução do *NCO* não pode mais ser resolvida da forma acima. Então a derivada do *NCO* pode ser reformulada como um problema *QP* (Kadam e Marquardt, 2004), que pode ser formulado como:

$$\begin{aligned} &\min_{\Delta z} \frac{1}{2}\left[\Delta z^T L_{zz} \Delta z + \Delta p^T L_{zp} \Delta z\right] + f_z{}^T \Delta z \\ &s.a. \\ &\quad g_z \Delta z \geq -g_p \Delta p - g \end{aligned} \tag{3.95}$$

onde $f_z = \partial f/\partial z$, $g_z = \partial g/\partial z$ e $g_p = \partial g/\partial p$. Lembre que todas as atualizações das funções são feitas com as informações da solução ótima z^*, λ^* e p. E portanto, a atualização rápida de z, λ e u(t), pode ser escrita como:

$$\begin{aligned} &z = z^* + \Delta z \\ &u(t) = u(t)^* + \Delta u(t) \\ &\lambda = \lambda^*\left(p + \Delta p\right) \end{aligned} \tag{3.96}$$

Com as trajetórias de controle e estado são atualizados através da integração do *DAE* com $p+\Delta p$. O erro na otimalidade de $z(p)$ é definido como:

$$\varepsilon_{opt} = \frac{\left\| L_z\left(z, p + \Delta p, \lambda\right)\right\|_\infty}{\left\|\lambda\right\|_2} \qquad \text{e} \qquad \varepsilon_{\inf} = \frac{\left\| g\left(z, p + \Delta p\right)\right\|_\infty}{\left\|z\right\|_2} \tag{3.97}$$

O erro da sensibilidade do Lagrangeano (ε_{opt}) qualifica a otimalidade da nova solução e o erro da inviabilidade da restrição (ε_{inf}) qualifica a viabilidade da solução perturbada. Lembre que quaisquer conjuntos de restrições ativas modificadas estão considerados nesta formulação.

Considerando as tolerâncias máximas de otimalidade como sendo τ_{opt} e da viabilidade como τ_{inf}, o disparador irá acionar uma nova otimização do *DRTO* quando $\varepsilon_{opt} > \tau_{opt}$ e $\varepsilon_{opt} > \tau_{inf}$.

A aplicação desta técnica na ferramenta de disparo do otimizador é efetuada primeiramente através da solução do *DAOP* no *DRTO*, obtendo z^* e λ^* e calculando L_{zz} e L_{zp}. Em seguida verifica a perturbação da solução ótima a cada intervalo de tempo Δt_k no nível do *MPC*. Em t_k, a ação de controle u_{k-1} é implementada e obtida a nova estimativa do estado, com os parâmetros atualizados. Neste momento as estimativas em t_{k-1} são avançadas no tempo ($t_k = t_{k-1}+\Delta t$) e o horizonte de tempo final (t_f) é reduzido da mesma quantidade ($t_{f,k} = t_{f,k-1}+\Delta t$). Desta forma, as trajetórias de referência também são deslocadas de *k-1* para *k*. Neste momento, são efetuadas as avaliações de $g(z,p)$ e as sensibilidades f_z e g_z e o estado em *k* são calculados após a integração do modelo. Com estas informações, a atualização rápida da solução ótima é efetuada resolvendo o *QP* correspondente, obtendo-se Δz, λ e o novo conjunto de restrições ativas. Com isso, atualiza-se $z = z^* + \Delta z$ e calcula-se o erro de otimalidade (ε_{opt}) e de inviabilidade (ε_{opt}). Se não passar no critério de otimalidade e viabilidade, então uma nova otimização deve ser feita pelo *DRTO*, caso contrário, deve avançar no tempo para *k+1* e repetir o processo (vide Figura 3.36).

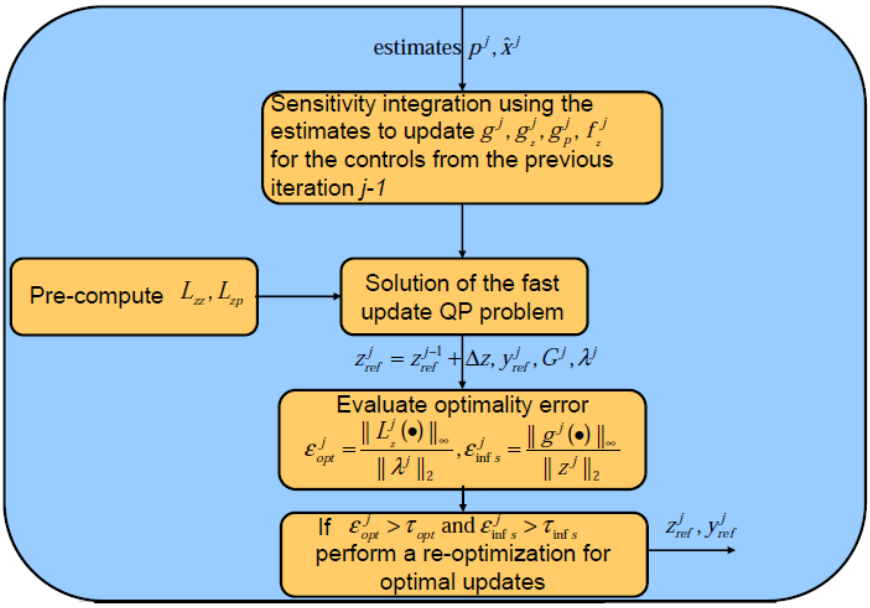


Figura 3.36 – Processo de disparo do *DRTO*.
(Fonte: slide de Kadam e Marquardt, 2004).

3.4.5 Solução do *NLP* do problema discretizado

Ao resolver um problema de otimização dinâmica pelos métodos diretos, o *DAOP* é transformado em um *NLP* (no processo de discretização). Este *NLP* resultante pode ser resolvido por um algoritmo do tipo *SQP* especializado. Os métodos de *SQP* (Nocedal e Wright, 2006), procuram resolver as condições de otimalidade de primeira ordem (condições de *KKT - Karush-Kuhn-Tucker*), gerando direções de busca do tipo Newton. Os métodos *SQP* se diferem pela forma de cálculo da Hessiana, método de escolha do comprimento do passo em busca de linha e da definição da direção de busca através da solução de (*QP*). Com isso, podem-se classificar os métodos *SQP* pelas seguintes categorias (Biegler e Grossmann, 2004): **conjuntos ativos versus métodos de barreira** - para resolver as restrições de desigualdades na geração das direções de busca; **aproximações quasi-Newton versus Hessiana exata** - cálculo da Hessiana para as etapas do tipo Newton; **busca linha (line search) versus métodos de região de confiança (trust region)** - para impor a convergência global das iterações *SQP*.

Estes métodos se mostraram eficientes para resolver os problemas de controle ótimo (Barclay et al., 1998), onde estes problemas têm as seguintes características: problemas de grandes dimensões (muitas variáveis e restrições); derivadas primeiras (Jacobianas) esparsas e estruturadas; muitos graus de liberdades e restrições ativas; e avaliações custosas de resíduos das restrições. Cabe ressaltar que os *NLP's* resultantes da discretização pelos métodos *single-shooting* e *multi-shooting* são bem menores do que aqueles resolvidos pelos métodos simultâneos. Porém, nos dois primeiros casos, é necessário primeiramente resolver o modelo do processo.

Duas classes de métodos se sobressaíram, foram os algoritmos *SQP's* tradicionais (como o *SNOPT*) e os algoritmos de pontos interiores (como o *IPOPT* - Vide Apêndice B). As suas características principais foram didaticamente apresentadas por Biegler e Grossmann (2004) e estão mostradas na Tabela 3.6.

Tabela 3.6 - Características básicas do algoritmo *SNOPT* e *IPOPT*.

Método	Tipo	Restrições de desigualdade	Globalização	Cálculo da Hessiana	Referência
SNOPT	*SQP*	Conj. Ativos	Line Search	Quasi-Newton	Gill et al., 1998
IPOPT	Ponto Interior	Barreira	Line Search	Exata ou Quasi-Newton	Wächter e Biegler, 2002

Na abordagem de solução das restrições de desigualdade através da identificação dos conjuntos ativos (*SQP's* tradicionais), as identificações das restrições ativas são efetuadas através da solução de um problema combinatorial. Para evitar este problema combinatorial, resolve-se um problema de barreira, onde o parâmetro de barreira μ tende a zero quando o mesmo se aproxima da solução ótima. Por isso, métodos de barreira (no *IPOPT*) podem exigir mais iterações para resolver o *NLP*, enquanto métodos conjuntos ativos (no *SNOPT*) requerem a solução de um subproblema de *QP*.

Na solução de *DAOP* pequenos (com modelos simples e poucas restrições de desigualdade ou conjunto bem conhecido de restrições ativas e poucas combinações - até cerca de 1000 variáveis), os métodos de conjuntos ativos (*SNOPT*) se tornam mais eficientes. Porém, se os problemas tiverem muitas restrições de desigualdade, os métodos de barreira (*IPOPT*)

se tornam mais eficientes. Ou seja, o *SNOPT* é muito sensível ao tamanho e complexidade do problema, e o *IPOPT* é um algoritmo menos sensível a estes fatores.

Com a popularização dos pacotes de simulação dinâmica e de algoritmos de diferenciação automática e simbólica, os cálculos das matrizes Jacobianas se tornaram bem precisos e eficientes. Isto estimulou o uso de Hessianas exatas na atualização das informações de segunda ordem no algoritmo de otimização. Esta avaliação é mais custosa do que as aproximações quasi-Newton (ex. *BFGS*), porém mais precisa e não se contaminam após algumas iterações. O *SNOPT* usa uma aproximação quasi-Newton da Hessiana, sendo que o *IPOPT* utiliza tanto a aproximação quasi-Newton como o cálculo exato da Hessiana.

Uma vez que a direção de busca (d^k) foi definida, na etapa de *QP* ou problema de barreira, define-se o tamanho do passo $\alpha \in [0, 1]$ e calcula-se o novo ponto $x^{k+1} = x^k + \alpha d^k$ que proporcione um progresso suficiente da função de mérito (minimize). Esta função é resultante da combinação das violações das restrições e da função objetivo. As opções comuns são o uso de estratégias de busca em linha convencional e os métodos de regiões de confiança, denominado a abordagem de filtro (Wächter & Biegler, 2006).

Blaszczyk *et al.* (2007) fizeram um estudo comparativo destes métodos. Segundo eles, uma vantagem dos métodos *SQP* sobre métodos de pontos interiores consiste em seu pequeno número de parâmetros de sintonia, frente à complexidade da sintonia dos algoritmos de pontos interiores (requer certa heurística). Por outro lado, métodos *SQP* têm dificuldades para resolver problemas degenerados, pois fica difícil identificar o conjunto de restrições ativas na obtenção da solução ótima. Os algoritmos de pontos interiores não são muito sensíveis à degeneração do problema de otimização. Eles apresentaram um estudo comparativo muito interessante destes algoritmos, baseado nos gráficos de *benchmark* de Dolan e Moré (2002).

Apesar de uma grande quantidade de estudos destes algoritmos, há ainda necessidade de investigação sobre comportamento, a eficácia e a robustez dos algoritmos *SQP* e de pontos interiores (Gould, 2003). Neste momento parece que, em termos de eficiência e robustez, algoritmos de pontos interiores são mais recomendáveis para os métodos simultâneos de solução de *DAOP* e os *SQP's* para os métodos seqüenciais. Mas, estes algoritmos ainda têm tido desenvolvimentos contínuos, frente à concorrência no interesse destes métodos. Esta situação ainda deverá perdurar pelos próximos anos.

3.5 Métodos de monitoração e diagnóstico de sistemas de controle e otimização

Ao desenvolver um sistema de *DRTO*, é importante se dispor de mecanismos de monitoração e diagnóstico, pois no surgimento de algum problema (falha ou desempenho ruim) o engenheiro deve investigá-lo e tomar ações corretivas. Para perceber os problemas ocorridos durante a operação do sistema, é necessário monitorar a planta e os resultados do otimizador. Ao detectar algum problema ou não conformidade no uso do sistema, o engenheiro deverá buscar informações de diagnóstico fornecidas pelo otimizador.

3.5.1 Iniciativas de diagnóstico e solução de problemas do otimizador

Ao investigar, na literatura aberta, alguns métodos e sistemas de monitoração e diagnóstico em otimização dinâmica, não foram encontrados relatos sobre este tema. Em *DRTO*, mais especificamente, não se tem conhecimento, na literatura aberta, da existência de tal ferramenta.

É comum se encontrar algoritmos de *LP* (programação linear) e *NLP* que geram relatórios de saídas, onde estes contêm resumos ou informações detalhadas das iterações dos algoritmos em questão. É importante destacar que estes relatórios apresentam indicadores e informações técnicas importantes sobre as iterações dos algoritmos de *NLP*. Porém, muitas vezes são de difícil entendimento e análise das causas básicas dos problemas, principalmente para engenheiros que não são especialistas no assunto. Especificamente, quando o *DAOP* é de larga escala (grande número de variáveis e restrições), a quantidade de informações geradas podem se tornar muito grande e consumir muito tempo para serem analisadas.

Ao se desenvolver um algoritmo de *NLP*, usualmente os pesquisadores se preocupam em relatar informações intermediarias e finais dos cálculos dos algoritmos em arquivos de saídas. Estes algoritmos apresentam resumos de cada iteração do otimizador, contendo os principais indicadores da evolução do sistema na busca na solução ótima.

As boas práticas no desenvolvimento de algoritmos de *NLP* levam em conta três aspectos principais destes métodos, são eles: a convergência, definição da direção de busca e do tamanho do passo no avanço para a solução. Há situações onde as taxas de convergências se tornam baixas causadas por vários fatores, podendo-se destacar a inviabilidade do problema, a singularidade, mau condicionamento, e distorções na estrutura da matriz Hessiana do Lagrangeano da função objetivo. Isto implica em uma curvatura inadequada da Hessiana, que pode até violar as condições necessárias de segunda ordem.

Outra condição de falha comum nestes algoritmos é um decréscimo insuficiente na função de mérito do otimizador. Isto ocorre pela dificuldade em conciliar a violação das restrições com a melhoria do valor da função objetivo. Este problema pode ocorrer por erros de modelagem, incompatibilidades das restrições, escalonamento inadequados das variáveis, dentre outros. Além disso, não-linearidades no modelo podem gerar dificuldades na obtenção da direção de busca adequada e podem causar falhas na identificação do conjunto de restrições ativas nos algoritmos que utilizam esta abordagem. Outro problema relevante, na falha do otimizador, é a escolha inadequada dos parâmetros de sintonia do otimizador, que podem estar muito relaxados ou rigorosos demais. Atualmente, o ajuste destes parâmetros de sintonia é muito dependente da experiência do usuário do otimizador.

No processo de diagnóstico dos *NLP's*, há também a possibilidade de se efetuar uma análise dos multiplicadores de Lagrange dos algoritmos, bem como uma análise de sensibilidade da solução de forma estruturada. Ambos os processos auxiliam ao usuário no processo de melhoria da solução ou da resolução de problemas de inviabilidades.

Estes aspectos são normalmente considerados nos projetos de algoritmos de otimização e os indicadores de desempenho ou falhas são usualmente relatados nos arquivos de saída

dos otimizadores. Usualmente se relatam as informações da solução do *DAOP* e as próprias saídas dos algoritmos de *NLP*.

Diante dos problemas acima citados, pesquisadores desenvolveram aplicativos que fazem algum tipo de diagnóstico em *LP's* e *NLP's*, mas não para *DAOP's*. A grande maioria se dedica a diagnóstico de problemas de *LP* somente. Aqui, são apresentadas algumas características de alguns *softwares* de diagnóstico, para mostrar as linhas de conduta destes pacotes e os tipos de problemas que os mesmos procuram resolver. Destacam-se aqui os pacotes: *Analyze*, *AIMMS*, *MProbe*, *GAMSCHK* e o *GAMS/Examiner*.

O pacote *Analyze*, desenvolvido por Greenberg (1987 e 1996), procura analisar modelos e soluções de problemas de *LP*. Este pacote verifica e valida modelos de otimização, onde o foco principal está em permitir que se efetue o diagnóstico de inviabilidades no problema em questão. Este pacote permite identificar erros nos dados de entradas e inconsistências estruturas no modelo. Ele utiliza a técnica de conjunto irredutível de inviabilidade (*IIS*) (Van Loon, 1981). Além disso, permite realizar análise de sensibilidade pós-ótima (a posteriori).

Outro pacote conhecido é o *AIMMS* (Roelofs e Bisschop, 2010). O módulo de diagnóstico foi desenvolvido por McCarl (1998), e verifica erros de lógicas e/ou modelagem que causam inviabilidades ou resultados espúrios. Uma causa comum é a presença de erros nos coeficientes da matriz A da programação linear ou limites inadequados ou omitidos involuntariamente.

Este pacote contém um módulo chamado "*inspetor"* de programação matemática, que procura criar condições para responder perguntas como: Por que o modelo é inviável ou ilimitado? Por que o valor da função objetivo está muito diferente do esperado? Por que as margens de preços são tão irreais? Através da visualização das variáveis, restrições, objetivos e matriz A (Jacobiana), pode-se identificar erros de formulação ou de dados de entrada que causam inviabilidades, ilimitação ou soluções irreais. Para auxiliar nesta visualização, pode-se utilizar também a técnica de *IIS*, para identificar e eliminar restrições problemáticas em *LP's*. Além disso, o pacote apresenta um mapeamento estruturado do problema, apresentando inclusive estatísticas sobre as variáveis e restrições do problema de otimização.

O *MProbe* (Chinneck, 2001) é outra ferramenta de uso geral para análise de *LP's* e *NLP's*. E procura responder as mesmas perguntas dos pacotes citados anteriormente. Por se aplicarem a problemas de *NLP*, ele analisa também a concavidade da superfície de busca (É côncavo ou convexo?). Para isso, ele faz um teste de concavidade, analisando a diferença entre o valor verdadeiro da função e o valor do hiperplano interpolado no intervalo considerado.

Dentre vários outros pacotes, tem-se o *GAMS/Examiner* (GAMS, 2011), o *GAMSCHK* (McCarl, 1998) e o *MINOS-IIS* (Chinneck, 1994), que analisam a inviabilidade do problema e a otimalidade da solução obtida em LP's e em alguns casos em *NLP's*.

As aplicações de *RTO*, como o *ROMeo* (SimSci, 2002), também tem infra-estrutura de monitoração e diagnóstico da solução obtida pelo otimizador. As ferramentas de

monitoração são bem estruturadas, mas as facilidades de diagnóstico são bem rudimentares. Elas se limitam a registrar as informações geradas pelos algoritmos de *NLP*. Outra ferramenta de monitoração e diagnóstico de *RTO* é o *Aspen RTO Watch* (Adams, 2003). Ela registra dados históricos das soluções obtidas, monitora e analisam desvios do estado estacionário, eventos discretos gerados pelos operadores, e deposita informações em bancos de dados para auxiliar no diagnóstico de problemas. Este pacote tem uma interface homem-máquina amigável, que verifica determinados resultados (dos ajustes de parâmetros, otimização, e detecção de estado estacionário), calcula alguns indicadores de desempenho (ex.: eficiência da otimização, fator de serviço, estatísticas das rodadas, resumo das restrições ativas, dentre outros).

Este panorama resumido mostra a carência de sistemática de diagnóstico de otimizadores, principalmente os dinâmicos. Na maioria dos casos, são disponibilizados apenas os relatórios de saídas dos algoritmos de *NLP*, e em outros casos há ferramentas que analisam problemas e inviabilidades de *LP's*. E, finalmente, alguns sistemas de *RTO* têm ferramentas de monitoração eficientes e outras de diagnóstico com as mesmas informações geradas pelos algoritmos de *NLP*. Além disso, há iniciativas com interesses específicos de monitorar e diagnosticar sistemas de *RTO* (Zyngier, 2006), com enfoque nos efeitos das incertezas do modelo de otimização nos resultados do otimizador, onde se procura estimar o máximo ganho esperado sujeito a estas incertezas nos parâmetros do modelo (denominado *Profit GAP*). Com esta análise, planejam-se experimentos para ajustes de parâmetros do modelo de forma a maximizar os resultados. Esta técnica foi aplicada a problemas de mistura em linha de gasolina, usando modelos de *LP*.

Estes fatos estimulam a realização de análises e soluções de problemas em otimizadores de forma sistemática. Isto remete à utilização de conceitos de garantia de qualidade. No intuito de obter informações sobre as sistemáticas de diagnóstico, verificou-se que os conceitos contidos na técnica de seis sigmas, juntamente com a metodologia *DMAIC* podem ser úteis na idealização de uma metodologia de diagnóstico e na construção de uma ferramenta apropriada com este propósito.

O objetivo principal desta técnica é identificar, eliminar e prevenir falhas ou defeitos ocorridos de forma proativa. Este processo passa pela análise de falhas e aprendizado com os problemas passados, e com isso, melhorar o funcionamento dos processos. Esta técnica tem algumas etapas importantes, são elas: a etapa de detecção, análise, correção e prevenção. Ao analisar as falhas, determinam-se as suas causas básicas e em seguida, proporem-se maneiras de eliminar os problemas encontrados para melhorar a robustez e desempenho dos processos ou sistemas.

Uma boa estratégia é se concentrar no uso de técnicas que possam ajudar a detectar e analisar falhas de sistemas, e também prevenir possíveis problemas futuros. Sempre que possível, agir de forma proativa, de modo a eliminar as causas e atenuar seus efeitos antes que os usuários sejam afetados pelas falhas. Esta é a filosofia principal da garantia de qualidade. Por outro lado, uma estratégia inadequada está em permitir que ocorram falhas sem as devidas explicações das suas causas básicas, e principalmente deixar a cargo do usuário encontrá-las e então reagir no intuito de resolver o problema.

Utilizar metodologias e ferramentas adequadas para resolver problemas com os sistemas de automação são práticas que permitem o aprendizado e a melhoria da qualidade dos processos e produtos. A experiência dos técnicos de per si, sem teoria provoca o aprendizado, porém não é eficiente no processo de melhoria de qualidade e competitividade. Por isso, é necessário utilizar uma abordagem sistêmica, onde são respondidas questões do tipo: O que? Como? Por quê? E, além disso: Quem? Quando? Não adianta simplesmente colocar pessoas em uma sala de reuniões e discutir o assunto em questão. A metodologia seis sigma vem estruturar este processo de solução de problemas, e está apoiada em três pilares fundamentais, são eles:

Métodos - são ferramentas e procedimentos eficazes;
Medição - são quantificações e comunicação de problemas e resultados;
Controle - é o plano ou processo que resolve os problemas detectados pela medição.

O processo seis sigma é uma metodologia que envolve cinco atividades, utilizando o conceito conhecido como modelo *DMAIC* (Tayntor, 2007), que é o acrônimo das suas fases: Definir, Medir, Analisar, Incrementar (Melhorar) e Controlar.

3.5.2 Metodologia *DMAIC*

A metodologia *DMAIC* é baseada na medida do desempenho do processo, onde a análise do comportamento do processo é baseada em fatos e dados. A robustez da solução é obtida com a monitoração e diagnóstico dos resultados do otimizador, buscando a excelência na operação minimizando as taxas de falhas e não-conformidades na operação. Esta metodologia é composta das seguintes atividades:

Definir

Esta atividade consiste em definir o problema a ser resolvido. Isto passa pela definição das necessidades a serem atendidas e dos desejos dos usuários, transformando-as em especificações do sistema. Deve-se definir claramente o escopo do problema a ser resolvido e da solução a ser melhorada, traçar objetivos e metas a ser alcançadas. Para isso, precisam-se identificar as funcionalidades e saídas do sistema de otimização. Na caracterização das funcionalidades, deve-se procurar identificar as fontes de informação, as caracterização das entradas envolvidas, do entendimento das funções e operações envolvidas, caracterização às saídas do sistema e da identificação dos destinos das saídas do sistema. Esta caracterização ajuda a definir quais saídas permitem avaliar o desempenho do sistema em relação aos requisitos do usuário. Na definição das saídas, devem-se priorizar as saídas mais importantes e com isso definir as especificações a serem atendidas e resultados a serem alcançados. Como resultado desta atividade, obtém-se uma definição objetiva dos problemas a serem resolvidos, bem como das variáveis criticas e suas especificações.

Medir

Servirá para dar a visão de como está o desempenho do sistema e indicar pontos de oportunidades de melhorias. Esta etapa consiste em acompanhar as variáveis-chave que afetam a eficiência e eficácia do sistema. Para isso, é necessário ter um plano de coleta de dados que possa assegurar a representatividade e fidelidade da análise a ser feita. E desta

forma, estabelecer as não conformidades encontradas em relação às especificações estabelecidas para o problema de otimização (limites de controle e de especificações de produtos e processo).

A etapa de medida consiste em quantificar o problema utilizando indicadores e estatísticas que demonstrem nível de desempenho de cada etapa da solução do problema de otimização, identificando os pontos críticos e passíveis de melhoria. As entradas e saídas do sistema precisam ser mensuradas, estabelecendo as relações causa-efeito dos problemas encontrados.

Analisar

Nesta etapa, deve-se analisar e determinar as causas básicas dos problemas que precisam de melhoria (poucas e vitais). Determinar relações de causa-efeito entre as variáveis de entrada e saídas. A análise dos resultados permite identificar falhas no sistema de otimização, motivando aos usuários a buscar as causas primárias dos seus problemas. Isto normalmente leva ao estabelecimento de hipóteses, à formulação de experimentos e à elaboração de planos de ação visando realizar melhorias no sistema de otimização. Devem-se aplicar ferramentas analíticas e estatísticas para identificar as causas dos problemas e priorizar as melhorias a serem feitas. Na identificação das causas potenciais de cada problema encontrado é necessário fazer uma análise e visualização de dados, análise de correlação e de benchmark, para se descobrir as relações causais dos problemas encontrados e/ou de desempenho insatisfatório.

Melhorar

Melhorar os resultados do sistema significa eliminar as causas dos problemas. Deve-se propor uma avaliação e implementação de ações e soluções que eliminem a causas identificadas do problema. Muitas vezes é necessário fazer testes para resolver tal problema. A etapa de melhoria também passa por analisar oportunidades oferecidas pelo problema de otimização em questão. Isso passa por um estudo de casos de otimização, onde são exploradas oportunidades desconhecidas. Estes estudos incluem a análise de sensibilidade dos resultados aos parâmetros e ações de controle. No processo de melhoria, deve-se selecionar a melhor solução ou melhores soluções dentre as opções disponíveis. Isso pode passar por uma análise de benchmark do sistema.

Controlar

Esta atividade consiste em controlar o desempenho do sistema. Devem-se estabelecer sistemas de monitoração e controles que garantam que os resultados obtidos sejam mantidos em longo prazo. São definidas medidas padronizadas para manter desempenho do sistema de *DRTO*. Este processo consiste na análise e correção do problema sempre que necessário. O estabelecimento de um sistema permanente de avaliação e controle é fundamental para garantia da qualidade alcançada e identificação de desvios ou novos problemas, os quais devem exigir ações corretivas e padronizações de procedimentos.

Para garantir que o alcance da meta seja mantido em longo prazo, são implementados diversos mecanismos para monitorar continuamente o desempenho de cada processo.

Dentre as técnicas usuais, destacam-se as seguintes: cartas de controle, planos de controle, relatórios sistemáticos e painéis de controle (onde são apresentados indicadores de desempenho). A fase de controle é muito importante para que o *DMAIC* seja visto como um ciclo, o que garante a sustentabilidade do sistema de otimização de processos.

Diante destas questões acima apresentadas nota-se que a iniciativa de construção da ferramenta de diagnóstico tem um viés forte de criação e de agregação de experiências e estruturação de tópicos específicos das metodologias usadas nos *DRTO's*.

3.5.3 Método de análise das condições de otimalidade

A análise das condições de otimalidade é útil para verificar se a solução ótima obtida pelos métodos diretos respeita as referidas condições no domínio de tempo contínuo. Tanartkit e Biegler (1995 e 1997), Cervantes (2000) e Biegler *et al.* (2002) abordaram muito bem este assunto. Um resumo desta abordagem foi descrita da Seção 3.4.2 - *Adaptação da malhas em otimização dinâmica*, tópico *Adaptação por Métodos Simultâneos*, neste capítulo. Os conceitos e utilização são os mesmos utilizados na adaptação de malhas.

3.5.4 Avaliação de desempenho do otimizador

A comparação de diferentes métodos de otimização é uma atividade que avalia aspectos relativos ao desempenho, robustez e qualidade da solução dos métodos em análise (Duraiski *et al.*, 2008). Desta forma, é importante estabelecer algumas métricas para comparar os métodos analisados. Além disso, é importante estabelecer termos de comparação com padrões de referências e isenta de ambigüidades na interpretação dos resultados. A seguir, são apresentados alguns aspectos deste tópico:

Desempenho do otimizador

Espera-se que certo método tenha nível de desempenho desejado. O mesmo deve ser eficiente: econômico na utilização de recursos. O desempenho do otimizador é avaliado comparando-se vários aspectos, por exemplo: medindo a quantidade de problemas resolvidos com sucesso, o número de avaliações da função objetivo e das restrições, o número de iterações no *loop* externo, tempo de CPU gasto na obtenção da solução, dentre outros. Dentre os diversos aspectos que definem o desempenho de um método podemos destacar:

Eficiência do otimizador. A eficiência representa uma medida segundo a qual os recursos são convertidos em resultados de forma mais econômica. A eficiência refere-se à relação entre os resultados obtidos e os recursos empregados. O software não deve desperdiçar a utilização dos recursos:

$$\chi_i = \frac{100}{N} \sum_{j=1}^{N} \frac{S_j^*}{S_{i,j}} \tag{3.98}$$

onde, para o i-ésimo algoritmo, χ_i é a sua eficiência, $S_{i,j}$ é o seu número de avaliações da função objetivo para resolver o problema j, S_j^* é o menor número de avaliações entre os

algoritmos para resolver o problema j. Quando o i-ésimo algoritmo não consegue resolver o j-ésimo problema, então $S_{i,j} = \infty$.

Eficácia computacional. Neste caso, deve-se medir o tempo de *CPU* gasto em cada caso, bem como a possibilidade de executar em processamento paralelo de forma que a solução possa ser obtida no menor tempo possível, viabilizando o seu uso em aplicações em tempo real. A eficácia mede a relação entre os resultados obtidos e o benchmark, ou seja, ser eficaz é conseguir atingir um dado objetivo:

$$\theta_i = \frac{100}{N} \sum_{j=1}^{N} \frac{T_j^*}{T_{i,j}} \tag{3.99}$$

onde, para o i-ésimo algoritmo, θ_i é a sua eficácia, $T_{i,j}$ é o seu tempo computacional para resolver o problema j, T^*_j é o menor tempo entre os algoritmos para resolver o problema *j*. Quando o i-ésimo algoritmo não consegue resolver o j-ésimo problema, então $T_{i,j} = \infty$.

Robustez do otimizador

Um método é robusto se o mesmo desempenha as funções esperadas mesmo em situações não previstos. A robustez do otimizador é avaliada medindo a quantidade de problemas resolvidos com sucesso e a dependência da solução com a estimativa inicial e com os parâmetros de configuração do otimizador:

$$\eta_i = 100 \frac{N_i}{N} \tag{3.100}$$

onde, para o *i*-ésimo algoritmo, η_i é a sua robustez para resolver problemas, N_i é o seu número de problemas resolvidos e *N* é o número total de problemas.

Qualidade da Solução

Distância do ótimo de referência. Utilizando-se problemas *benchmark*, com solução conhecida, se define qual algoritmo teria mais capacidade de obter a solução ótima. Isto deverá ser feito comparando-se os valores obtidos para a função objetivo e os perfis das variáveis em cada caso.

Obediência às restrições. Verificar se as restrições de trajetória e de desigualdade foram adequadamente respeitadas. Nesta análise, procura-se avaliar os resíduos das restrições.

Consistência da solução. Nesta análise, devem-se fazer as seguintes perguntas: Partindo de diferentes condições iniciais, o algoritmo obtém perfis ótimos coerentes com os resultados esperados? Alterando os parâmetros de sintonia do otimizador, os resultados são consistentes?

A medida da qualidade pode ser medida na seguinte forma:

$$\xi_i = \frac{100}{N} \sum_{j=1}^{N} \frac{(d_j^* + \varepsilon)}{(d_{i,j} + \varepsilon)} \tag{3.101}$$

onde, para o i-ésimo algoritmo, ξ_i é a qualidade da solução obtida, $d_j^* = \min_j d_{i,j}$ é a melhor qualidade da solução para o problema *j*, $d_{i,j}$ é a qualidade da solução para o problema j definida como $d_{i,j} = \frac{\left\| x_{i,j} - x_j^* \right\|_2}{\varepsilon_x} + \frac{\left\| \phi_{i,j} - \phi_j^* \right\|_2}{\varepsilon_\phi}$, ε é a precisão da máquina, ε_x é a tolerância na variável independente, ε_S é a tolerância na função objetivo, x_j^* é a solução exata do problema *j*, $\phi_{i,j}$ é o seu número de avaliações da função objetivo para resolver o problema j, ϕ_j^* é o menor número de avaliações entre os algoritmos para resolver o problema j.

Padrões de referência através de perfis de desempenho

Uma maneira eficiente de realizar uma avaliação de desempenho de algoritmos de otimização é executá-la utilizando ferramentas de *benchmarking*. Esta metodologia foi desenvolvida por Dolan e Moré em 2002, e utiliza padrões de referências para traçar os perfis de desempenho do otimizador. Esta forma de apresentação remove algumas ambigüidades na interpretação dos resultados de otimizadores diferentes com problemas diferentes. Estas métricas fornecem informações sobre a robustez e eficiência do algoritmo, e da qualidade da solução.

A idéia desta metodologia é estabelecer uma medida de relação de um indicador de desempenho do caso em questão (usualmente tempo de *CPU* para resolver o problema de otimização), com o melhor desempenho observado, além de determinar a porcentagem de problemas resolvidos com esta relação de desempenhos. Esta abordagem evita as medidas dimensionais e utiliza apenas medidas relativas. Desta forma, podem-se comparar algoritmos diferentes com problemas diferentes sem entrar no mérito de suas naturezas.

Resultados de benchmark são gerados executando uma configuração **C** em uma seqüência de problemas **P** e indicadores de desempenho de interesse (***KPI***). A **configuração (C)** é um problema *DAOP* usando uma estrutura definida com um determinado conjunto de parâmetros. O **problema (P)** é uma condição operacional definida. Podem-se comparar os desempenhos de casos como o uso de duas estruturas de modelos diferentes; benefício de alterar alguns parâmetros de sintonia do otimizador para rodadas consecutivas (com algumas perturbações), como por exemplo: diferentes definições de restrições (liga/desliga restrições, alteração de seu valor)

Pode-se avaliar e comparar um conjunto de configurações **C** em um conjunto de problemas ***P*** de teste, sendo n_C o número de configurações, n_P o número de problemas e $KPI_{p,c}$ o indicador de desempenho para resolver problema ***p*** de configuração ***c*** (exemplo: tempo de computação, número de avaliação de funções, …). Uma propriedade importante dos perfis

de desempenho é que eles são insensíveis aos resultados em um pequeno número de problemas.

Uma linha de base de comparação é obtida através da comparação do desempenho do problema ***p*** com a configuração ***c*** com o melhor desempenho para qualquer configuração testada com este problema.

O parâmetro ρ representa o desempenho relativo de um determinado *KPI*. É a relação do *KPI* do problema em questão comparado com o melhor *KPI* obtido para o mesmo, e pode ser escrito na forma:

$$\rho_{p,c} = \frac{KPI_{p,c}}{\min\{KPI_{p,c} : c \in C\}} \text{ ou } \rho_{p,c} = \frac{\max\{KPI_{p,c} : c \in C\}}{KPI_{p,c}};\ 1 \le \rho_{p,c} \le \rho_M \quad (3.102)$$

incluindo qualidade da solução, a expressão toma a forma de:

$$\begin{cases} \rho_{p,c} = \dfrac{KPI_{p,c}}{\min\{KPI_{p,c} : c \in C\}} & se \quad \xi_{p,c} = \dfrac{d_p^* + \varepsilon}{d_{p,c} + \varepsilon} \ge \delta \\ \rho_{p,c} = \rho_M & se \quad \xi_{p,c} = \dfrac{d_p^* + \varepsilon}{d_{p,c} + \varepsilon} < \delta \end{cases} \quad (3.103)$$

e δ podem ser categorizadas em "*Muito Ruim*", "*Ruim*", "*Aceitável*", "*Bom*", "*MuitoBom*".

O parâmetro ρ_M significa a relação máxima possível de ser obtida. Ela deve ser um valor tal que $\rho_M \ge \rho_{p,c} \quad \forall p \in P, c \in C$ e $\rho_{p,c} = \rho_M$ se e somente se a configuração ***c*** não resolver adequadamente o problema ***p***. O desempenho da configuração **c**, para um dado problema, pode ser obtido através de uma avaliação de desempenho global da configuração **c**, sendo definida como:

$$\eta_c(\tau) = \frac{1}{n_p} size\{p \in P : \rho_{p,c} \le \tau\} \quad (3.104)$$

onde $\rho_c(\tau)$ é probabilidade para uma configuração $c \in C$ que tenha uma razão de desempenho $\rho_{p,c}$ esteja dentro de um fator $\tau \in \Re$ da melhor relação possível. A função η_c é a função de distribuição cumulativa para razão de desempenho $0 \le \eta_c(\tau) \le 1$, como mostra o exemplo ilustrativo da Figura 3.37. Se desejar avaliar a eficiência de uma configuração, deve-se analisar $\eta_c(\tau = 1)$, que é a probabilidade da configuração superar todas as outras configurações.

A escolha de ρ_M deve ser criteriosa, onde $\rho_{p,c} \in [1, \rho_M]$, e deve ser tal que $\eta_c(\rho_M) = 1$ e a probabilidade η_c^* da configuração ***c*** ter sucesso ao resolver o problema seja $\eta_c^* \equiv \lim_{\tau \to \rho_M} \eta_c(\tau)$.

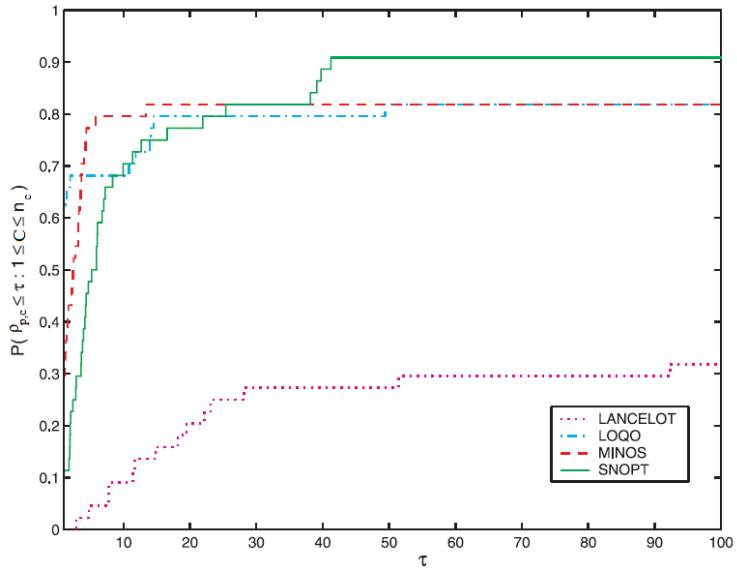


Figura 3.37 – Perfil de desempenho - sub-conjunto do *COPS* (Dolan e Moré, 2002).

Se o usuário estiver interessado em analisar as configurações com alta probabilidade de sucesso, então é necessário comparar diferentes η_c^* e escolher a configuração com o maior valor de η_c^*. O valor da η_c^* pode ser obtidas encontrando η_c linhas base para valores grandes de τ.

3.5.5 Análise de sensibilidade da solução

Análise de sensibilidade dos sistemas de *DAE's* desempenha um papel importante otimização dinâmica. Ela é utilizada na obtenção da solução do controle ótimo pelos métodos seqüenciais ou métodos indiretos, na análise da otimalidade da solução, na análise da solução do otimizador. A análise de sensibilidade tem sido estudada de longa data (Fiacco, 1983). Vários estudos têm sido feitos para aprimorar os métodos de obtenção das sensibilidades dos sistemas *DAE* (Maly e Petzold, 1996; Li e Petzold, 2000 e Li et al., 2000). A otimização dinâmica é consumidora de tempo de máquina, e há a necessidade de executar cálculos de sensibilidade de forma rápida e eficiente.

Uma maneira comum de obtenção de informações de sensibilidade para sistemas dinâmicos é a integração numérica de um sistema *DAE* aumentada com equações de sensibilidade. Nos últimos tempos, este assunto tem recebido a atenção de pesquisadores que tem despendido certo esforço para construir algoritmos eficientes. Maly e Petzold (1996), por exemplo, propuseram um método corretor simultâneo, onde os sistemas *DAE* e sensibilidade são resolvidos simultaneamente. Ao longo dos últimos anos, têm sido apresentados alguns métodos associados a softwares especializados. Schlegel et al. (2004) sugeriram um método de extrapolação baseado na diferenciação implícita. O método foi implementado no código *LIMEX* para integração de *DAE* por Deuflhard et al. (1987 e 1990). Um pouco depois foi acrescentada ao *DASPK* uma implementação eficiente da integração das sensibilidades utilizando as equações adjuntas (Li e Petzold, 1999 e 2002).

O cálculo das sensibilidades de um sistema *DAE* pode ser descrito da seguinte maneira.
Seja um sistema *DAE* representado na forma:

$$\dot{x} = F(x, y, u, p, t); \qquad x(0) = x_0 \tag{3.105a}$$

$$G(x, y, u, p, t) = 0 \tag{3.105b}$$

Diferenciando (3.105a) com respeito a cada variável de controle, temos o sistema de equações de sensibilidade:

$$\frac{d\dot{x}}{du} = \frac{\partial F}{\partial x}\frac{\partial x}{\partial u} + \frac{\partial F}{\partial y}\frac{\partial y}{\partial u} + \frac{\partial F}{\partial u} \tag{3.106}$$

Fazendo $s^x = \dfrac{\partial x}{\partial u}$ e $s^y = \dfrac{\partial y}{\partial u}$ tem-se:

$$\dot{s}^x = \frac{\partial F}{\partial x} s^x + \frac{\partial F}{\partial y} s^y + \frac{\partial F}{\partial u}$$

Diferenciando (3.105b) com respeito a cada variável de controle, temos o sistema de equações de sensibilidade:

$$\frac{\partial G}{\partial x}\frac{\partial x}{\partial u} + \frac{\partial G}{\partial y}\frac{\partial y}{\partial u} + \frac{\partial G}{\partial u} = 0 \quad \text{ou} \quad \frac{\partial G}{\partial x} s^x + \frac{\partial G}{\partial y} s^y + \frac{\partial G}{\partial u} = 0 \tag{3.107}$$

Isolando s^y, obtém-se:

$$s^y = -\left(\frac{\partial G}{\partial y}\right)^{-1}\left(\frac{\partial G}{\partial x} s^x + \frac{\partial G}{\partial u}\right) \tag{3.108}$$

Portanto, $\dot{s}^x$ é acrescentado ao sistema *DAE* original e integrado como um sistema *DAE* aumentado. E s^y é calculado diretamente a partir da sua equação algébrica.

3.5.6 Avaliação do erro de quadratura

Nos tópicos apresentados na seção *3.4.2 - Adaptação da malhas em otimização dinâmica*, *Adaptação pelo Método de Betts* e *Adaptação por Métodos Simultâneos*, são apresentados os critérios de avaliação de erro de quadratura. A seguir são apresentados alguns critérios de avaliação de qualidade das malhas discretas.

Uma malha deve ser mais regular, onde os seus requisitos de qualidade estão baseados na limitação do erro de truncamento e na suavidade da função. Para propor um método de otimização de malhas, é necessário construir um critério de desempenho, que irá conduzir o processo de otimização. Normalmente esta função procura a regularidade e adaptação da malha.

A regularidade da malha significa que os elementos adjacentes devem ter tamanhos semelhantes. A função regularidade para uma dimensão é dada por:

$$J_R = \sum_{i=1}^{N-1} \left(h_{i+1} - h_i\right)^2 \text{ ou } J_R = \sum_{i=1}^{N-1} \left(\frac{h_{i+1}}{h_i} - 1 \right)^2 \text{ onde } h_i = x_i - x_{i-1} \text{ para } i = 1, ..., N \qquad (3.109)$$

É importante ressaltar que os critérios de regularidade e adaptação são concorrentes, isto é, a adaptação tende a concentrar pontos em locais de maior erro ou gradiente, reduzindo a regularidade da distribuição. O critério de adaptação pode ser representado pela Equação 3.110:

$$J_A = \sum_{i=1}^{N} \left[\phi(x_i) - \phi(x_{i-1})\right]^2 h_i^{\,2} \qquad (3.110)$$

Para cada célula, podem-se combinar os critérios na seguinte forma:

$$J_G = \lambda_R J_R + \lambda_A J_A \qquad (3.111)$$

4 Proposta de sistema de DRTO

A construção de um sistema de otimização dinâmica em tempo real (*DRTO – Dynamic Real Time Optimization*) representa um desafio importante na atividade de otimização de processo. Não há solução no mundo que trate eficientemente o problema de otimização dinâmica em tempo real. Há muitas iniciativas na solução de otimização dinâmica *offline*, onde se estuda um determinado cenário e se estabelece uma política de otimização do mesmo processo. Há várias iniciativas em *NMPC*, onde se controla e otimiza um processo em um horizonte de tempo curto e usualmente com períodos de amostragens fixos e pequenos. Mesmo assim, há poucas soluções industriais em tempo real. Também há algumas iniciativas de otimização dinâmica de processos em batelada com horizontes de tempos curtos (bateladas rápidas). O mercado ainda carece de uma solução de otimização dinâmica em tempo real em que se considere a realização de diferentes receitas de produção, transições de processo na produção (campanhas) e até mesmo estabelecer uma política otimizada de produção quando há mudanças súbitas na produção, como alterações de ordens de produção (como mudanças nas especificações de produtos, e tanques de destino) e manobras operacionais (como tirar um equipamento ou linha de produção de operação).

Um sistema de *DRTO* consiste em um conjunto de aplicações que otimiza a planta dinamicamente. Para isso, de forma semelhante ao *RTO* (*Real Time Optimization*), é necessário realizar a formulação do problema de otimização e obtenção dos perfis ótimos das variáveis de controle e implantá-los, dadas às condições iniciais consistentes e utilizando um solver de otimização.

É importante distinguir uma aplicação de *DRTO* do *NMPC*. O *DRTO* é uma aplicação que tem características que reúne as funcionalidades de controle em batelada ou até mesmo contínuo com mudança de receita, programação de produção (ordem de produção) e operações de horizontes longos. E o *NMPC*, de forma diferente, tem um objetivo claro de controlar a planta e busca um alvo. Por exemplo, num processo de mistura em linha de produtos de qualidades diferentes, o *DRTO* deve buscar a receita ótima onde são minimizadas as folgas de especificações de produtos e prever a qualidade final e o tempo em que o tanque deve encher. Já no *NMPC*, o mesmo tem um horizonte pequeno, onde o alvo é buscar a receita ótima dada pelo otimizador e rejeitar as perturbações que eventualmente aparecerem durante o processamento.

Na sua essência, os dois sistemas são semelhantes, mas na pratica são diferentes. No *NMPC*, as predições devem ser rápidas, o tempo de execução do controlador deve ser pequeno (segundos), o horizonte do modelo é curto (horas) e o modelo simplificado. No *DRTO*, as estimações de estados são menos freqüentes, o tempo execução poder ser um pouco mais demorado (minutos), o horizonte pode ser mais longo (até de dias), e o modelo deve representar adequadamente o processo para horizontes de integração longos.

Os *softwares* de otimização dinâmica, tanto comerciais quanto acadêmicas, não possuem todas as funcionalidades desejadas. Estas funcionalidades se apresentam em *softwares* diferentes. O objetivo desta proposta é utilizar as melhores características destes *softwares* de forma estruturada e propor algumas modificações na forma de uso e novas facilidades.

Além disso, a maior parte das funcionalidades dos *softwares* de otimização dinâmica não é usada nas aplicações *online* (*NMPC's*). Como a tecnologia de *DRTO* é uma técnica em fase de maturação e grande parte das aplicações são acadêmicas ou problemas industriais de pequeno porte, uma dúvida importante consistiu na prova de conceito do uso de otimização dinâmica para processo de grande porte e com modelos complexos. Como foi demonstrado por Almeida e Secchi (2006 e 2011), é possível utilizar esta tecnologia de forma eficiente. Esta avaliação forneceu subsídios importantes na configuração desta proposta, tanto no aspecto de robustez quanto no desempenho, como também para a confiabilidade no sistema proposto.

Primeiramente, são apresentados os requisitos do sistema de *DRTO*, onde são analisadas as necessidades de construção do modelo de *DAOP*, na obtenção da solução do problema de otimização, no gerenciamento do sistema, análise e implementação dos resultados do otimizador, e diagnóstico & sintonia do otimizador.

Em seguida é apresentada a estrutura da ferramenta de *DRTO*, onde são descritas as propostas para as aplicações *online* e *offline*. São descritas as características e propostas de funcionalidades de cada módulo do sistema de *DRTO*, considerando os aspectos de construção do modelo, de solução de *DAOP*, de gerenciamento das aplicações, avaliação e implementação de resultados, e monitoração, diagnóstico e sintonia). Além disso, são descritas as formas de uso do sistema proposto.

Uma vez apresentada a estrutura do otimizador em tempo real, apresenta-se a proposta de ferramenta de monitoração e diagnóstico de *DRTO*. São apresentadas nesta seção a metodologia, requisitos e o processo de monitoração e diagnóstico. Neste processo são apresentadas as propostas das ferramentas de monitoração e diagnóstico & sintonia do sistema de *DRTO*. Finalmente são apresentadas as formas de uso da ferramenta de diagnóstico associada à de monitoração, onde são abordadas as duas importantes utilidades no diagnóstico de falhas e busca de novas oportunidades de otimização.

Por último, é efetuada uma análise critica e apresentadas as contribuições feitas para os sistemas de *DRTO*.

4.1 Proposta de ferramenta de DRTO

O sistema de *DRTO* consiste de um conjunto de ferramentas para realizar otimização dinâmica do processo em tempo real. Para isso, precisa de aplicações conectadas à planta que solucionem o problema de otimização dinâmica em malha fechada, sugerindo perfis ótimos de controle. Como parte das atividades do sistema, tem-se a construção, configuração, ajuste do modelo de otimização. Também faz parte desta atividade o acompanhamento dos resultados do otimizador, a busca de oportunidades de otimização e de solução de problemas encontrados durante a execução do otimizador.

O *DRTO* que está sendo proposto neste trabalho deve integrar o conjunto de facilidades do ambiente *EMSO* (*Environment for Modeling, Simulation and Optimization*), desenvolvido por Soares (2003). O *software EMSO* tem por objetivo principal o desenvolvimento de um ambiente integrado de ferramentas computacionais para simulação e otimização de processos. Este simulador pode realizar simulações estacionárias e dinâmicas de processos

de forma eficiente, e tem capacidade de tratar de sistemas de equações algébrico-diferenciais de índice elevado (Brenan et al., 1989; Costa et al., 2003), que é importante para a solução de problemas de *DRTO*.

O coração de um sistema de *DRTO* é constituído da solução de um problema de otimização dinâmica no domínio do tempo contínuo, representado por um problema de otimização de um sistema algébrico-diferencial (*DAOP - Differential-Algebraic Optimization Problem*). O modelo do processo é escrito na forma de equações algébrico-diferenciais (*DAE - Differential-Algebraic Equation*) e as condições iniciais consistentes são fornecidas por um estimador de estados baseado em medidas do processo. Nas formulações destes problemas ainda podem estar presentes restrições de caminho, de pontos interiores e de tempo final. O problema de otimização pode ter somente um estágio ou múltiplos estágios de tempo. Normalmente os estágios são definidos por receitas de operação ou quando há alguma mudança na estratégia de otimização (ex.: etapa de aquecimento e resfriamento de um reator). Há situações onde são incluídos estágios para isolar os arcos singulares presentes nos perfis de controle e o tempo final no estágio passa a ser uma variável de controle.

Os problemas de otimização dinâmica são normalmente resolvidos através de métodos diretos, que transformam a *DAOP* em um problema de *NLP* com parametrização das variáveis de controle. É comum a utilização de aproximações de diferenças finitas para resolver problemas de otimização dinâmica (na utilização de métodos diretos). Os métodos de otimização dinâmica devem ser escolhidos conforme sua conveniência.

Para resolver estes problemas, é necessário utilizar algoritmos poderosos. Precisa-se de uma boa formulação do problema de otimização, implementação eficiente de algoritmos de otimização e software que pode obter as melhores soluções com certa rapidez. As principais características da otimização em tempo real são: deve ser eficiente (especialmente para problemas de grande porte), precisa e robusta. O pacote deve ser capaz de resolver a maioria dos problemas com sucesso.

Um conjunto de aplicações é necessário para a realização da otimização dinâmica em tempo real, tem-se a aplicação de otimização dinâmica *offline*, um conjunto de aplicações de otimização *online* e de ferramentas de monitoração e diagnóstico da otimização dinâmica.

O sistema de *DRTO* é um conjunto de ferramentas *online* para realizar uma otimização dinâmica em tempo real, sendo necessário formular o problema (*DAOP*), fazer a aquisição das informações da planta, inicializar a otimização, resolver o *DAOP* e implementar as ações de controle na planta ou em aplicações de controle avançado. Isto pode ser visto na Figura 4.1.

Na etapa de aquisição e tratamento de dados dos instrumentos e inicialização, realiza-se a coleta de dados dos instrumentos da planta, a validação de dados coletados com eliminação de erros grosseiros, o ajuste de parâmetros do modelo e a estimativa dos estados iniciais do processo, passando por uma reconciliação de dados transientes. Na etapa de formulação do problema de otimização dinâmica, configura-se o problema de otimização em alguma interface adequada onde se estabelece a função objetivo, constrói-se o modelo do processo e se estabelece as restrições do problema. Na etapa de solução do problema de otimização

dinâmica, executa-se o tratamento do problema de otimização (ex.: discretização das variáveis) deixando-o na forma adequada para ser resolvido por um algoritmo de otimização, calculando a trajetória ótima das variáveis de controle. Uma vez resolvido o problema, as trajetórias calculadas devem ser validadas e implementadas nas aplicações de *MPC*, na forma de trajetórias de referência e valores alvos das variáveis manipuladas pelo *MPC*, ou diretamente nos instrumentos da planta, na forma de *setpoints* de controle regulatório.

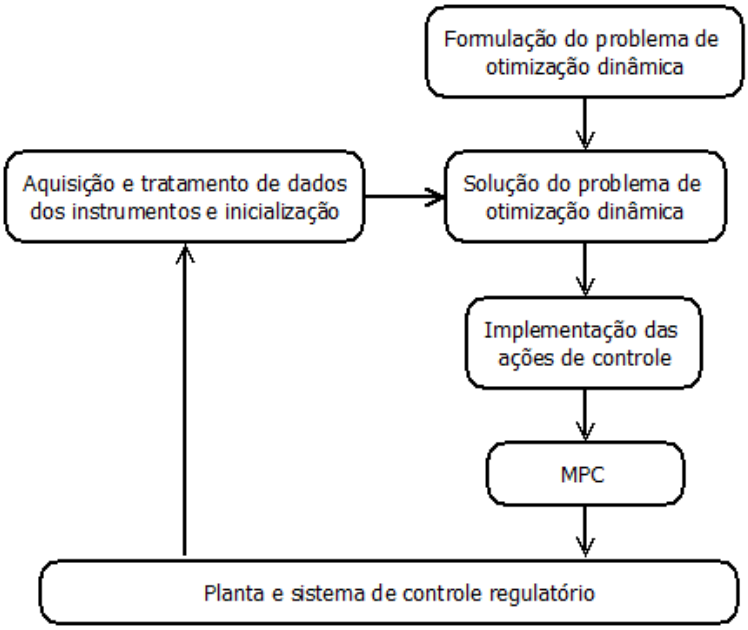


Figura 4.1 – Estrutura geral do *DRTO* – Modo *online*.

A aplicação de otimização *offline* é executada de forma desconectada da planta. Ela é utilizada pelos engenheiros de automação e de processos na construção do modelo de otimização dinâmica, calibração do modelo desenvolvido, estudo de casos de otimização dinâmica, simulação e visualização da solução do otimizador *online* e no diagnóstico do otimizador *online*.

A estrutura da otimização dinâmica *offline* é semelhante à aplicação *online*, onde a coleta de dados de instrumentos da planta é substituída por um arquivo de dados e a implementação das ações de controle é substituída por um arquivo de relatório ou um visualizador de resultados. Ao contrário da aplicação *online*, a aplicação *offline* não é executada de forma cíclica, podendo ser feito através de uma interface amigável. A Figura 4.2 mostra a sua estrutura geral.

A aplicação de otimização dinâmica *offline* pode ser utilizada para construir o modelo de otimização, para realizar estudos de casos, estudos de benefícios e auxiliar no diagnóstico de problemas ocorridos com o otimizador. Na realização de estudos de casos, a aplicação auxilia na busca de oportunidades (Ex.: modificando alguma especificação do processo, avaliando alternativas de qualidade carga, usando uma configuração diferente do processo, recolocando equipamentos em operação, dentre outros). Esta aplicação também pode ser utilizada na análise dos resultados do otimizador (simulando e visualizando os resultados utilizando os perfis ótimos obtidos pelo otimizador *online*), para verificação da aderência

dos resultados com a ordem de produção estabelecida no planejamento e programação de produção.

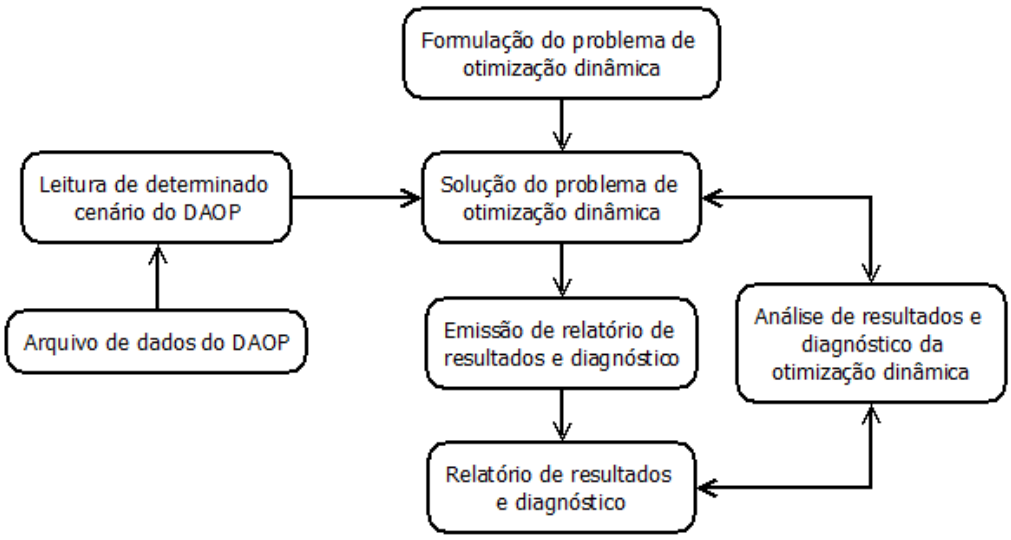


Figura 4.2 – Estrutura geral da otimização dinâmica *offline*.

Para projetar um sistema de *DRTO*, é importante estabelecer primeiramente os requisitos do sistema, onde são estabelecidas a aplicabilidade e modos de uso da aplicação. Além disso, são estabelecidos os requisitos básicos para os algoritmos (como robustez e eficiência), para cada macro-atividade, e para a estrutura do sistema de *DRTO*. Com base nestes requisitos, é proposta uma estrutura para o sistema e feita uma análise critica da solução projetada.

4.1.1 Requisitos do sistema de *DRTO*

Os problemas de otimização dinâmica podem ter pequenas ou grandes dimensões e são de diferentes níveis de complexidades, em função da quantidade de variáveis envolvidas e da natureza das equações que descrevem o processo. Quando o problema de otimização se torna muito grande, precisamos de um algoritmo de otimização que saiba lidar com estruturas de problemas mais complexas. Os algoritmos que resolvem este tipo de problema de otimização tendem a serem complexos com maiores números de parâmetros de sintonia e mais difíceis de serem utilizados.

Um dos fatores de sucesso mais importantes no uso de aplicativos em tempo real, além de resolver eficientemente o problema de otimização, é a capacidade de diagnosticar e resolver problemas quando o otimizador de falhar. Ao resolver problemas e responder às perguntas dos usuários, é possível obter maior credibilidade e confiabilidade na aplicação. E, como conseqüência, os operadores não interrompem a utilização do aplicativo. Com o objetivo de obter essa confiança do pessoal operacional, propõe-se o uso de ferramentas de diagnóstico e sintonia. Para se obter sucesso no uso do sistema é necessário ter bons algoritmos implementados. Um bom algoritmo deve ter os seguintes atributos:

Robustez - O algoritmo deve ser confiável para toda a diversidade de problemas a serem resolvidos e deve ser garantida teoricamente a convergência para uma solução a partir de

qualquer ponto de partida. Por exemplo, em aplicação *DRTO*, diferentes qualidades de alimentação, pontos de operação, algumas perturbações no processo, ou então estruturas de problemas diferentes, tais como, diferentes ordens de produção da área de programação. A robustez do otimizador é avaliada pela medição da quantidade de problemas resolvidos com sucesso e da dependência da solução em relação à estimativa inicial e aos parâmetros de configuração do otimizador.

Generalidade - O algoritmo deve ser capaz de resolver problemas com estruturas diferentes, com ou sem restrições de igualdade e desigualdade, lineares ou não lineares. Além disso, o algoritmo deve também ser capaz de lidar com problemas com diferentes tipos de funções objetivo, linear ou não-linear, com um ou múltiplos estágios, e com um ou múltiplos objetivos.

Eficiência - o algoritmo deve ser eficiente, com taxa de convergência rápida. Na prática, tem-se um tempo limitado pelo ciclo de execução do sistema *DRTO* para resolver o problema de otimização. O desempenho do otimizador é avaliado comparando-se vários aspectos, tais como: número de avaliações da função objetivo e das restrições, número de iterações, esforço computacional na obtenção da solução, dentre outros. Uma das formas comuns de medição da eficiência do algoritmo é através da eficácia computacional. Neste caso, deve-se medir o tempo de *CPU* gasto em cada caso, bem como a possibilidade de executar em processamento paralelo de forma que a solução possa ser obtida no menor tempo possível, viabilizando o seu uso em aplicações em tempo real.

Facilidade de uso - o algoritmo deve ser fácil de usar e configurar pelos engenheiros de automação. Algoritmos de otimização geralmente têm muitos parâmetros de sintonia que são dependentes do problema. Isso significa que são exigidas mudanças freqüentes de valores e tornando-se assim difícil de usar na prática. Algumas vezes é obrigado a ter essa complexidade, devido à necessidade do uso de algoritmos robustos, precisos e eficientes.

Qualidade da Solução - o algoritmo deve ter a capacidade de convergir para uma solução ótima precisa. Ou seja, o otimizador deve convergir para uma solução onde as tolerâncias nas violações de restrições e viabilidades sejam aceitáveis pela planta real e engenheiros. Algumas formas de avaliações podem ser destacadas, são elas: ***Distância do ótimo real*** - Utilizando-se problemas *benchmark*, com solução conhecida, qual algoritmo teria mais capacidade de obter a solução ótima? Isto deverá ser feito comparando-se os valores obtidos para a função objetivo e os perfis das variáveis em cada caso. ***Obediência às restrições*** - Verificar se as restrições de trajetória e de desigualdade foram adequadamente respeitadas. Nesta análise, procura-se avaliar as violações das restrições. ***Consistência da solução*** - Nesta análise, devem-se fazer as seguintes perguntas: Partindo de diferentes condições iniciais, o algoritmo obtém perfis ótimos coerentes com os resultados esperados? Alterando os parâmetros de sintonia do otimizador, os resultados são consistentes?

É importante notar que existe um equilíbrio entre a robustez, qualidade da solução, generalidade e facilidade de uso versus eficiência. E algumas vezes, é necessário recorrer a algumas estratégias computacionais, como processamento paralelo e códigos de computação de alto desempenho, a fim de priorizar a robustez e qualidade da solução. A utilização de ferramenta de diagnóstico e sintonia pode ser muito útil para auxiliar ao engenheiro encontrar um ajuste adequado do otimizador.

O sistema de *DRTO* deve ter uma estrutura que permita todas as formas de sua utilização, e deve ter as seguintes facilidades:

- construção do modelo de otimização;
- atualização do modelo de otimização;
- solução do problema otimização;
- avaliação e implementação dos resultados;
- gerenciamento e seqüenciamento de tarefas;
- monitoração dos resultados;
- diagnóstico e sintonia.

O módulo de gerenciamento e seqüenciamento de tarefas deve coordenar a maior parte das funções do sistema de *DRTO*, como mostra a Figura 4.3. O modelo de otimização contido no módulo de construção do modelo é utilizado tanto para a tarefa de atualização de modelo e do estado inicial da planta quanto para a solução do problema otimização. Uma vez obtido o resultado do otimizador, o mesmo é avaliado e implementado na planta. No uso da ferramenta de diagnóstico e sintonia, pode-se decidir por resolver o problema de otimização dinâmica em condições específicas.

Figura 4.3 – Estrutura conceitual do *DRTO*.

Com estas definições básicas, é possível estabelecer os requisitos do sistema de *DRTO* a ser projetado. O sistema de *DRTO*, em função de suas particularidades, tem as seguintes funcionalidades: construção do problema de otimização dinâmica, atualização do modelo do processo, atualização do estado da planta, gerenciamento da execução do otimizador, solução do problema de otimização dinâmica, avaliação e implementação de resultados do otimizador, estudos de casos visualização da solução *offline*. Além disso, há funcionalidades auxiliares que ajudam o engenheiro e o operador resolver problemas encontrados durante a solução e a avaliar os resultados obtidos pelo otimizador. Para realizar isso, são necessárias as funcionalidades adicionais de diagnóstico & sintonia e monitoração & avaliação de resultados do otimizador. A seguir são apresentados alguns pontos referentes a funcionalidades requeridas para um sistema de *DRTO*.

4.1.1.1 Requisitos de construção do modelo de otimização dinâmica

O usuário deverá construir o modelo de otimização dinâmica no editor de modelos conforme as regras de linguagem definidas para o software. Estas regras deverão ser adequadas para interpretar os diferentes *DAOP's* descritos no formato padrão da Equação 4.1.

$$
\begin{aligned}
&\underset{u,t_f}{Min} \quad \phi\left(x(t_f), y(t_f), t_f\right) \\
&s.a. \\
&\quad F\left(\dot{x}, x, y, u, t\right) = 0; \quad x(0) = x_0 \\
&\quad x_L^P \le x^P(t) \le x_U^P \\
&\quad y_L^P \le y^P(t) \le y_U^P \\
&\quad u_L^P \le u^P(t) \le u_U^P \\
&\quad x_L^E \le x^E(t_f) \le x_U^E \\
&\quad y_L^E \le y^E(t_f) \le y_U^E \\
&\quad u_L^E \le u^E(t_f) \le u_U^E
\end{aligned}
\tag{4.1}
$$

onde e ϕ é a função objetivo, F o modelo dinâmico do processo, os sobrescritos *P, E, L e U* representam as restrições de caminho, restrições terminais, limite inferior e limite superior respectivamente. Sendo x o vetor de variáveis diferenciais, $\dot{x}$ o seu operador diferencial, y o vetor de variáveis algébricas, u o vetor de variáveis independentes, t o tempo e t_f o tempo final.

O problema de otimização dinâmica (com a descrição do comportamento dinâmico do processo) deve ser formulado de forma fácil e intuitiva. Esta construção deve ser feita usando um ambiente de simulação dinâmica e modelagem de *DAOP*, com uma interface homem-máquina (*IHM*) apropriada. Tanto na otimização *online* como na *offline*, o ambiente de modelagem deve ser o mesmo. A modelagem deve ser feita na forma mais parecida dos procedimentos de modelagem em engenharia química, usando uma linguagem intuitiva para os engenheiros. Deverá existir um interpretador que traduz o problema de otimização dinâmica no formato de modelo de engenharia química para um formato padrão de *DAOP*.

O modelo de otimização dinâmica deverá contemplar tanto problemas de otimização dinâmica de simples estágios quanto múltiplos estágios. Ele poderá ter um único modelo dinâmico do processo para todos os estágios como também modelos diferentes. As restrições e objetivos poderão ser os mesmos ou diferentes em cada estágio.

4.1.1.2 Requisitos para atualização do modelo do processo

O modelo de otimização contém incertezas devido à falta de fidelidade da representação do comportamento do processo, ruídos e erros nos sensores de medição (Forbes e Marlin, 1996). Estas incertezas se transferem para a solução do otimizador, por isso, é necessário manter a fidelidade do modelo com a planta. Desta forma, deve-se calibrar periodicamente

o modelo do processo. Para isso é necessário determinar os parâmetros do modelo com os dados da planta, dispondo de ferramentas estatísticas para analisar a qualidade do ajuste. Como os dados da planta podem conter erros aleatórios e sistemáticos, pode ser necessário passar por uma etapa de reconciliação de dados e detecção de erros grosseiros para remover os erros de medição, sendo necessário dispor de informações confiáveis das variâncias dos instrumentos de medida. Toda esta etapa ou parte dela poderá ser suprimida dependendo da qualidade das informações do processo. Uma solução alternativa é a realização simultânea do ajuste de parâmetros e reconciliação de dados.

4.1.1.3 Requisitos de atualização do estado da planta

Para executar a otimização (tanto *online* como *offline*) é necessária a obtenção das condições iniciais consistentes do processo. As condições iniciais usualmente são obtidas de uma situação real do processo. As condições iniciais consistentes podem ser estimadas tanto no modo *online* como no *offline*.

As condições iniciais devem ser consistentes com o modelo e coerentes com o processo real, pois são de fundamental importância para a solução do *DAOP*. O estabelecimento das condições iniciais do processo é efetuado através da obtenção das informações do processo a partir da aquisição de dados de instrumentos. Estes dados de instrumentos normalmente contêm erros sistemáticos, ruídos ou até mesmo valores falhos. Raramente estes dados de processo são consistentes, ou seja, não satisfazem as equações de balanços em engenharia química, exigidos pelo modelo do processo. Além disso, o número de informações obtidas de instrumentos é limitado e algumas informações são obtidas de entradas manuais, ou seja, nem todos os estados da planta são medidos. Por todos estes fatos, os dados da planta devem ser reconciliados durante a estimação do estado do processo.

4.1.1.4 Requisitos para o gerenciamento de execução do otimizador

Periodicamente o algoritmo de otimização é invocado para resolver um *DAOP*. Os algoritmos de otimização dinâmica podem ter desempenhos rápidos ou lentos, dependendo do problema. E a decisão de executar o otimizador de forma cíclica com freqüência constante pode não ser conveniente, pois podem ocorrer mudanças no processo e ser necessário re-otimizar antes do horário do próximo ciclo. Uma solução interessante é o uso de um mecanismo disparador que monitora o processo e verifica a ocorrência de alterações relacionadas de estrutura do problema de otimização dinâmica ou perturbações significativas (ou seja, perturbam as condições de otimalidade ou viabilidade do *DAOP*) (Kadam et al., 2003) de forma a invalidar a última receita ótima do *DAOP*. Neste momento, algoritmo de otimização deverá ser executado novamente, mesmo que o otimizador esteja em busca de uma solução ótima. Lembre que o otimizador poder levar vários minutos para encontrar uma solução e pode ocorrer algum evento que motive a re-otimização do processo nesse ínterim. Neste caso, a otimização deverá ser interrompida e re-iniciada nas novas condições de processo. Processos contínuos, com diferentes pontos de operação ou freqüentes transições de processo, normalmente exigem uma execução cíclica do otimizador.

4.1.1.5 Requisitos de solução do problema de otimização dinâmica

A resolução do *DAOP* ,em processos químicos, usualmente é efetuada através do uso de métodos diretos, pois os métodos indiretos não são apropriados para resolver a maior parte dos problemas encontrados em engenharia química. Três abordagens são recomendadas: *single-shooting*, *multi-shooting* e colocação em elementos finitos. A escolha do método a ser utilizado é dependente do problema e está ligado ao tamanho e complexidade do problema. Os esforços computacionais de cada método devem ser considerados e se for necessário pode-se ter como opção o uso de computação paralela nas tarefas de avaliações de funções, álgebra linear e integração do sistema algébrico-diferencial.

A primeira fase dos algoritmos dos métodos diretos consiste na tarefa de parametrização das variáveis de controle. Esta parametrização é necessária para os três métodos citados anteriormente. A discretização dos controles é usualmente feita através de aproximações polinomiais por partes, onde as mais comuns são a constante por partes e lineares por partes. Há situações onde os valores iniciais das variáveis de controle podem ser livremente escolhidos, mas a situação mais comum é iniciar a otimização com o valor presente destas variáveis. Além disso, as manipulações dos controles na operação de plantas reais são feitas através de ações de controle em degraus (constantes por partes). Portanto, os otimizadores devem levar em conta esta característica dos sistemas de controle digitais.

Outro fator importante é a qualidade da discretização do horizonte de otimização, onde o número de elementos discretos e os seus intervalos de tempos são fatores cruciais na obtenção de todo o potencial de otimização do *DAOP*. Quanto menor o número de elementos e maiores os intervalos de tempo nos elementos, maior a perda de oportunidade de otimização do processo. Isto se deve ao fato de reduzir o número de graus de liberdade do sistema. Por outro lado, quanto mais elementos discretos no problema, maior o esforço computacional do algoritmo de otimização. Isto pode acarretar num tempo computacional tão grande que torna inviável a utilização em tempo real. Além disso, pode-se ter um excesso de manipulações no processo. Para tratar destes problemas, devem-se utilizar estratégias de adaptação dos perfis de controle, e de agrupamento de elementos (para reduzir o número de manipulações na planta). A forma de utilização destas facilidades deve ser estabelecida em função do problema.

A solução ótima de um *DAOP* pode ser composta por uma série de arcos singulares e não-singulares, onde algumas variáveis de estado e/ou de controle podem ter trechos nos seus limites e outros trechos em transição. Os instantes de chaveamento (restrição ativa para não ativa e vice-versa) são desconhecidos a priori e devem coincidir com um ponto discreto da malha ou com o final de um determinado estágio de otimização. Para encontrar estes momentos, é necessário aplicar algum método de detecção de estrutura de arcos e reformulação do *DAOP*. A realização desta tarefa proporciona um maior refinamento da solução obtida.

4.1.1.6 Requisitos de avaliação e implementação de resultados

Os resultados da solução de um problema de otimização dinâmica são perfis ótimos das variáveis de controle. A solução obtida com a utilização de métodos diretos é uma

aproximação da solução do *DAOP* no domínio de tempo contínuo. Portanto a solução ótima obtida pelo algoritmo de *NLP* podem não satisfazer as condições de otimalidade de Pontryagin. Por isso, deve-se verificar se a solução de um problema de otimização dinâmica é ótima.

As soluções obtidas pelo otimizador podem ser diferentes devido a alterações no processo, no problema de otimização ou até mesmo nos parâmetros do modelo do processo. Os parâmetros do modelo contêm incertezas e as mesmas se transferem para a solução do otimizador. Devido a estes fatores, nem todas as soluções obtidas pelo otimizador devem ser implementadas, sendo necessário analisar os resultados e decidir quando uma nova receita deve ser implementada na operação da planta. O objetivo é evitar que se altere a receita sem necessidade. Esta análise de resultados verifica se a nova solução proposta pelo otimizador é diferente da última. Portanto, é necessário discriminar as soluções obtidas pelo otimizador. Isto reduz significativamente o número de mudanças de direções do processo, como reportam Miletic e Marlin (1996) para aplicações de *RTO*. Além disso, é recomendável verificar se as condições iniciais fornecidas na estimação de estados são significativamente diferentes das obtidas na fase de inicialização do otimizador, pois as condições iniciais afetam diretamente a solução do otimizador. Nem sempre os modelos e métodos de solução de ambos são os mesmos.

Para reproduzir adequadamente as operações de plantas industriais, devem-se calcular as velocidades de alteração dos controles através das manipulações de Δu ou na forma de valores alvos dos controles avançados (*MPC's*). Para realizar esta tarefa, o otimizador deve ter a capacidade de executar receitar em bateladas (tempo e ação), isto é, a cada intervalo de tempo pré-estabelecido, o sistema de *DRTO* deve enviar os valores ótimos de controle para o sistema de controle.

Sistematicamente deve ser realizada uma monitoração dos resultados do otimizador, que define o grau de confiança e satisfação dos usuários com o sistema de *DRTO*. Esta monitoração deve apresentar a evolução das últimas soluções fornecida pelo otimizador (ex.: perfis de controles e estados, falhas na obtenção da solução, tempo de *CPU* para obter a solução, etc.). Devem-se estabelecer indicadores de desempenho (% solução, número de iterações, número de avaliações de funções, ...) e de robustez do otimizador (% problemas resolvidos). Também deve apresentar medidas da qualidade da solução (ex.: distância do ótimo real – usando benchmark, obediência às restrições, ...) e da fidelidade do modelo de processo (ex.: valores dos parâmetros, incertezas das predições, erros sistemáticos das medidas, ...). Estas informações devem ser fornecidas pelo otimizador ao final de cada execução do sistema de *DRTO*.

Na percepção de queda de robustez, instabilidades nas soluções, diferenças nas qualidades das soluções, dentre outros, pode-se demandar pela realização de um diagnóstico, sintonia, estudo de oportunidades de otimização para diferentes cenários e outras ações que a monitoração apresentar desvios.

4.1.1.7 Requisitos para estudos de casos de otimização dinâmica

Uma atividade importante no processo de otimização dinâmica é a realização de estudos de casos com o *DAOP* em questão. Este estudo de casos pode ser realizado, de forma *offline*,

com as condições iniciais de uma determinada rodada do otimizador *online* ou até mesmo a partir de um caso hipotético. Neste último caso, devem-se estabelecer valores iniciais para as variáveis de estado diferenciais e para as trajetórias de controle e realizar a inicialização da simulação dinâmica (neste caso, resolver o sistema *NLA - Nonlinear Algebraic* do modelo).

Os estudos de casos podem ser feitos com diferentes propósitos, são eles: repetir um caso da otimização *online*, verificar a solução do otimizador em condições iniciais diferentes da planta (hipotéticas ou não), realizar análises de sensibilidade da solução do problema de otimização (movendo restrições ou alterando incertezas do modelo), estudar casos de mudanças de estrutura do *DAOP* (ex.: excluir ou incluir restrições) e estudar os efeitos das alterações dos parâmetros de sintonia do otimizador.

Outro ponto importante é a realização da avaliação da robustez da solução do otimizador. Uma solução robusta é aquela onde as violações das restrições e o valor da função objetivo são pouco sensíveis às pequenas variações nos perfis de controle. Se houver alta sensibilidade com alguma variável de controle, é preciso analisar as incertezas da solução associadas à mesma e implementar as ações de controle com parcimônia.

4.1.1.8 Requisitos de visualização da solução do otimizador online

A tarefa de simulação e visualização da solução do otimizador consiste na repetição de um caso obtido pelo otimizador *online* e realizado no modo simulação *offline*. Para realizar isto, é necessário que sejam importadas as informações de condições iniciais, parâmetros do modelo, informações do problema de otimização e trajetórias ótimos de controle. Com isso, pode-se simular a solução do problema resolvido pelo otimizador *online* e verificar a qualidade da solução. Esta verificação é feita sobre violações indevidas ou não das restrições, condições de otimalidade ou até mesmo fazer uma análise de sensibilidade da solução às perturbações nas variáveis de controle.

4.1.1.9 Requisitos de diagnóstico & sintonia do otimizador

Quando houver falha na obtenção de uma solução ou problemas de robustez e desempenho fraco do otimizador é recomendável fazer um diagnóstico do problema ocorrido e, se necessário, fazer uma nova sintonia do otimizador. O diagnóstico deve se focalizar nas informações fornecidas pelo otimizador. As falhas mais comuns são: problemas de convergência ou divergência, problema inviável ou ilimitado, problemas numéricos ou de construção de modelo, e problemas de programação.

Para se fazer um diagnóstico adequado, é necessário que o otimizador forneça informações úteis para que se encontrem as causas do problema. Para isso, o otimizador deve fornecer informações estruturais sobre o modelo dinâmico (ex.: índice, graus de liberdade, singularidade), o algoritmo de otimização (ex.: qualidade da discretização), o solver de *NLP* (ex.: violações das restrições, função objetivo, inviabilidades), o integrador de *DAE* (ex.: não convergência, muitas reduções do passo), dentre outros. O otimizador deve gerar um relatório específico para que seja executado um diagnóstico preciso do problema e deve fornecer as condições iniciais e configuração do problema de forma que seja possível reproduzir o problema ou até mesmo executar testes posteriores.

Ao fazer um diagnóstico de um problema ocorrido durante a solução do otimizador *online*, pode passar por realizar a otimização no modo *offline*. Para isso, é necessário importar os dados da otimização *online* da mesma forma feita para o estudo de casos. A diferença é que podem ser realizados estudos de diagnósticos. Estes estudos podem contemplar os seguintes casos: de otimalidade, viabilidade, iteração a iteração e de sintonia.

O caso otimalidade consiste em verificar qual o melhor valor possível da função objetivo e das trajetórias de controle do problema sem restrições de otimização. Isto é, removem-se todas as restrições que não descaracterize o problema regulatório. Ou seja, mantém as restrições de domínio (ex.: fração molar de *0* a *1*, vazão a bomba *0* a F_{max}), restrições de alvo (ex.: volume final de um produto no tanque – senão transborda). Desta forma, obtém o valor utópico da função objetivo. E se o otimizador *offline* encontrar uma solução, provavelmente tem-se um problema de inviabilidade no problema do otimizador *online*.

O caso viabilidade consiste em verificar a existência do caso regulatório. Isto é, anula-se o índice de desempenho da função objetivo do problema original. Se existir uma solução viável, provavelmente a causa do problema encontrado com o otimizador *online* deve estar associada à convergência ou questões numéricas durante a busca da solução.

Neste caso, também se pode realizar a otimização *offline* com flexibilização das restrições, conforme descrito na parte 2 desta dissertação (solução de inviabilidades em problemas de otimização dinâmica). Com isso, o otimizador deve encontrar uma solução com restrições relaxadas. As medidas dos relaxamentos das restrições fornecem indicações qualitativas e quantitativas dos relaxamentos necessários para tornar o problema viável.

O caso iteração a iteração consiste em executar o problema de otimização (podendo ser o problema do otimizador *online* ou não) parando ao final de cada iteração e verificar as informações das soluções parciais do otimizador. Para isso, propõe-se um controle de pausa no otimizador. Este controle pode ser feito através da gerência de uma variável com as posições "*Encerrar*", "*Parar*" e "*Continuar*". Para encerrar, o otimizador interrompe e finaliza a otimização. Ao parar, o otimizador interrompe momentaneamente sua execução ao final da iteração e apresenta os resultados. Para continuar, o otimizador continua a execução a partir do ponto parado. Esta parada deve ser feita através de um controle introduzido no solver de *NLP* (quando tiver o código fonte) ou no final da reportagem das informações de saída do otimizador.

O caso sintonia consiste na análise de sensibilidade do otimizador às alterações nos valores dos parâmetros do otimizador ou até mesmo de métodos utilizados pelo mesmo. Isto auxilia o usuário encontrar o melhor ajuste do otimizador para uma determinada configuração de problema de otimização.

O sistema de diagnóstico também deve ter capacidade de fazer uma análise da robustez da solução obtida. Ao perceber alguma instabilidade nas soluções do otimizador, pode-se realizar uma análise de robustez utilizando as informações de sensibilidades das restrições e da função objetivo com relação às ações de controle. Estas informações de sensibilidade devem ser fornecidas pelo otimizador durante a execução normal do mesmo.

Também deve ser possível realizar análises de sensibilidade da solução e desempenho do otimizador aos parâmetros de sintonia do mesmo. Podem-se fazer curvas de robustez e desempenho do otimizador e decidir sobre mudanças de sintonia do sistema de *DRTO*.

4.1.2 Estrutura da ferramenta de DRTO

Para atender aos requisitos básicos anteriormente apresentados, propõe-se a seguinte estrutura para o sistema de *DRTO*, conforme representado na Figura 4.4. No processo de otimização dinâmica em tempo real, devem-se receber as informações da planta através de servidor *OPC* conectado ao seu sistema de controle, recebe informações de planejamento e programação de produção através de base de dados de informações e recebe as informações do modelo de otimização dinâmica de arquivo (*DAOP*), gerado pelo editor de modelos. Com estas informações o *DRTO* deverá obter as condições iniciais do problema de otimização, deverá atualizar o modelo do processo e realizar a otimização dinâmica. Uma vez obtidos os resultados do otimizador, os mesmos devem ser analisados e enviados para o sistema de controle da planta via *OPC*. No processo de solução do *DAOP*, o sistema *DRTO* também deve gerar e enviar informações de monitoração e diagnóstico do sistema. As informações para diagnóstico são enviadas por arquivos para posterior análise *offline*. E as informações de monitoração são enviadas para uma base de dados de informações para acompanhamento dos resultados e desempenho do sistema de *DRTO*.

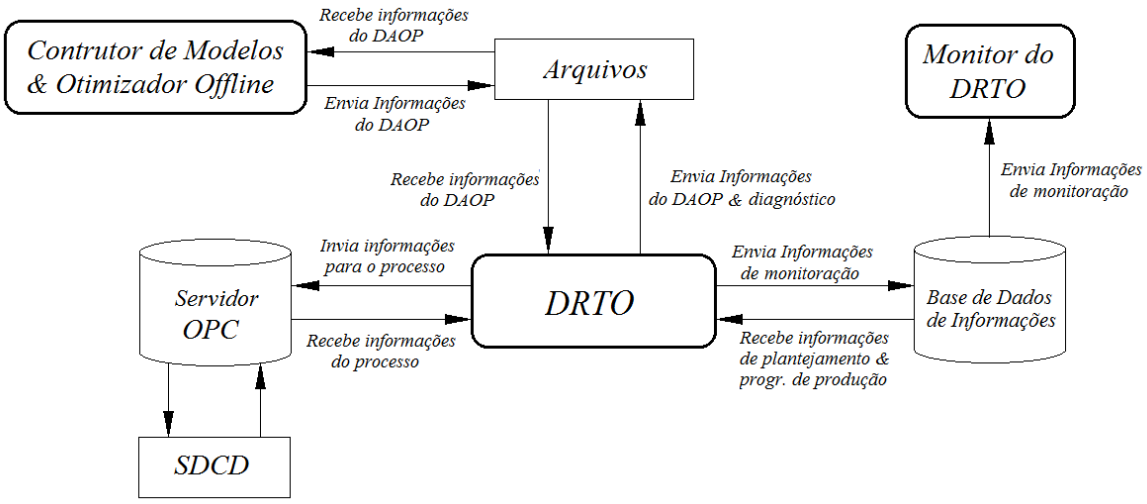


Figura 4.4 – Esquema geral do sistema de *DRTO*.

De forma mais detalhada, o sistema de *DRTO* (vide Figura 4.5), deverá possuir um mecanismo de gerenciamento que coordena as atividades de comunicação com o *SDCD*, com banco de dados e arquivos. Este mesmo gerenciador envia informações do processo para a estimação de estados, onde também são realizadas a reconciliação de dados e detecção de erros grosseiros, e atualização dos parâmetros do modelo do processo. Sendo que suas informações geradas são retornadas ao gerenciador. Com estas informações, o gerenciador efetua o disparo da execução do otimizador através da verificação do estado da planta e das condições da rodada anterior. O otimizador dinâmico resolve o *DAOP* em questão com as condições iniciais e parâmetros atualizados e retorna os resultados para o gerenciador. O gerenciador então analisa os resultados e implementa a receita ótima

através de um sistema de seqüenciamento de bateladas, escrevendo de tempos em tempo o ponto ótimo a ser perseguido pelo sistema de controle.

Para atender a esta estrutura, propõe-se aqui a seguinte arquitetura para o sistema. Ela contempla uma aplicação de otimização *offline* e um conjunto de aplicações *online* e em tempo real. Há uma série de funções que são comuns à otimização *offline* e *online*. A seguir serão apresentadas as estruturas de ambos os sistemas.

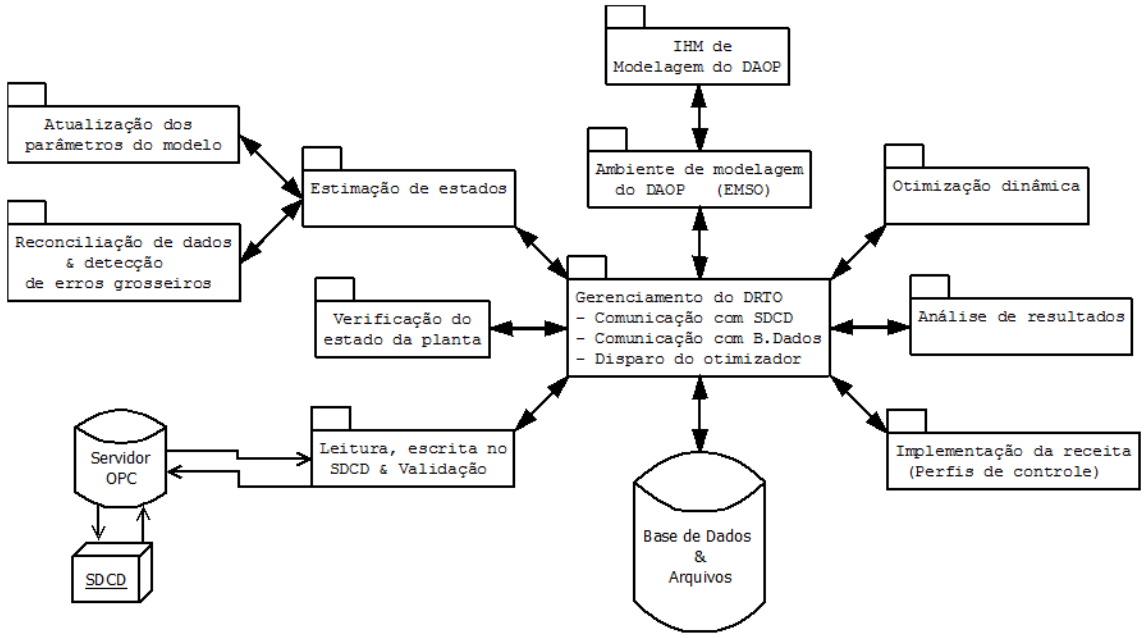


Figura 4.5 – Esquema geral das funcionalidades do *DRTO*.

A aplicação de otimização *offline* é um programa executável que tem as seguintes funcionalidades: simulação no ambiente *EMSO*, métodos de solução de *DAOP*, estimação de estados e calibração do modelo do processo. O sistema de *DRTO* consiste de três programas executáveis que realizam o gerenciamento do sistema, a estimação de estados e a otimização do processo.

O estimador *online* obtém as condições iniciais consistentes e os parâmetros do modelo. O gerenciador deverá disparar a estimação e a otimização dinâmica, além de realizar a comunicação com a planta. A aplicação de otimização dinâmica *online* é semelhante à ferramenta *offline*. A estrutura detalhada do sistema é descrito no apêndice C.

4.1.2.1 Construção do modelo de otimização dinâmica

A construção de modelos de otimização, simulação e otimização dinâmica e operações *offline* são tarefas que estão ligadas ao ambiente de simulação e otimização do *EMSO*. Todas as atividades *offline* são realizadas utilizando um ambiente apropriado, no caso a *IHM* do *EMSO*. O usuário deverá construir o modelo de otimização dinâmica, realizar a calibração inicial do modelo construído e transportar este modelo validado para uso no ambiente *online*.

A *IHM* (vide Figura 4.6) é uma interface que contém um *menu* de opções de operações com o otimizador, uma janela de navegação dos arquivos e módulos do modelo em questão (na forma de árvore de diretórios), uma janela de navegação no resultados do otimizador (na mesma forma de árvore de diretórios), uma janela de edição do modelo e fluxograma (tanto na forma textual quanto na forma gráfica). Além disso, possui janelas de comunicação com as informações de saída do otimizador e outra de visualização gráfica dos resultados.

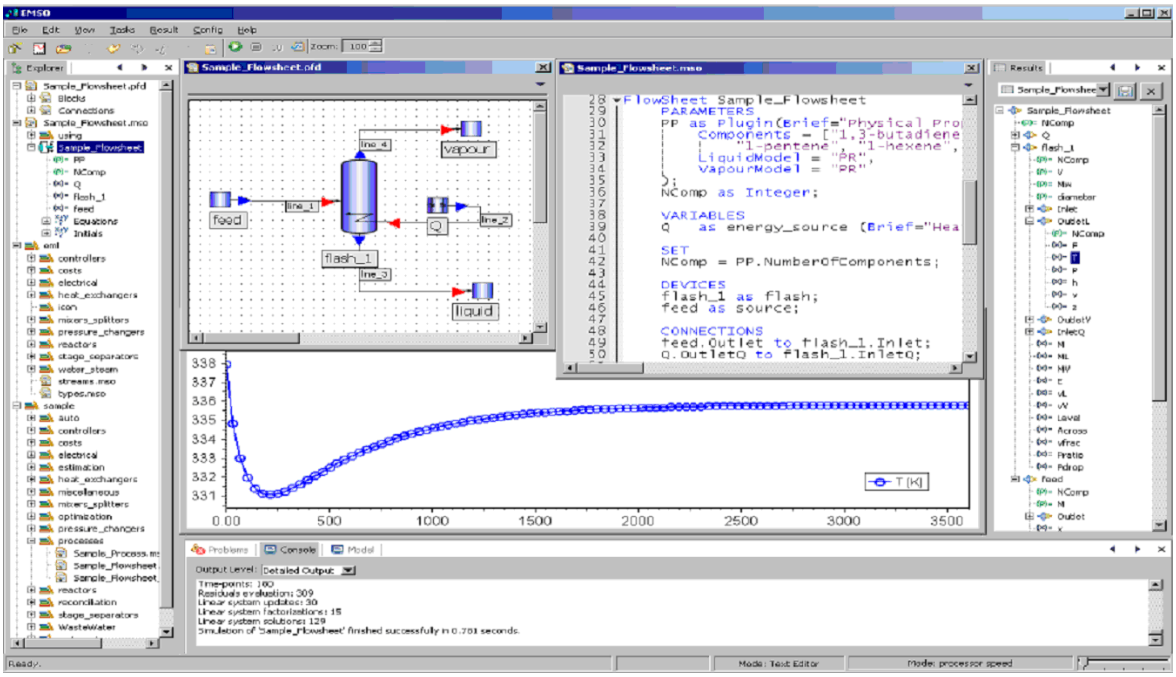


Figura 4.6 – Layout da *IHM* do *EMSO*.

A primeira atividade a ser executada é a construção do modelo de otimização dinâmica. Para formular, é necessário construir o modelo do comportamento dinâmico do processo e o problema de otimização dinâmica usando uma linguagem própria (no caso *EML – EMSO Modeling Language*). Este modelo contém as seções indicadas na Figura 4.7, de acordo com a sintaxe do *EML*.

O modelo do processo deve ser calibrado com dados reais da planta. Na construção do *DAOP*, deve-se estabelecer a estrutura do problema de otimização (função objetivo, restrições e variáveis de controle) e definir os valores para as restrições. Uma vez construído o problema, deve-se estabelecer as condições iniciais (consistentes) do processo, bem como os perfis iniciais das variáveis de controle. Com isso, o problema está pronto para ser resolvido. Municiado de um algoritmo de solução de *DAOP* e de *NLP*, resolve-se o problema de otimização dinâmica e em seguida, apresentam-se os resultados do otimizador num ambiente apropriado, no caso a *IHM* do *EMSO*.

Para construir o problema de otimização dinâmica, deve-se escrever o modelo dinâmico do processo e o problema de otimização dinâmica. O modelo dinâmico do processo é construído escrevendo o modelo matemático de cada operação unitária e colocando em um fluxograma do processo. O problema de otimização dinâmica pode ser descrito na forma de problema de simples e múltiplos estágios. No problema multi-estágios em cada estágio são escritas as definições específicas do problema de otimização. Além disso, devem ser

estabelecidas as condições de junções dos estágios. Este modelo também tem definições globais, que são comuns a todos os estágios. Se o problema for de simples estágio, todas as definições do problema de otimização são globais. De acordo com a formulação padrão do *DAOP*, para descrever o problema de otimização dinâmica é necessário definir e parametrizar as variáveis de controle do problema e a manipulação do tempo final. É necessário também descrever a função objetivo e restrições, como também as definições dos métodos de otimização utilizados.

```
#*
 Series of CSTRs with PI Controller
*#

using "Types";

ConcInt as Real(Unit='kmol/m^3*h',
                Default=0,Lower=-20,Upper=20);

Model CSTR

PARAMETERS
  tau as time_h;
  k   as frequency;

VARIABLES
  Ca     as conc_mol;
  Ca_in  as conc_mol;
  Ca_out as conc_mol;

EQUATIONS

  diff(Ca) = (Ca_in - Ca) / tau - k * Ca;
  Ca_out = Ca;

end # CSTR


Model PI

PARAMETERS
  Kc as positive;
  Ti as time_h;

VARIABLES
  Cm        as conc_mol;
  Cmss      as conc_mol;
  Set_point as conc_mol;
  e         as conc_mol(Lower=-20);
  w         as ConcInt;

EQUATIONS

  Cm = Cmss + Kc * (e + w / Ti);
  diff(w) = e;

end # PI
```

```
FlowSheet SERIES
 DEVICES
  CSTR1 as CSTR;
  CSTR2 as CSTR;
  CSTR3 as CSTR;
  CONTROLLER as PI;

 VARIABLES
  Ca_d as conc_mol;

 SET
  CSTR1.tau = 2 * 'h';
  CSTR2.tau = 2 * 'h';
  CSTR3.tau = 2 * 'h';
  CSTR1.k = 0.5 * '1/h';
  CSTR2.k = 0.5 * '1/h';
  CSTR3.k = 0.5 * '1/h';
  CONTROLLER.Kc = 10;
  CONTROLLER.Ti = 4 * 'h';

 EQUATIONS
  CONTROLLER.Cm + Ca_d = CSTR1.Ca_in;
  CSTR1.Ca_out = CSTR2.Ca_in;
  CSTR2.Ca_out = CSTR3.Ca_in;
  CONTROLLER.e = CONTROLLER.Set_point - CSTR3.Ca_out;

  if time < 5 * 'h' then
    Ca_d = 0.4 * 'kmol/m^3';
  else
    Ca_d = 0.6 * 'kmol/m^3';
  end
  if time < 50 * 'h' then
    CONTROLLER.Set_point = 0.1 * 'kmol/m^3';
  else
    CONTROLLER.Set_point = 0.15 * 'kmol/m^3';
  end

 SPECIFY
  CONTROLLER.Cmss = 0.4 * 'kmol/m^3';

 INITIAL
  CSTR1.Ca = 0.4 * 'kmol/m^3';
  CSTR2.Ca = 0.2 * 'kmol/m^3';
  CSTR3.Ca = 0.1 * 'kmol/m^3';
  CONTROLLER.w = 0 * 'h*kmol/m^3';

 OPTIONS
    TimeStart = 0;
    TimeStep = 0.5;
    TimeEnd = 100;
    TimeUnit = 'h';
end
```

Figura 4.7 – Sintaxe da *EML* para modelagem dinâmica de processo (Soares, 2003).

O usuário deve representar o modelo de otimização de forma estruturada. Neste caso, é necessário que seja definida uma sintaxe de construção do *DAOP* e projetar um interpretador das funções *EML* adicionais. Para definir as características do interpretador da *EML* para o modelo de otimização dinâmica, propõem-se algumas sintaxes adicionais para o sistema, onde incluem: as definições: das variáveis de controle, de tempos finais (para problemas de tempo final é livre), da função objetivo, das restrições e das opções do método de solução (dos métodos de solução de *DAOP*, de *NLP*, de adaptação de malhas e detecção de estrutura da solução). Além disso, podem-se incluir as definições das equações adicionais do modelos de otimização e as definições dos estágios do *DAOP* (problema com único estágio ou definições globais e problema com múltiplos estágios). A proposta de sintaxe da formulação do *DAOP* é apresentada no apêndice D.

Interpretação da sintaxe do modelo e construção do DAOP padrão

Na aplicação de otimização, ao selecionar um problema a ser resolvido, pode realizar as tarefas de resolver um problema de otimização (na seção `DAOPOptimization`), calibrar o modelo construído (na seção `Estimation`), reconciliar as medidas (na seção `Reconciliation`), realizar estudos de casos (na seção `DAOPCaseStudy`) ou verificar a consistência da tarefa em questão (pela função `Check Consistency`). Na otimização *offline,* pode-se executar uma tarefa (pela função `Run`), fazer uma pausa na tarefa ao final de uma

iteração (pela função `Pause`) e inspecionar as informações intermediárias, interromper a execução da tarefa (pela função `Stop`), como mostra a Figura 4.8. Ao executar as tarefas acima citadas, a aplicação primeiramente expande a seção, interpreta os comandos do texto e verifica a consistência do problema a ser resolvido.

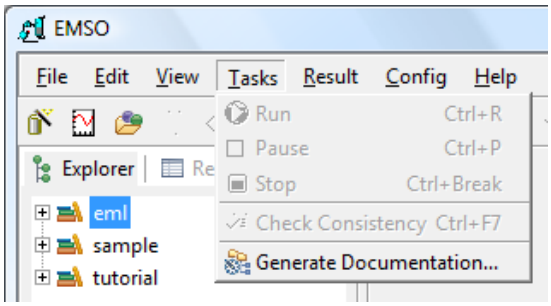


Figura 4.8 – operações do *EMSO*.

No caso das tarefas de solução de um problema de otimização (na seção `DAOPOptimization`) e da realização de estudos de casos (na seção `DAOPCaseStudy`), o otimizador deverá reformular o problema de otimização para um *DAOP* na forma padrão. Neste caso, todas as expressões que estiverem nas funções objetivo e não forem variáveis de estado, deverão ser transformadas em equações diferenciais ou algébricas.

Além disso, as variáveis de controle podem ter limites de velocidades. Estas variáveis de controle são redefinidas como variáveis de estado com perfis lineares por partes contínuas e as variável a serem manipuladas pelo otimizador terão suas taxas de variações constantes por partes. Isto equivale a reformular o *DAOP* padrão da seguinte forma:

$$
\begin{aligned}
&\underset{u,t_f}{Min} \quad \phi\left(x\left(t_f\right), y\left(t_f\right), t_f\right) \\
&s.a. \\
&\qquad F\left(\dot{x}, x, y, u, t\right) = 0; \quad x\left(0\right) = x_0 \\
&\qquad \frac{du}{dt} = w; \qquad\qquad u\left(0\right) = u_0 \\
&\qquad x_L^P \le x^P\left(t\right) \le x_U^P \\
&\qquad y_L^P \le y^P\left(t\right) \le y_U^P \\
&\qquad u_L^P \le u^P\left(t\right) \le u_U^P \\
&\qquad x_L^E \le x^E\left(t_f\right) \le x_U^E \\
&\qquad y_L^E \le y^E\left(t_f\right) \le y_U^E \\
&\qquad u_L^E \le u^E\left(t_f\right) \le u_U^E \\
&\qquad w_L \le w\left(t\right) \le w_U
\end{aligned}
\tag{4.2}
$$

Na realidade não há motivações, em termos industriais, de fazer algo diferente disso, pois esta estratégia garante a segurança da operação industrial e suaviza a solução do otimizador. O ponto negativo desta reformulação é o aumento do número de equações do sistema a ser otimizado. Vale lembrar que, se o usuário optar originalmente por um perfil

linear por partes contínuo (*PW_LIN_C*), o mesmo número de equações será acrescido no sistema.

Além das variáveis de controle, as definições de tempos finais livres e de expressões de desigualdades também podem ser reformulados. As implementações das sintaxes destas reformulações podem ser vistas no apêndice E

Interpretação da sintaxe para a utilização de arquivos na otimização

Um aspecto importante de um sistema de *DRTO* é a sua capacidade de reproduzir, no modo *offline*, qualquer situação vivida na otimização *online*. As aplicações de otimização em tempo real, que se tenha conhecimento, não têm a capacidade de reproduzir um caso real sem algum esforço adicional de modelagem no modo *offline*. Portanto, propõe-se aqui uma infra-estrutura de utiliza arquivos de dados, onde alguns dados de entrada são fornecidos pelo otimizador em tempo real (pode-se inclusive alterá-los) e são sobrescritas no *DAOP* original algumas mudança que ocorreram em tempo real em relação ao arquivo de modelo de otimização (*DAOP.mso*). Além disso, os dados de entradas da otimização *online* (condições iniciais, parâmetros, *biases* de reconciliação) são fornecidos pelos mesmos arquivos. Os detalhes das estruturas dos arquivos são apresentados no apêndice F.

Outra facilidade importante é a capacidade de visualizar e verificar a solução obtida pelo otimizador *online*. Em qualquer momento pode-se executar o otimizador *offline* no caso SIMULAÇÃO, para analisar em detalhes o resultado do otimizador. Neste caso, agrega-se o arquivo de perfis de controle da solução ótima, obtida pelo otimizador *online*. Em alguns momentos pode ser conveniente que o otimizador *offline* exporte alguns dados (ex.: parâmetros do modelo) em uma situação particular. O otimizador *online* exporta sistematicamente seus dados para serem usados no modo *offline*.

Para que o otimizador importe os exporte cada um destes arquivos, é necessário estabelecer a instrução na área de opções de cada seção do modelo de otimização (vide apêndice F para verificar a sintaxe implementada).

Há um conjunto de diretivas que complementa o modelo de otimização dinâmica. Estas diretivas são seções de modelagem que tem objetivos específicos de modelagem da calibração do modelo do processo, de detecção de erros grosseiros e reconciliação de dados da planta, e de estudos de casos de otimização dinâmica (Soares, 2007).

Seção de calibração do modelo e reconciliação de dados das medidas

Para que seja possível realizar a calibração do modelo do processo, é necessário editar através da *IHM* do *EMSO* as instruções de calibração. Na calibração do modelo é necessário definir quais parâmetros precisam ser determinados e seus domínios. Além disso, é necessário fornecer os dados experimentais e informações de suas variâncias. Também é necessário definir o método de ajuste. A formulação da estimação de parâmetros para *DRTO* é a mesma existente atualmente no *EMSO* e a sintaxe da seção `Estimation` é a mesma utilizada neste software. A diferença está na forma de resolver o problema (será discutido mais adiante).

Da mesma forma que na calibração do modelo, para que seja possível realizar a reconciliação de dados das medições do processo, é necessário editar através da *IHM* do *EMSO* as instruções de reconciliação. Na reconciliação de dados é necessário definir quais variáveis precisam ser reconciliadas e quais estão livres para serem alteradas. Além disso, é necessário fornecer os dados experimentais e informações de suas variâncias. A sintaxe da seção `Reconciliation` tem vários itens em comum com a seção `Estimation`, e é a mesma utilizada atualmente no *EMSO*. Da mesma forma da calibração do modelo, a diferença está no método de solução do problema de otimização.

Pelo fato da estratégia de solução e de apresentação de resultados serem diferentes, comandos novos estão localizados na área de `OPTIONS` (vide a sintaxe no apêndice G). Nesta área, o usuário pode definir os arquivos para exportação das informações de reconciliação de dados, do ajuste de parâmetros, e das condições iniciais da estimação de estados (conforme descrito acima).

As opções são definidas para utilizar os métodos de solução de estimação de parâmetros, reconciliação de dados e estimação de estados. Nesta seção define-se o método de estimação utilizado, a estratégia de estimação a ser adotada e seus parâmetros de sintonia.

Os métodos considerados são:
CEKF - Filtro de Kalman estendido com restrições;
MHE - Estimador de horizonte móvel.

Seção do modelo de estudo de casos

No estudo de casos do *DAOP*, diferentemente do estudo de casos convencional do *EMSO*, procura-se resolver um caso da *DAOP* de cada vez. O estudo de casos procura explorar as proposições alternativas para o *DAOP*, tanto para resolver problemas encontrados em rodadas anteriores com para a busca de novas oportunidades de otimização. Da mesma forma que na otimização do processo, para que seja possível realizar os estudos de casos, é necessário editar através da *IHM* do *EMSO* as instruções destes estudos. No estudo de casos é necessário definir apenas as alterações que deverão ser feitas no problema de otimização. Estas alterações provocarão mudanças na estrutura do *DAOP* ou apenas alteram as posições das restrições de controle e estado, função objetivo, ou algoritmos de solução.

Esta seção pode ser utilizada para fazer alguns estudos em particular. Há quatro casos especiais de uso do otimizador, são eles: visualização, viabilidade, otimalidade e flexibilidade. Estes casos têm objetivos específicos de verificar e diagnosticar rodadas do otimizador *online*.

O **caso visualização** consiste em verificar os resultados de uma determinada rodada sem executar novamente o algoritmo de otimização (vide a sintaxe no apêndice G). A grande utilidade deste caso e a possibilidade de examinar exaustivamente os resultados de qualquer rodada do otimizador *online*.

O **caso viabilidade** consiste em remover a função objetivo do *DAOP* em estudo (vide a sintaxe no apêndice G). O objetivo deste caso é permitir a verificação da viabilidade do problema sem objetivo de otimização, isto é, o caso regulatório.

O **caso otimalidade** consiste em remover as restrições que não descaracterize a operação do processo sem otimização (vide a sintaxe no apêndice G). O objetivo deste caso é verificar a possibilidade de melhorar a função objetivo pela remoção de restrições, isto fornece um valor potencial da função objetivo.

O **caso flexibilidade** consiste em resolver inviabilidades do *DAOP* através da flexibilização de restrições. Este caso permite diagnosticar tal inviabilidade e identificar restrições problemáticas, bem como quantificar a sua solução (vide a sintaxe no apêndice G). Uma forma eficiente de diagnosticar e resolver estas inviabilidades é flexibilizar as restrições conforme o método proposto na parte 2 desta tese. As definições das restrições rígidas de um *DAOP* são flexibilizadas conforme a descrição dos atributos adicionais das seções MINIMIZE/MAXIMIZE, CONTROL e CONSTRAINTS, que definem as propriedades das restrições do problema.

Neste caso, o interpretador irá reformular o *DAOP* original para a forma da Equação 4.1. O mesmo redefine a função objetivo como $\gamma(t_f)$, cria uma nova variável diferencial γ, acrescenta a equação $d\gamma/dt = \Delta\gamma$ com $\gamma(t_0) = 0$, e cria uma nova variável de controle $\Delta\gamma$. O interpretador também acrescenta variáveis de folga (s^x, s^y, s^u) conforme descrito na seção de restrições do estudo de casos. Também são acrescentadas as restrições ψ^{ϕ}, ψ^{x}, ψ^{y} e ψ^{u} com os limites $\psi^{\phi} \geq 0$, $\psi^{x} \geq 0$, $\psi^{y} \geq 0$ e $\psi^{u} \geq 0$ e as restrições g^{x}, g^{y}, g^{u} conforme a Equação 4.3. Além disso, as restrições originais, que foram relaxadas, passam a ser os limites das variáveis g^{x}, g^{y} e g^{u}. Os `<Relaxamento Máximo Inferior>` (s_x^L, s_y^L, s_u^L), os `<Relaxamento Máximo Superior>` (s_x^L, s_y^L, s_u^L) e `<Variância da restrição>` (σ_x, σ_y e σ_u) referentes às variáveis de folga são descritos como atributos adicionais das restrições, conforme apresentado acima.

$$\min_{u,\Delta\gamma,s^x,s^y,s^u,p,t_i} \gamma(t_f)$$

s.a.

$$F(\dot{x}(t), x(t), y(t), u(t), p, t) = 0; \qquad x(t_f) = x_0 \quad t \in [t_0, t_f]$$

$$\frac{d\gamma}{dt} = \Delta\gamma(t) \qquad \gamma(t_0) = 0$$

$$\psi^{\phi}(t_f) = \gamma(t_f) w_0 - \phi(x(t_f)) + \phi^L$$

$$\psi_i{}^x(t) = \Delta\gamma(t) - \left(\frac{s^x{}_i(t)}{\sigma^x{}_i}\right)^2 \qquad onde \quad i \in S_x = \{1, \cdots, nx\}$$

$$\psi_j{}^y(t) = \Delta\gamma(t) - \left(\frac{s^y{}_j(t)}{\sigma^y{}_j}\right)^2 \qquad onde \quad j \in S_y = \{1, \cdots, ny\}$$

$$\psi_k{}^u(t) = \Delta\gamma(t) - \left(\frac{s^u{}_k(t)}{\sigma^u{}_k}\right)^2 \qquad onde \quad i \in S_u = \{1, \cdots, nu\}$$

$$g^x(t) = x(t) + s^x(t)$$

$$g^y(t) = y(t) + s^y(t)$$

$$g^u(t) = u(t) + s^u(t)$$

$$x^L \le g^x(t) \le x^U \qquad onde \quad g^x(t) \in \Re^{S_x}$$

$$y^L \le g^y(t) \le y^U \qquad onde \quad g^y(t) \in \Re^{S_y}$$

$$u^L \le g^u(t) \le u^U \qquad onde \quad g^u(t) \in \Re^{S_u} \qquad (4.3)$$

$$s_L^x \le s^x(t) \le s_U^x \qquad onde \quad s^x(t) \in \Re^{S_x}$$

$$s_L^y \le s^y(t) \le s_U^y \qquad onde \quad s^y(t) \in \Re^{S_y}$$

$$s_L^u \le s^u(t) \le s_U^u \qquad onde \quad s^u(t) \in \Re^{S_u}$$

$$\psi^{\phi}(t) \ge 0; \quad \psi^x(t) \ge 0; \quad \psi^y(t) \ge 0; \quad \psi^u(t) \ge 0;$$

4.1.2.2 Visualização das saídas do otimizador

Quanto à visualização das saídas do otimizador, as informações de cada iteração do otimizador, os perfis de estados e controles da solução obtida e as informações de desempenho e estado final da otimização são escritas no arquivo de saída *DAOP.diag* e *NLPSolver.out*. Estas informações serão interpretadas e processadas posteriormente pelas aplicações de monitoração e diagnóstico do *DRTO*. A visualização dos resultados é apresentada na mesma forma atual do *EMSO*: visualização em árvore e no console de saídas. É relevante chamar a atenção para este mecanismo, pois permite que a solução do *DAOP* obtido pelo otimizador *online* seja visualizada e analisada no ambiente *offline*, sem qualquer esforço adicional de programação e manipulação de arquivos.

Na visualização em árvore, tem-se uma estrutura de variáveis de cada seção do modelo de otimização. Os detalhes da implementação desta visualização podem ser observados do apêndice H.

As informações de saída do otimizador (do algoritmo de *NLP* e *DAOP* - ex.: resumo dos indicadores da iteração), a cada iteração, podem ser visualizadas através da janela de console de saída do *EMSO*. Esta pode apresentar os resultados das iterações na forma de rolagem ou pausadamente. Se o usuário desejar visualizar as saídas de uma determinada iteração, o mesmo deve comandar a pausa no *menu* de tarefas (Figura 4.8). Neste momento deverão ser mostradas as informações da iteração em questão. Ao final da execução do otimizador, o mesmo apresenta, no console de saída, as informações do final da otimização e de desempenho do otimizador. Os itens apresentados no console são diferentes para cada algoritmo utilizado pelo otimizador.

4.1.2.3 Atualização do estado e modelo de otimização dinâmica

Este processo consiste na estimação de estados e ajuste de parâmetros do modelo. Esta atividade consiste na obtenção de dados históricos da planta e execução da estimação de estados, onde são obtidos todos os valores dos estados corrigidos e dos parâmetros do modelo do processo, como mostra a Figura 4.9. Nesta etapa também pode ser realizada a detecção de erros grosseiros e reconciliação de dados de medição da planta, Além disso, são obtidas também as estatísticas dessa estimação. Na etapa de reconciliação se obtém os desvios (*biases*) das variáveis medidas, para uso posterior nas etapas de implementação das ações de controle ótimo e de diagnóstico da reconciliação. Estas informações são então depositadas em banco de dados ou arquivo (*InitConditions.dat*, *ModelParameters.dat* e *DataReconciliation.dat*). Uma vez realizada esta etapa, obtém-se as condições iniciais consistentes e os parâmetros do modelo atualizados. Desta forma pode ser diretamente utilizado pelo otimizador *online* ou *offline*. Resumindo, o módulo de atualização do modelo e estado da planta deve possuir as seguintes facilidades:

- Ajuste e validação de parâmetros do modelo do processo;
- Estimação de estados do processo;
- Reconciliação de dados da planta;
- Exportação das condições iniciais, parâmetros e dados reconciliados.

A estimação de estados da planta é uma tarefa realizada tanto pela aplicação de otimização *offline "EMSO_DYNOPT.EXE"* como pela aplicação *online "EMSO_SE.EXE"*. Ambas as aplicações utilizam a biblioteca *"SE_SOLVER.DLL"*, que é responsável por realizar as tarefas de acima mencionadas (vide apêndice C). A reconciliação de dados é realizada com instruções contidas na seção `Reconciliation` do modelo construído, o ajuste de parâmetros na seção `Estimation`, e a estimação de estados.

A estimação *online* tem como ponto positivo a recorrência de informações que pode fornecer informações importantes para monitorar e diagnosticar a qualidade da estimação. A repetição deste processo fornece informações importantes sobre a tendência do processo, dos parâmetros do modelo e dos *biases* dos instrumentos. Como ponto negativo, a estimação *online* está mais sujeito às perturbações nos dados do processo, dificultando a estimação de estados. Neste caso, tem-se um cuidado adicional de monitorar os *biases* gerados na reconciliação, pois serão importantes na análise de consistência da reconciliação e também utilizados na atualização das condições iniciais do otimizador, quando não houver uma reconciliação no momento do disparo do otimizador. Será uma reconciliação aproximada utilizando os *biases* das medições detectados na última reconciliação válida.

Lembre que tanto a calibração do modelo quanto a estimação de estado são efetuadas com informações dinâmicas do processo, portanto a planta deverá ser perturbada durante a obtenção de dados experimentais. A freqüência de estimação de estados, ajuste de parâmetros e reconciliação de dados pode ser a mesma ou diferentes. Esta decisão irá depender da freqüência de desajuste do modelo e dos instrumentos.

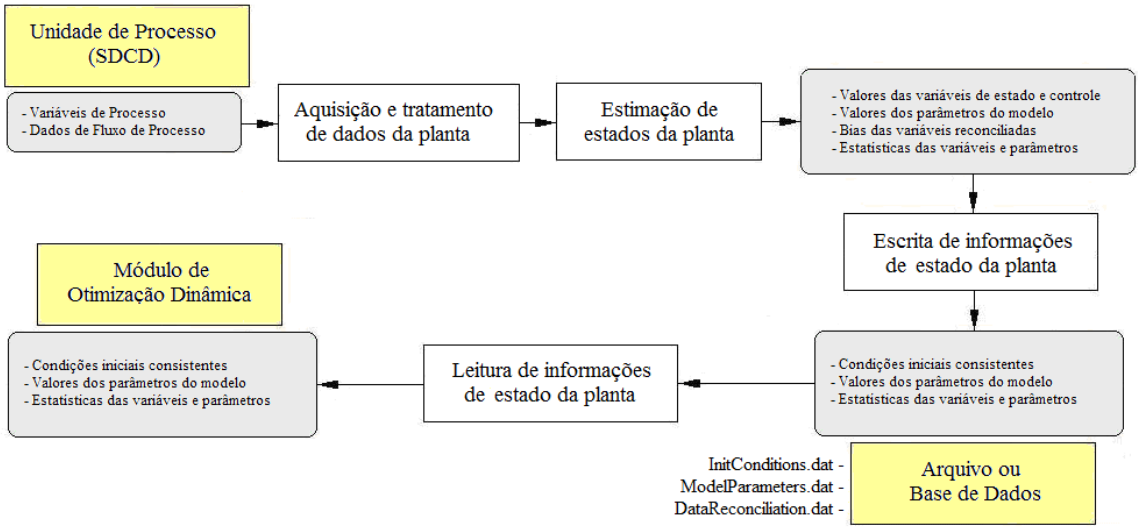


Figura 4.9 – Processo de obtenção das condições iniciais e parâmetros do modelo.

Um modelo dinâmico contém dois tipos de parâmetros, são: os parâmetros fixos, que são normalmente constantes físicas e dimensionais; e os parâmetros variantes no tempo e no estado, que são normalmente relacionados a índices de desempenho. Além disso, há parâmetros que estão relacionados ao comportamento estático, que podem ser ajustados com dados do estado estacionário do processo, e parâmetros relacionados ao comportamento dinâmico do processo. Por simplicidade de representação, iremos considerar que os parâmetros fixos estão embutidos no modelo e não são considerados parâmetros a serem ajustados com dados da planta.

Os parâmetros estáticos dizem respeito a aqueles que usualmente mudam lentamente durante o correr do processo (usualmente dias). Portanto, estes parâmetros podem ser estimados utilizando somente dados estáticos. Esta escolha deve ser feita de acordo com a conveniência da pessoa responsável pela manutenção do modelo de otimização. Caso decida por esta estratégia, é necessário que se detecte os estado estacionário da planta e se estime somente este subconjunto de parâmetros. Às vezes os parâmetros podem mudar rapidamente. Os parâmetros deste tipo consistem de constantes de dinâmica de processo, bem como de parâmetros estáticos que mudam com freqüência (ex.: atividade de catalisadores ou fatores de sujeira em trocadores de calor, em alguns casos). Neste caso, deve-se fazer a atualização de um subconjunto de parâmetros que necessitam de dados dinâmicos do processo para serem estimados.

Nem sempre a determinação de todos os parâmetros do modelo se mostra uma estratégia adequada, já que há a possibilidade de termos interação entre parâmetros ou até mesmo um mau condicionamento do problema a ser resolvido. Para minimizar este problema, pode-se

utilizar o algoritmo *SELEST* (Secchi *et al.*, 2006), que selecionar o melhor conjunto de parâmetros a serem determinados, baseado em suas identificabilidades, a partir do conjunto de todos os parâmetros do modelo. Este algoritmo utiliza uma matriz baseada sensibilidade das saídas com os parâmetros e na independência entre eles, tal como proposto por Li *et al.* (2004). Um índice de degradação de previsibilidade e um índice de degradação da correlação dos parâmetros são utilizados como critérios de parada. Uma vez selecionado o conjunto de parâmetros a serem estimados, então os dados da planta são coletados para executar a calibração do modelo. Podem-se utilizar técnicas de planejamento de experimentos ou não. A etapa seguinte é a determinação de parâmetros propriamente dita.

Para realizar a estimação de estados e o ajuste dos parâmetros de um modelo dinâmico, é preciso coletar uma série temporal de dados de forma a capturar informações do comportamento dinâmico do processo. Esta série temporal é utilizada tanto para o ajuste de parâmetros, quanto para a reconciliação de dados e estimação de estados. Diferentemente do caso estacionário, não necessita que se aguarde a planta atingir o estado estacionário para estimar os estados, podendo ser realizada a cada ciclo de amostragem de dados da planta.

Na estimação, tem-se uma janela móvel de horizonte *N* do *MHE* ou maior. Recomenda-se que sejam guardadas informações da planta com numa freqüência fixa de 10% do tempo de estabilização da variável de estado de dinâmica mais rápida e 10% tempo de estabilização da variável de estado de dinâmica mais lenta. No momento de realizar a estimação, devem-se re-amostrar estas duas series temporais e compor a série que fará a estimação desejada. Esta nova serie não precisa ter intervalos regulares, apenas precisa conter a quantidade e distribuição adequada dos pontos do processo de forma a capturar os comportamentos dinâmicos e estáticos do processo.

Se tiver os dados reconciliados e os parâmetros estimados confiáveis e atualizados, então basta resolver o sistema *NLA* usando as funcionalidades do *EMSO*. Isto será feito pelo gerenciador do *DRTO*, mediante a integração do modelo a partir do último estado estimado válido até o instante atual e solução do sistema *NLA*.

Se não tivermos os dados reconciliados e/ou parâmetros atualizados, pode-se utilizar filtros de Kalman estendidos com restrições (*CEKF – Constrained Extended Kalman Filter* – Gesthuisen et al., 2001), o estimador de horizonte móvel (*MHE – Moving Horizon Estimator*) considerado por Rao et al. (2001 e 2003) e Abu-el-zeet et al. (2002) ou até mesmo o próprio algoritmo de otimização dinâmica. No primeiro caso, obtém as condições iniciais consistentes, e no segundo caso, são obtidas as condições iniciais, os dados reconciliados e até mesmo os parâmetros atualizados. A implementação do estimador é representada na Figura 4.10, onde a solução é obtida por um processo de otimização, com uma seqüência de predições usando o modelo do processo e correções com dados experimentais. A cada estimação a matriz de covariância dos estados é atualizada.

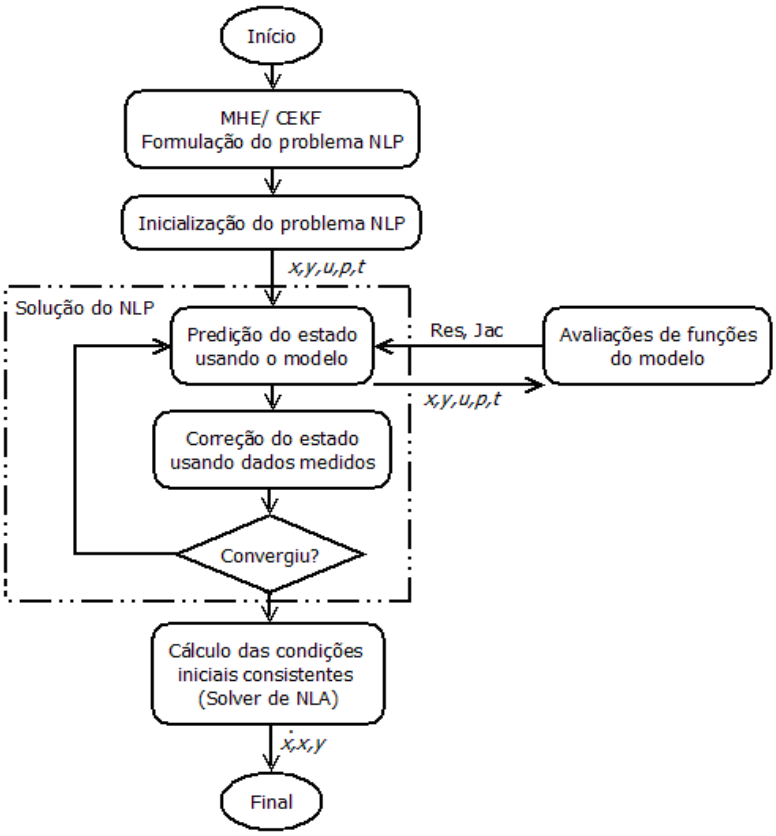


Figura 4.10 – Algoritmo de estimação de estados.

Para realizar a estimação utilizando a abordagem do *CEKF*, pode-se utilizar o algoritmo do *MHE* com horizonte zero (*N = 0*). Os algoritmos *CEKF* e *MHE* podem ser formulados como um problema de otimização dinâmica com *N* estágios ou um *DAOP* discretizado em *N* elementos finitos. Para resolver o problema de estimação, é necessário somente mudar a formulação do problema de otimização dinâmica para a seguinte forma:

Para a estimação do estado:

$$\min_{\omega(t_i),\nu(t_i)} \quad \psi = \omega(t_0)^T P^{-1}\omega(t_0) + \sum_{i=1}^{N} \omega(t_i)^T Q^{-1}\omega(t_i) + \nu(t_i)^T R^{-1}\nu(t_i) \quad i = 1,\ldots,N$$

s.a.

$$\begin{aligned}
&\dot{x} = f(x,u,p,t) + \omega(t), \quad x(0) = x_0 \\
&y_i^{\exp} = h(x,u,p,t_i) + \nu(t_i) \quad i = 1,\ldots,N \\
&x_{\min} \le x(t) \le x_{\max} \\
&u_{\min} \le u(t) \le u_{\max} \\
&\omega_{\min} \le \omega(t) \le \omega_{\max} \\
&\nu_{\min} \le \nu(t_i) \le \nu_{\max}
\end{aligned} \tag{4.4}$$

Para atualização do estado:

$$\dot{x} = f\left(x,u,p,t\right) + \omega^{*}\left(t\right),\ x\left(0\right) = x_0$$
$$\dot{P} = f_x P + P f_x - P h_x^{\ T} R^{-1} h_x P + Q \ \ onde\ \ f_x = \partial f \big/ \partial x \ \ e\ \ h_x = \partial h \big/ \partial x \qquad i = 1,\ldots,N$$
$$y = h\left(x,u,p,t\right) + \nu^{*}\left(t\right)$$

Onde $\omega(t)$ é o ruído de processo e $v(t_i)$ o erro de medida no instante t_i, que são considerados como sendo aleatórios gaussianos e brancos com média zero e covariância $Q(t)$ e $R(t)$, respectivamente. P é a matriz de covariância dos estados.

Desta forma, pode-se resolver o problema de reconciliação de dados, de estimação de parâmetros e estimação de estados usando as abordagens *CEKF*, *MHE* como sendo um problema de otimização dinâmica reformulado. Com isso, podem-se utilizar os algoritmos de solução do problema de otimização dinâmica *single-shooting*, *multi-shooting* e colocação em elementos finitos para resolver a estimação de estados. Propõe-se aqui a utilização de uma formulação única de *DAOP* de múltiplos estágios do *MHE*, para resolver os problemas de estimação de parâmetros, reconciliação de dados e estimação de estados. O número de estágios é o horizonte do *MHE* (*N*). Nesta formulação, cada variável medida (y^{exp}) é considerada como uma variável de controle, que não está livre para ser otimizada, que permanece constante entre duas amostragens do filtro. Isto remete a perfis constantes por partes das variáveis medidas. Se desejar utilizar o *CEKF*, basta definir o problema de otimização como simples estágios, onde a amostragem em k-1 são as condições iniciais do *DAOP* e a estimação em k é o tempo final do problema de otimização.

O que diferença da estimação de estados, da reconciliação e estimação de parâmetros é a escolha das variáveis de controle livres para serem otimizadas. Na estimação de estados as variáveis de decisão são ω e ν, na reconciliação é ν, e no ajuste de parâmetros p. Sendo que ω e ν são variáveis de controle constantes por partes e p parâmetros constantes ao longo de todo o horizonte de estimação. Com isso, a formulação única proposta aqui é a seguinte:

$$\min_{\omega_k(t),\nu_k(t),p} \quad \psi_k\left(t_f^{(k)}\right) \qquad k = 1,\ldots,N$$

s.a.

$$\begin{aligned}
&\dot{x}_k = f\left(x_k,u_k,p,t\right) + \omega_k\left(t_k\right), \quad x(0) = x_0 \\
&y_k = h\left(x_k,u_k,p,t\right) + \nu_k\left(t\right) \\
&\dot{\psi}_k = \omega_k\left(t\right)^T Q^{-1} \omega_k\left(t\right) + \nu_k\left(t\right)^T R^{-1} \nu_k\left(t\right), \quad \psi_1\left(0\right) = \omega_0^{\ T} P \omega_0 \\
&p_k = p_{k-1} \\
&x_{\min} \le x_k\left(t\right) \le x_{\max} \\
&u_{\min} \le u_k\left(t\right) \le u_{\max} \\
&p_{\min} \le p \le p_{\max} \\
&\omega_{\min} \le \omega_k\left(t\right) \le \omega_{\max} \\
&\nu_{\min} \le \nu_k\left(t\right) \le \nu_{\max}
\end{aligned} \tag{4.5}$$

e a atualização do estado é dada por:

$$\dot{x} = f\left(x,u,p,t\right) + \omega^{*}\left(t\right),\ x\left(0\right) = x_0$$
$$\dot{P} = f_x P + P f_x - P h_x^{\ T} R^{-1} h_x P + Q \quad onde \quad f_x = \partial f / \partial x \quad e \quad h_x = \partial h / \partial x$$
$$y = h\left(x,u,p,t\right) + \nu^{*}\left(t\right)$$

Na estimação de estados *online* com o processo, ao utilizar o *MHE* para fazer estimação dos estados, utilizam-se as últimas *N+1* medidas, sendo *N* o tamanho do horizonte de tempo, conforme esquema mostrado na Figura 4.11. A escolha do tamanho do horizonte *N* é um parâmetro de sintonia na *MHE*. Quanto maior *N*, melhor o desempenho da estimação, porem aumenta o custo computacional na obtenção da solução. Desta forma, é preciso conciliar estes dois fatores. Neste caso, sugere-se a utilização do algoritmo de seleção do número mínimo de medidas proposto por Salau em 2009. Este algoritmo é baseado na análise de observabilidade, onde a estabilidade da estimação pelo *MHE* é garantida quando o tamanho do horizonte for maior que o índice de observabilidade do sistema (Rao e Rawlings, 2002). Outro aspecto importante no *MHE* é a forma de incorporação dos efeitos dos dados passados fora do horizonte de estimação como discutido em Salau (2009).

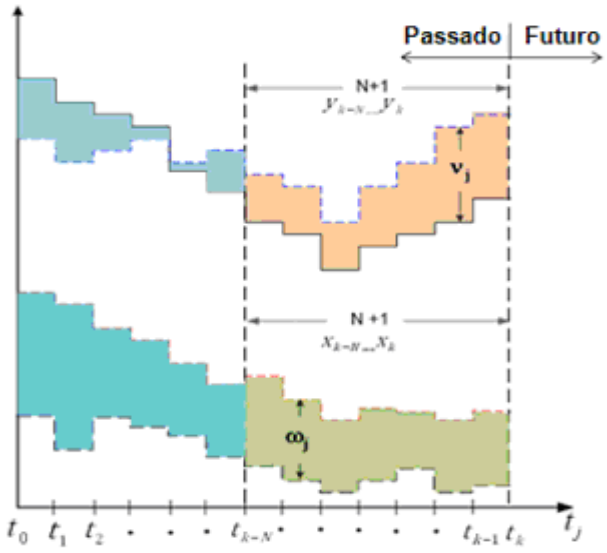


Figura 4.11 – Esquema do estimador de horizonte móvel (Salau, 2009).

As atividades de reconciliação, ajuste de parâmetros e estimação de estados podem ser executadas seqüencialmente ou simultaneamente. Se for seqüencial (Figura 4.12a), deve ser feito um ajuste inicial dos parâmetros e em seguida realizar a seqüência de ajustes: reconciliação de dados, ajuste de parâmetros e estimação de estados. Também pode ser realizada a reconciliação de dados e ajuste de parâmetros simultaneamente e estimação de estados na seqüência (Figura 4.12b). Outra opção é realizar reconciliação, ajuste de parâmetros e estimação de estados simultaneamente (Figura 4.12c).

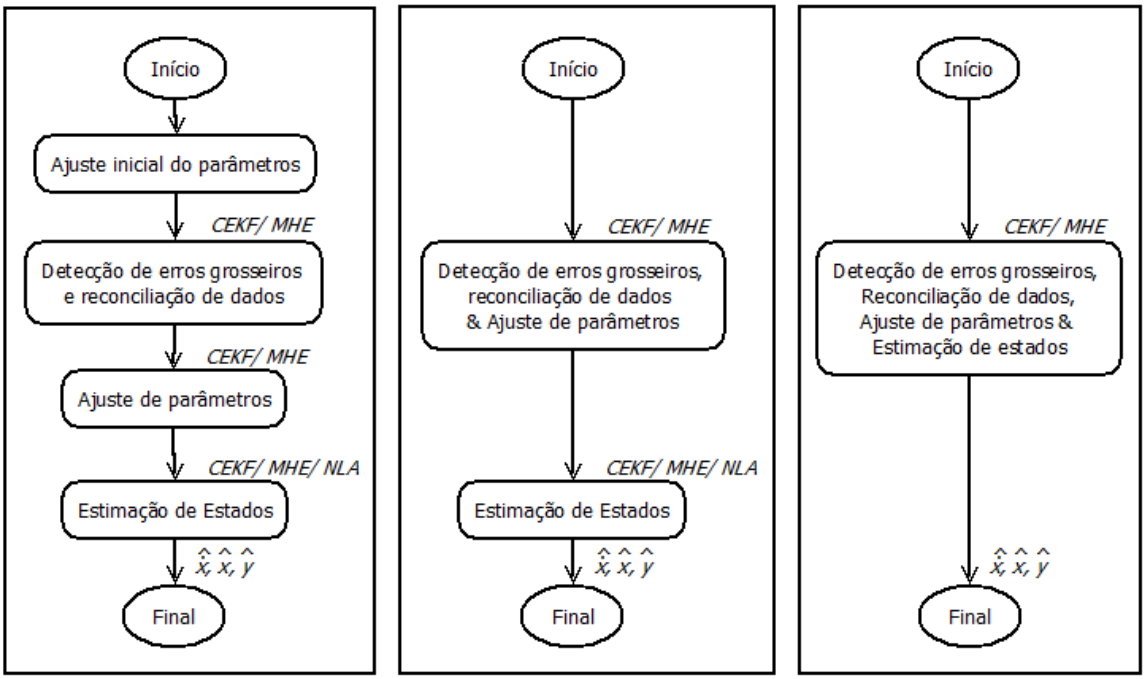


Figura 4.12 – Esquemas de estimação de estados.
(a) seqüencial; (b) reconciliação e ajuste de parâmetros simultâneo; (c) simultâneo.

Estas três atividades não necessariamente precisam ser realizadas no mesmo momento, pois as dinâmicas de perda de calibração de instrumentos (mudança de *BIAS*), de alterações de parâmetros e de mudanças de estados são muito diferentes. Na aplicação *offline* pode ser interessante realizá-los simultaneamente, mas na aplicação *online*, é vantajoso somente em alguns momentos. Não faz muito sentido reajustar os *biases* e os parâmetros do modelo a cada rodada da estimação, porém se houver uma mudança significativa no processo (perturbação ou manobra no processo), passa a ser interessante realizar todas as etapas simultaneamente.

Cabe lembrar que a mesma informação de instrumento da planta (devidamente validada e fornecida pela aplicação de gerenciamento) é utilizada para realizar as tarefas de ajuste de parâmetros do modelo, a detecção de erros grosseiros e reconciliação de dados e estimação de estados. Este fato favorece a realização simultânea de todas estas atividades. As dificuldades na utilização da abordagem simultânea estão da capacidade dos métodos numéricos atuais lidarem eficientemente com as interações tão fortes dos diferentes aspectos do problema de estimação simultânea. A sintonia destes algoritmos requer um conhecimento mais profundo dos erros e variâncias envolvidas no problema.

Com os parâmetros determinados, estimam-se as propriedades estatísticas dos parâmetros (matriz de covariâncias dos parâmetros). Isto define o quão precisos são os parâmetros estimados. Além disso, também são estimadas as propriedades estatísticas do modelo (intervalos de confianças das predições), ou seja, dadas incertezas nos parâmetros, calculam-se as incertezas dos valores calculados. Com os parâmetros determinados e a matriz de covariâncias obtidas, realizam-se os testes de adequação do modelo. Estes testes definem se o modelo é bom o suficiente para ser usado para p propósito determinado. Se este modelo não for adequado, coleta-se uma nova série de dados ou refaz a determinação

de parâmetros com uma nova formulação ou definições do problema de otimização. Se o modelo for adequado e for único, então o modelo pode ser considerado ajustado. Se houver um modelo rival, então podem ser efetuados os testes de discriminação entre modelos. Neste caso, define-se qual modelo é o melhor. Se o modelo ajustado for melhor do que o rival, então se substitui o modelo, caso contrário descarta-se. Este procedimento está esquematizado na Figura 4.13.

Uma atividade adicional na estimação de parâmetros é o diagnóstico da atualização do modelo consiste na análise dos fatores envolvidos no ajuste de parâmetros. Neste diagnóstico, procura-se evitar que as perturbações nos dados da planta se transporte indevidamente para os parâmetros do modelo. Esta análise é feita pela verificação do condicionamento da matriz de covariância dos parâmetros. As medições e os parâmetros selecionados devem resultar em matrizes da covariância dos parâmetros bem condicionadas. No entanto, o cálculo pode tornar-se mal condicionado devido a questões em tempo real como a falta de medições ou variações das condições operacionais. Um método de diagnóstico do atualizador foi desenvolvido por Miletic e Marlin (1998) para detectar mau condicionamento na matriz de covariância dos parâmetros no momento da atualização do modelo, verificando o número de condição da matriz de covariância dos parâmetros. O número de condicionamento da matriz de covariância é obtido pela relação entre o maior e menor valor da diagonal principal da matriz dos valores singulares resultante da decomposição em valores singulares - *SVD*. Se o número de condicionamento for maior que o limite superior de controle pré-estabelecido pelo usuário, o problema de estimação está mal condicionado e a estimação é rejeitada. Neste caso, uma estratégia para a recuperação do mau condicionamento deve ser adotada para evitar a passagem desta variabilidade para o otimizador. Os limites de controle são obtidos através de uma análise da distribuição estatística das estimações sabidamente bem feitas. Ou podem ser ajustados para uma região de operação com base na experiência da planta. Os parâmetros estimados e os ajustes de medição de um problema de estimativa de parâmetro bem condicionado são enviados para o otimizador resolver o *DAOP*.

Dois fatores podem dar mais precisão e confiabilidade nos parâmetros estimados, são eles: a utilização de múltiplos conjuntos de dados e a análise de resultados da estimação. O múltiplo conjunto de dados fornece a oportunidade de se ter aproximadamente repetições de experimentos. Isso contribui para o aumento da precisão na predição, através da redução do intervalo de confiança dos parâmetros. Além disso, uma análise do comportamento histórico dos parâmetros nos auxilia a decidir se os parâmetros foram bem estimados ou não, pois há uma série de parâmetros que sofrem alterações lentamente (sem comportamento ruidoso). Também se considera importante fazer uma análise de significância das alterações nos parâmetros do modelo. Esta análise pode envolver todo o conjunto de parâmetros, bem como um subconjunto selecionado pelo responsável pelo modelo. Ele deve definir quais parâmetros podem ter comportamento ruidoso e devem ser atualizados freqüentemente. Caso a estimação não seja bem sucedida, devem-se utilizar os últimos parâmetros confiáveis estimados. Isto deve ser sistematicamente acompanhado pelo responsável pelo modelo de otimização, pois os parâmetros podem ficar muito velhos e não representar corretamente a planta. Neste caso, devem-se elaborar experimentos específicos ou atualizar parcialmente os parâmetros, elegendo os mais importantes.

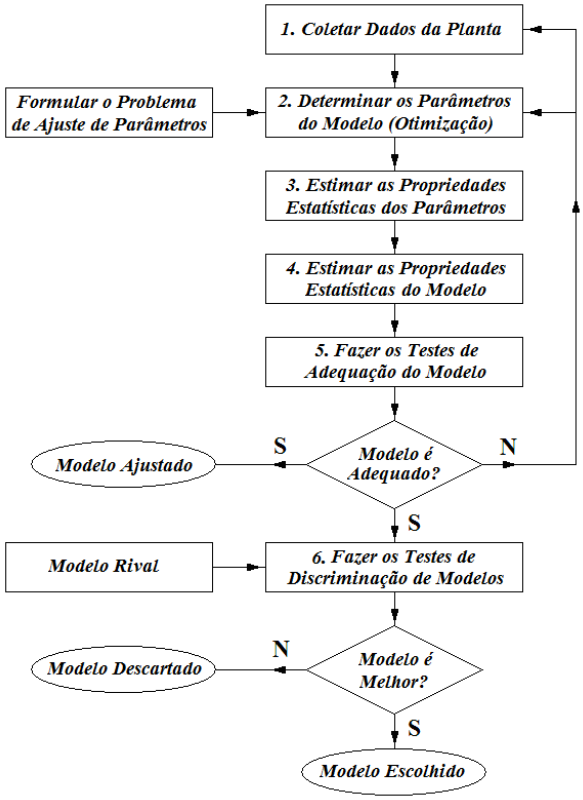


Figura 4.13 – Esquema geral do ajuste de parâmetros e validação do modelo.

De forma geral, pode-se utilizar este módulo com diferentes combinações de funções objetivo no *DAOP*. Na estimação de estados, acrescenta-se a etapa de atualização do estado da planta através da integração simultânea das equações de estado e da matriz de covariância do estado. No modo *offline*, esta atualização não tem muito efeito, mas ao estimar o estado de modo *online*, utilizando janelas de tempo móveis, esta correção passa a ser importante.

Os dados reconciliados devem ser acompanhados para verificar a sua coerência. Este acompanhamento pode ser feito através da análise histórica da tendência do processo, bem como dos *biases* detectados nos instrumentos. Além disso, é importante identificar e acompanhar os instrumentos falhos e considerados com erros grosseiros. Provavelmente, este acompanhamento resultará na requisição a intervenção da equipe de manutenção de instrumentos. Deve ser criado um procedimento de disparo desta manutenção. Caso este procedimento falhe, devem-se utilizar o cenário os *biases* da última reconciliação de dados bem sucedida.

4.1.2.4 Solução do problema otimização dinâmica

Para resolver um problema de otimização dinâmica, tanto *online* como *offline*, são necessários os meios de construção do *DAOP*, as condições em que o problema deve ser resolvido, os algoritmos necessários e a parametrização dos perfis de controle e em alguns casos os estados. A Figura 4.14 mostra os requisitos básicos do módulo de otimização dinâmica. A infra-estrutura necessária consiste de meios de construção de modelos de *DAOP* em ambiente de simulação e otimização dinâmica (como o *EMSO*, *gPROMS*,

Modelica, dentre outros), forma de tradução do modelo para a forma matemática, meios de solução do *DAOP* (através de métodos de discretização e adaptação) e meios de solução do *NLP* resultante (no caso, sugerem-se algoritmos bem sucedidos neste tipo de aplicação - de ponto interior – *IPOPT* e *OPT*++, e de *SQP* – *SNOPT*, *NPSOL*,...).

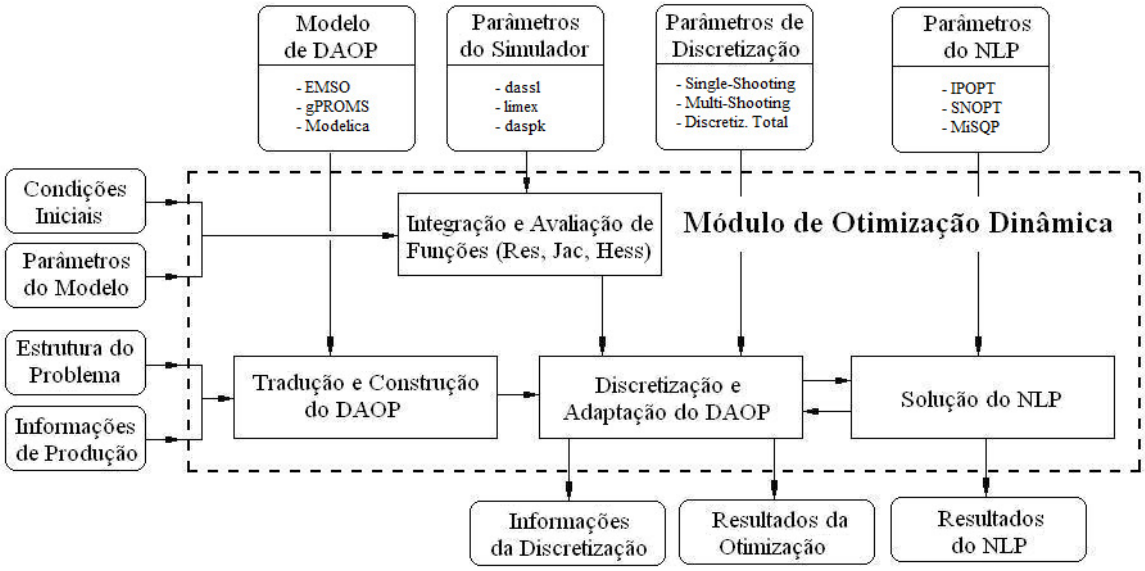


Figura 4.14 – Esquema do módulo de otimização dinâmica.

Dois problemas que afetam qualidade da solução do otimizador dinâmico são a qualidade da discretização (que é função no número e distribuição dos elementos finitos) e da localização do instante de tempo onde as restrições se tornam ativas ou são desativadas (pois o instante real da ativação ou desativação de uma restrição pode estar localizada no interior de um elemento). A Figura 4.15 apresenta um esquema simplificado da localização destas funcionalidades no módulo de solução do *DAOP*.

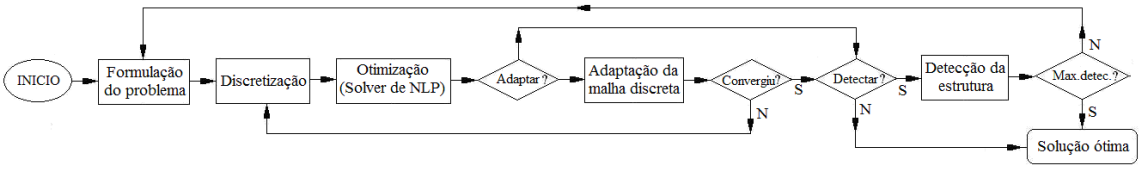


Figura 4.15 – Esquema simplificado da solução do *DAOP*.

Para se obter uma malha discreta de melhor qualidade é recomendável que seja efetuada adaptações na mesma. Para isso, sugere-se o uso de duas abordagens de características distintas, são elas: a adaptação da discretização das trajetórias de controle através de *wavelets* e adaptação com elementos finitos móveis. Há duas estratégias possíveis, a adaptação em separado ou juntamente com a solução do problema de otimização dinâmica. Na abordagem de adaptação simultânea, novas restrições são acrescentadas ao *DAOP* e no processo de solução são discretizadas e resolvidas pelo *NLP* como se fossem restrições de estado com variáveis de controle próprias (h_i – comprimentos dos elementos finitos).

Para se determinar com mais precisão os instantes de ativação e desativação das restrições, é recomendável a utilização de métodos de detecção de estruturas das soluções do otimizador. Estes métodos reformulam o problema *DAOP*, introduzindo estágios de

otimização com tempos finais livres. Desta forma, os instantes de chaveamentos deverão coincidir com os finais de estágios de otimização.

A Figura 4.16 mostra, de forma simplificada cada funcionalidade da solução do *DAOP*. As camadas fornecem uma noção da hierarquia de cada funcionalidade. Note que num loop mais externo temos a detecção de estrutura da solução. Num segundo loop, temos a adaptação da malha discreta e no loop mais interno, obtém-se a solução do *DAOP*. Isto significa que primeiramente é obtida uma solução do *DAOP*, depois modificada a malha discreta e finalmente alterada a estrutura do *DAOP* através da reformulação do problema.

Na estrutura proposta, a solução do *DAOP* é obtida através utilização da biblioteca *DAOP_SOLVER.DLL* (vide apêndice C). Este módulo é o coração do sistema de otimização dinâmica. Ele carrega, atribui valores a variáveis especificadas e parâmetros, interpreta o modelo de otimização transformando-o em um *DAOP* padrão e executa as tarefas de otimização dinâmica conforme o algoritmo escolhido. Também deve realizar a adaptação das malhas discretas e reformulação da estrutura do *DAOP*. Neste módulo devem-se ter as seguintes facilidades:

- Preparação do *DAOP* padrão;
- Verificação da consistência das condições iniciais;
- Discretização do problema de acordo com o método escolhido;
- Adaptação da malha discreta;
- Detecção da estrutura da solução do *DAOP* e reformulação do *DAOP*;
- Solução do *NLP* do problema discretizado.

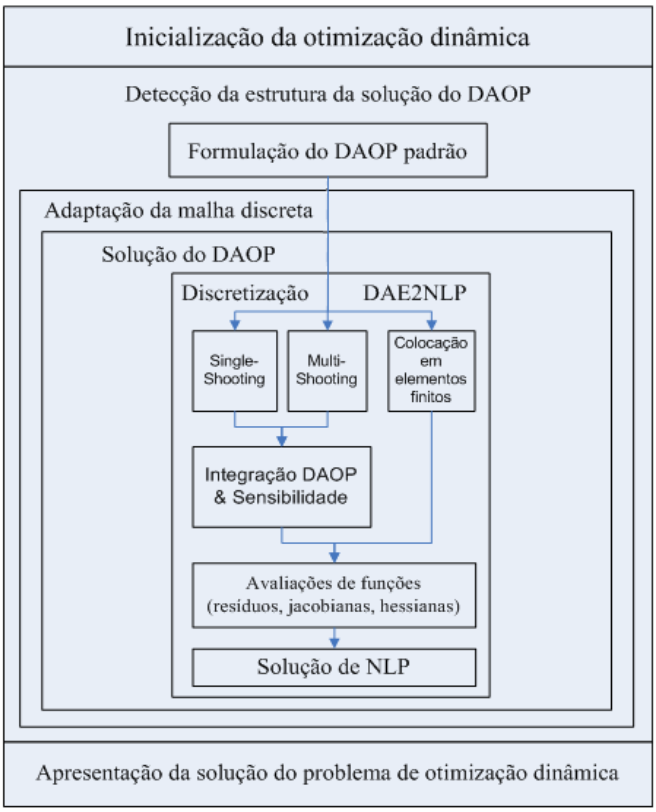


Figura 4.16 – Hierarquia da solução do *DAOP*.

Os algoritmos de *NLP* são utilizados para resolver tanto problemas de otimização estacionária como de otimização dinâmica, pois os problemas de otimização dinâmica se transformam em problemas de otimização estacionária através de técnicas de discretização.

O *DAOPSolver* é um módulo que executa as atividades acima citadas. Porém, durante o processo de solução do problema de otimização dinâmica, o mesmo é convertido em um *NLP* através da aplicação da técnica de discretização. Uma biblioteca específica (*DAOP2NLP.DLL*) efetua esta comunicação entre o algoritmo de *NLP* e o módulo de discretização. A biblioteca *DAOP2NLP* é constituída de um conjunto de funções que enviam e recebem informações do *DAOP* na avaliação de funções solicitada pelo algoritmo de *NLP* (vide apêndice C).

O problema de otimização dinâmica, que passa pela solução do *DAOP*, pode ser formulado de forma geral como:

$$\begin{aligned} &\underset{u,t_f}{Min} \quad f\left(x, y, u, t_f\right) \\ &s.a. \\ &\qquad g\left(\dot{x}, x, y, u, t\right) = 0 \\ &\qquad h\left(x, y, u, t\right) \geq 0 \end{aligned} \tag{4.6}$$

onde *f* representa a função objetivo, *g* o modelo dinâmico do processo e as restrições adicionais de igualdade, *h* representa os limites do sistema e as restrições adicionais de desigualdade.

O módulo *DAOP2NLP* executa o processo de discretização propriamente dito, dispondo de três diferentes abordagens aqui utilizadas para resolver este problema, são eles o método *single-shooting*, *multi-shooting* e colocação em elementos finitos. Para resolver o *NLP*, o solver necessita de avaliações de funções ligadas ao modelo do *DAOP*. O *NLP* necessita a cada iteração das avaliações do vetor dos resíduos das restrições (*EV_G_DAE*), da matriz Jacobiana do processo (*EV_JAC_DAE*), da matriz Hessiana da Lagrangeana da função objetivo (*EV_HESS_DAE*), do valor da função objetivo (*EV_F_DAE*) e do seu gradiente (*EV_GRAD_F*). Estes avaliações dos valores de funções e derivadas primeiras são obtidas da simulação dinâmica do *EMSO*, sendo que as avaliações das derivadas segundas podem ser obtidas por diferenciação simbólica ou numérica (por perturbação), ou por aproximações do tipo *BFGS*.

O *DAOP* genérico é dividido em vários estágios do problema de otimização e cada estágio é dividido em vários elementos finitos discretos. Portanto, o algoritmo de *NLP* recebe os vetores e matrizes de todos os estágios do *DAOP* discretizados e agregados. Esta agregação é feita posicionando cada estágio e estabelecendo as condições de junções dos estágios (condições de continuidade dos estados). Estes estágios, por sua vez, são obtidos das agregações dos elementos finitos discretos de cada estágio. Estes elementos finitos são unidos através das condições de continuidades, que dependendo do método precisam ser explicitadas ou não. Finalmente, a cada elemento, é realizada a integração do *DAOP*, feita por quadratura ou integração numérica. Este esquema simplificado é mostrado na Figura 4.17.

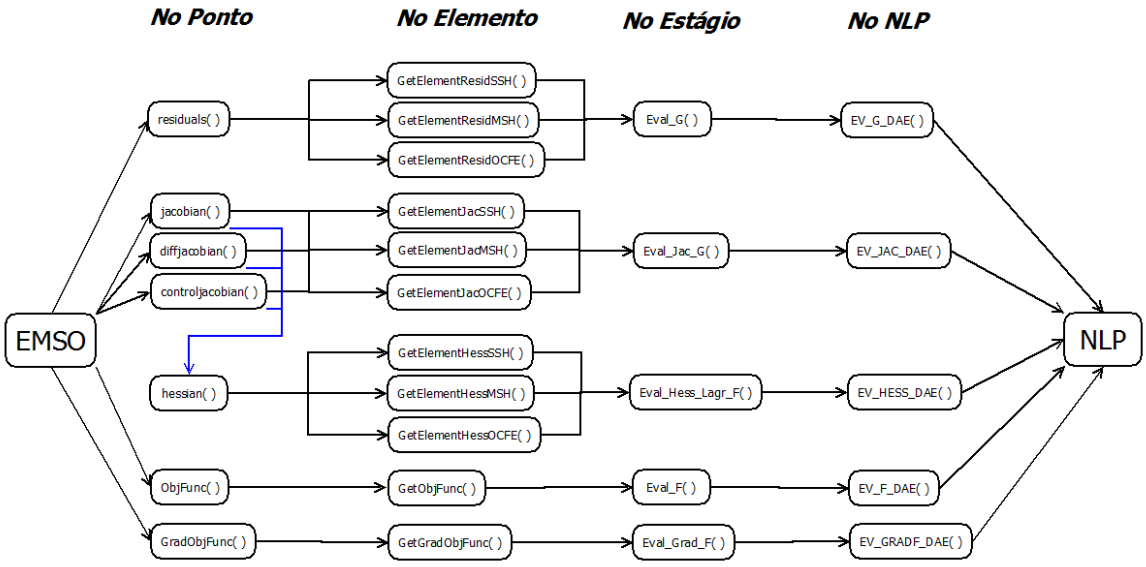


Figura 4.17 – Esquema simplificado do processo de discretização do *DAOP*.

A presente proposta de discretização consiste da construção de blocos modulares de forma a ter a mesma estrutura para os três métodos citados acima. O que muda é o conteúdo de cada bloco. A estrutura de cada bloco pode ser visto no apêndice I.

4.1.2.5 Avaliação e implementação dos resultados do otimizador

Os resultados do otimizador são analisados, validados e implementados neste módulo. Nele são verificadas as condições de saída do otimizador, deve-se verificar se os perfis de controle devem ser enviados para a planta e gerar uma série de informações úteis para o acompanhamento dos resultados do otimizador. Neste módulo devem-se ter as seguintes facilidades:

- Análise dos resultados do otimizador (estados e saídas);
- Verificação das condições de otimalidade da solução;
- Análise discriminante dos perfis de controle;
- Cálculos das sensibilidades com os controles;
- Cálculos dos indicadores básicos para a monitoração;
- Construção do relatório da solução.

4.1.2.6 Gerenciamento e seqüenciamento de tarefas

Este módulo coordena as atividades do sistema de *DRTO*, desde a estimação de estados até a implementação da política ótima de controle. O otimizador deve solucionar o problema de otimização toda a vez que há mudanças na estrutura do problema, ou nas condições do processo, ou seja, o aparecimento de alguma perturbação ou evento na planta ou até mesmo mudança no comportamento do processo (alterações nos parâmetros do modelo). O gerenciador do sistema deve monitorar e disparar esta atividade e passar informações para outros módulos. Além disso, o gerenciador deve passar as informações necessárias para as tarefas de monitoração de resultados e diagnóstico & sintonia do sistema. Como atividade fim, este módulo também implementa as ações de controle ótimo no sistema de controle da planta. Neste módulo devem-se ter as seguintes facilidades:

- Coleta e validação de dados dos instrumentos da planta;
- Monitoração da planta;
- Integração do modelo até o instante atual (no caso de estimação falha);
- Disparo das atividades de atualização do estado e solução do *DAOP*;
- Análise dos resultados do otimizador;
- Implementação dos perfis ótimos de controle no sistema de controle da planta;
- Geração de informações para monitoração de resultados;
- Geração de relatório para diagnóstico e sintonia.

Validação de dados da planta

Neste módulo, o sistema de coleta de dados do processo do *SDCD*, valida e realiza o pré-processamento. As funções básicas são as seguintes:

- Estabelecer a conexão *OPC* para o *SDCD*;
- Obter os dados do processo de *SDCD* em cada amostragem;
- Validar estes dados, detectar erros grosseiros e problemas de comunicação;
- Processo de pré-tratamento dos dados de processo (ex.: filtragem).

Esta funcionalidade consiste da aquisição de dados da planta através de um sistema digital de controle (normalmente *SDCD*) ou alguma base de dados. Este módulo deve coletar e validar os dados da planta referente a algumas variáveis diferenciais e algébricas e todas as variáveis de controle. Com a aquisição de dados da planta e o modelo do processo, realizar-se-á a reconciliação dinâmica de dados, seguida ou em conjunto com o ajuste e atualização de parâmetros do modelo e obtenção das condições iniciais consistentes.

A validação dos dados da planta visa detectar problemas grosseiros ligados à comunicação de dados e falhas de instrumentos. Esta avaliação consiste em verificar falhas de comunicação com o sistema de aquisição de dados, congelamento de dados, falhas de instrumentos (violação dos limites máximos e mínimos válidos) e de picos irreais nos dados de processo. De acordo com esta análise, as informações da planta são definidas como de qualidade boa, questionável ou falho. Os dados falhos da planta não devem ser reconciliados e devem ser considerados como não medidos no conjunto de dados para detecção de erros grosseiros e reconciliação de dados. Os efeitos de ruídos presentes nas medições podem ser reduzidos por filtros passa-baixa, normalmente um filtro de primeira ordem. As medições filtradas são utilizadas nas operações de estimação de estados como um todo. Este esquema é mostrado na Figura 4.18.

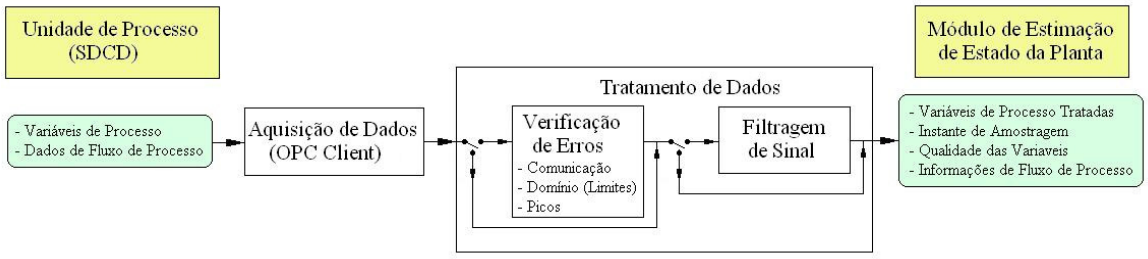


Figura 4.18 – Aquisição e tratamento de dados da planta.

Caso esta operação não tenha sucesso, ou seja, não obtenha solução satisfatória, o mesmo módulo envia as informações para o módulo de diagnóstico e aciona uma tarefa que integra o modelo com as últimas condições iniciais consistentes e parâmetros confiáveis. Esta integração é realizada pelo gerenciador do *DRTO* e leva em conta os perfis das variáveis de controle até o instante inicial da próxima rodada do otimizador dinâmico. Assim, o gerenciador fornece as condições iniciais consistentes para o módulo de otimização dinâmica, mas não necessariamente em concordância com os dados de planta.

Atualização das condições iniciais da otimização

Esta funcionalidade do gerenciador fornece as condições iniciais consistentes atualizadas para o módulo de otimização dinâmica. Esta tarefa deve ser executada pelo gerenciador do *DRTO*, pois tem um ciclo de execução mais curto e possui as informações do processo sempre atualizadas.

O gerenciador deverá fazer uma análise do último estado estimado, que consiste em verificar a idade deste estado estimado como também do cenário atual da planta. O gerenciador verifica se houve alguma falha na aquisição de dados, se a última estimação de estados não foi bem sucedida, se o momento da última estimação está muito distante do tempo atual ou ocorreu algum evento que alterou o cenário do problema de otimização. Isto deve ser feito, pois o estado pode estar desatualizado e necessita ser corrigido. Esta desatualização é devido à data antiga da última estimação ou devido aos degraus aplicados nas variáveis de controle da planta. Neste caso o gerenciador deverá integrar o modelo até o instante de execução do otimizador, utilizando o último estado confiável obtido pelo sistema. Nesta atualização, devem ser utilizados os fatores de correções (*BIAS*) das variáveis de controle obtidos na última reconciliação de dados da planta. Porém, se as condições da planta forem muito diferentes da última estimação, o gerenciador deve disparar uma nova estimação e atualizar o estado nas novas condições. Este procedimento evita perturbações desnecessárias no processo devido às estimações desnecessárias.

Monitoração de eventos da planta

Para executar as diferentes atividades do *DRTO*, é necessário um módulo que coordene as mesmas. O otimizador deve solucionar o problema de otimização toda a vez que há mudanças na estrutura do problema, ou nas condições do processo, ou seja, o aparecimento de alguma perturbação ou evento na planta ou até mesmo mudança no comportamento do processo (alterações nos parâmetros do modelo). Este módulo pode inclusive reiniciar a execução do módulo de otimização diante de uma mudança importante antes do otimizador terminar sua tarefa, antecipando a correção do problema a ser resolvido. O mesmo raciocínio pode ser aplicado ao ajuste de parâmetros ou estimação de estados. Ou seja, se houver algum evento que justifique uma nova estimação de parâmetros e/ou estados (ex.: perturbação importante, manobra na planta, etc), o gerenciador deverá solicitar à aplicação de estimação de estado que atualize os dados da planta.

O monitor de eventos verifica se houve mudança no *DAOP*. Esta análise deve ser acompanhada de uma avaliação das condições da planta, onde se verifica alguma mudança no cenário da planta (como manobras operacionais) ou nas instruções de produção (como mudanças de especificações de produtos ou tanques de destino). Caso isso tenha ocorrido,

o otimizador deve se disparado imediatamente para corrigir a última solução otimizada. Isto é, a cada iteração do otimizador, o monitor verifica se houve algum evento de interrupção da otimização. São considerados eventos de interrupção: o aparecimento de perturbações significativas no processo, mudança na estrutura do *DAOP* (restrições e controles), mudanças de receita (alterações nas posições das restrições em relação à receita prevista), mudança significativa nos parâmetros do modelo ou quando termina uma receita e inicia outra. O término pode ser por atingir o final da receita ou comando do operador. Se ocorrer algum evento de interrupção durante a execução do otimizador, o monitor pára imediatamente o processo de solução do *DAOP* sem parar o otimizador, e reinicia o processo de solução do otimizador *online* com base nos novos dados do *DAOP* e gera relatório de eventos.

Para definir se houve perturbações significativas que justificam uma nova rodada do otimizador, sufere-se a utilização do índice de conformidade o processo com a solução ótima. Estes índices capturam os desvios (previsto x realizado) entre a os valores das variáveis medidas (estado + controle) previstas pelo otimizador - $y^*(t_k)$ e os respectivos valores medidos no processo - $y(t_k)$. Note que estes valores devem ser corrigidos com os respectivos *biases* reconciliados. Este indicador corresponde à medida da distância - d_k (norma 2) entre os valores planejados e realizados no processo no instante atual (t_k), que representa o nível de não conformidade com a receita ótima. Desta forma os indicadores tomam a seguinte forma:

$$I_{Cf} = \frac{1}{ny}\sum_{i=1}^{ny}\frac{d_k\left[y_i^*(t_k), y_i(t_k)\right]+\varepsilon}{\left|y_i^*(t_k)\right|+\varepsilon} \tag{4.7}$$

onde I_{Cf} é o índice de conformidade relativo com a receita ótima, $y^*_i(t_k)$ é o valor ótimo previsto para o instante atual para a variável medida i, $y_i(t_k)$ é o valor atual corrigido para a mesma variável medida, ny é o número variáveis medidas, e ε a precisão da máquina, para evitar a divisão por zero.

Para definir se houve um evento de perturbação relevante no processo, deve-se analisar o grau de conformidade da realização na planta (realizado) com solução ótima (prevista). Se não há alterações significativas, provavelmente a planta está seguindo a trajetória ótima e não há porque disparar o otimizador, já que é uma atividade consumidora de tempo. A definição deste ponto corte depende de um acompanhamento estatístico das conformidades obtidas anteriormente. Com esta referência, pode-se obter um valor limite para o indicador (*benchmark*). Desta forma, a classificação é então baseada neste valor limite e a separação sugerida é a seguinte (vide Tabela 4.1):

Tabela 4.1 – Classificação da conformidade da solução ótima.

Classe	**I_{Cf}**	**Conformidade**
Equivalente	< 10% *Benchmark*	Conforme
Levemente diferente	10% - 100% *Benchmark*	Talvez conforme
Diferente	> 100% *Benchmark*	Não conforme

Disparo do otimizador

O disparador é um mecanismo que monitora o processo e verifica se houve perturbações significativas das condições de otimalidade do *DAOP* (Kadam et al., 2003). Ou seja, verifica se a receita ótima vigente não é mais aplicável ao processo. Se este for o caso, da mesma forma que o monitor de eventos, o sistema *DRTO* aciona o otimizador a fim de obter uma nova receita ótima, caso contrário o sistema *DRTO* simplesmente envia e executa a última receita ótima válida da última rodada calculada pelo otimizador.

Nas atividades em tempo real, há dois pontos importantes a serem analisados antes de executar a tarefa de otimização dinâmica: a análise da necessidade de execução da tarefa de otimização dinâmica e da viabilidade do estado estimado atual. Com estas duas verificações se evita a utilização de condições iniciais inválidas para o otimizador e a execução do otimizador em momentos inoportunos ou desnecessários.

Para decidir o momento de execução do otimizador, deve-se ter um algoritmo de monitoração de receitas do otimizador (onde são verificadas mudanças da estrutura do *DAOP*) e um algoritmo de disparo do otimizador, onde são verificadas as condições de otimalidade da receita implementada.

Além disso, este módulo deve verificar a conformidade da solução proposta pelo otimizador com o realizado na planta. Se houver uma diferença significativa entre o realizado e planejado, o otimizador deverá resolver novamente o problema de otimização e sugerir novas trajetórias de controle. Isto deve ser acompanhado sistematicamente pelo responsável pelo otimizador, pois uma diferença sistemática entre o proposto e realizado pode indicar deficiências na solução do problema de otimização dinâmica (como modelo inadequado, instabilidades nas soluções), perturbações freqüentes na planta (que invalidam a predição), falta de confiabilidade na produção (causado por incapacidade de realizar o proposto) e mudanças freqüentes na ordem de produção (pela interferência freqüência do pessoal de programação de produção). Esta análise também pode ser usada para avaliar a perda de oportunidade de otimização da produção, quantificando inclusive a sua perda financeira.

O disparador do *DRTO* é um módulo que invoca a execução do *DRTO* utilizando uma análise da sensibilidade do sistema às novas entradas do processo. O algoritmo verifica as condições de otimalidade e viabilidade, integrando o estado e as sensibilidades do modelo, linearizando o processo e resolvendo o *QP* das condições de otimalidade com a mesma estrutura do *DAOP*. Com isso, avalia o desvio da solução ótima obtida anteriormente (otimalidade e viabilidade) se este desvio for maior do que certa tolerância, então o gerenciador dispara o otimizador. Ou seja, se o processo estiver fora das condições de otimalidade ou com inviabilidade, então deve re-otimizar o processo. A condição de otimalidade, também deve ser acompanhada pelo Hamiltoniano da solução obtida, que deve ser constante ao longo do horizonte de otimização.

Análise de resultados do otimizador

Uma vez obtida a solução, o otimizador irá analisar os resultados em termos de coerência e significância estatística, isto é, se a solução proposta é significativamente diferente da

última solução. Caso afirmativo verifica-se as condições do processo e enviam-se os novos perfis ótimos das variáveis de controle e de estado para uma aplicação de controle avançado que está em uma camada abaixo na hierarquia de automação. Isto é feito através de análise estatística que discrimina duas series temporais. Usando as informações dos ajustes de parâmetros é possível decidir se a solução obtida pelo otimizador é estatisticamente diferente da última implementação. A análise de resultados é baseada em teste de hipóteses, sendo realizados a cada rodada do *DRTO* para discriminar os resultados na presença de ruídos, erros nas medições e alterações nos parâmetros do modelo. A decisão é função do grau de confiança na análise de discriminação. Se o usuário desejar ter mais confiança na identificação, serão obtidos menos resultados diferentes e menos alterações de receitas serão implementadas.

A análise de resultados deve ser pulada se a rodada do otimizador foi disparada por um evento de mudança de estrutura de receita, ou início de nova receita. Também pode ser incluída neste módulo a análise dos resultados do ajuste de modelos e o envio dos novos parâmetros para o modelo do otimizador. Esta análise verifica o estado de saída do otimizador, verifica se os perfis de controle devem ser enviados para a planta e gera uma série de informações úteis para o acompanhamento dos resultados do otimizador. Uma vez obtida nova receita a ser implementada, coloca-se *PRONTA-PARA-IMPLEMENTACAO*. Caso afirmativo, verificam-se as condições do processo e se envia os perfis ótimos das variáveis de controle e de estado para uma aplicação de controle avançado que está em uma camada abaixo na hierarquia de automação.

Ao resolver o problema de otimização dinâmica usando métodos diretos, a solução obtida é uma aproximação da solução ótima precisa e baseada na satisfação das condições de otimizalidade do *NLP*. Portanto, é necessário verificar as condições de otimalidade dos problemas de controle ótimo. Ao analisar as condições de otimalidade, utilizando as informações do *NLP*, equivale ao Hamiltoniano é constante ao longo do horizonte de otimização. Isto é feito através da avaliação do Hamiltoniano no domínio do tempo discreto. Isto porque é sabido que a solução obtida pelo uso de métodos diretos se aproxima da solução do *DAOP* quando os comprimentos dos elementos finitos tendem a zero.

Implementação dos resultados do otimizador

Uma vez obtida nova receita que deve ser implementada, a mesma pode ser dada por envio de *setpoint* dos controladores do *SDCD*, (que são digitais), bem como o valor alvo das variáveis manipuladas dos *MPC's*. Neste módulo, deve-se ter um sistema de agendamento das ações da receita como as utilizadas em processos de bateladas. Se alguma ação for diferente de ações em degrau, o sistema de implementação deverá converter a receita em pequenas ações em degrau. Além disso, na etapa de implementação também devem ser enviados os limites das restrições que o *MPC* deverá considerar no seu problema.

Ao Implementar os resultados do otimizador, o sistema deverá verificar o estado do processo. Isto é, verificar se a configuração do controle é a mesma em que o otimizador resolveu o problema (variáveis de controle, restrições). Se a configuração do sistema de controle (manipuladas e controladas ligadas) não sofreu alterações, o sistema envia as

ações de controle. Caso contrário, o gerenciador deverá disparar novamente o otimizador dinâmico, para que uma nova receita seja proposta.

Como existem erros sistemáticos nas medições dos instrumentos, se enviarmos os valores verdadeiros para os instrumentos, isso levará a um erro sistemático na implantação da receita ótima, e as variáveis de estado poderão não respeitar as restrições. Portanto, os valores sugeridos pelo otimizador têm desvios em relação aos valores medidos pelos instrumentos, sendo os valores das posições passados para o *SDCD* devem ser compensados pelos *biases* identificados na reconciliação de dados. $u_{BIAS} = \hat{u} - u_{exp}$; $u_m = u_{opt} - u_{BIAS}$.

Geração de informações de monitoração

Na monitoração/auditoria é feito o rastreamento dos estados e seqüências das execuções dos módulos (ex.: instante de tempo, operação, tarefa, status, condição). Para isso, este módulo prepara informações para monitoração e visualização dos resultados do sistema de *DRTO*. A função básica deste módulo é a preparação de informações de monitoração e emissão de relatório de cada etapa do processo de otimização. Para isso, é necessário gerar dados históricos para a monitoração dos eventos do otimizador, monitoração dos perfis dos estados e controles, monitoração dos indicadores de robustez, desempenho e qualidade da solução e monitoração dos indicadores técnicos e gerenciais do problema de otimização.

Neste acompanhamento do histórico das soluções, o sistema fornece dados sobre evolução das soluções. E nesta análise da evolução dos resultados são fornecidas informações que indicam a consistência das soluções, tais como: indicadores técnicos das rodadas (*KPI's – Key Perfomence Indicators*), indicadores do estado geral do otimizador, do problema e da solução e um painel das restrições do problema (planejado, realizado e índice de conformidade).

Nas informações de relatórios, este módulo deverá reportar informações do tipo: Informações gerais (Ex.: Instante, status e tempo de execução); estatística geral – últimas rodadas e últimas rodadas com sucesso (Ex.: fator de serviço, % *online*, % malha aberta e fechada, % sucesso, % falha, % restrições ativas); estados das variáveis; informações da reconciliação de dados e da atualização do modelo; informações da solução de otimização (função objetivo e restrições – ex.: valor, status, ativa, *shadow prices*); evolução das soluções do otimizador (alterações - ex.: valores lidos e ótimos calculados, variações absoluta e relativa); dentre outras (vide apêndice J).

Algumas ferramentas adicionais são interessantes para melhorar a produtividade no uso de otimizadores, são elas:

- Interfaces com Planilhas (Excel) e Banco de Dados – permite transferir dados do otimizador para planilha de cálculo ou banco de dados, para facilitar a análise e reportagem por parte da engenharia.
- Monitor de Eventos – permite analisar a freqüência das mudanças de estrutura do problema de otimização. Isto justifica os benefícios da aplicação de soluções de otimização.

Geração de informações de diagnóstico

Este módulo objetiva criar informações de forma organizada para viabilizar atividade de diagnóstico e sintonia do otimizador, que consiste da leitura de dados de uma determinada situação passada pela aplicação *online* e reproduzir esta situação com o objetivo de identificar problemas ou falhas específicas do otimizador. Na realização desta atividade, pode-se interromper o otimizador em pontos específicos e analisar uma determinada iteração, identificando a fonte do problema que pode ser devido à natureza do problema, do processo de discretização ou do algoritmo de otimização. Para permitir a realização eficiente do diagnóstico, o sistema de *DRTO* deverá fornecer informações para realizar as seguintes atividades:

- Análise dos estados e falhas do otimizador;
- Análise das características do problema de otimização;
- Análise dos resultados do otimizador;
- Análise de sensibilidade dos parâmetros de sintonia do otimizador;
- Análise de convergência, robustez, desempenho e viabilidade da solução.
- Investigação das causas de falhas ocorridas de diferentes níveis de detalhes;
- Testes específicos de verificação de hipóteses sobre o problema ocorrido.

Este módulo deverá tratar as informações e criar arquivos que permitam a realização do diagnóstico. Estes arquivos incluem, o arquivo de modelo de otimização (*DAOP.mso*), arquivo que contém as informações de alterações da estrutura do *DAOP* (*DAOP.dat*), que deverão sobrescrever algumas informações do *DAOP*. Além disso, tem arquivos das condições iniciais (*InitCondintions.dat*), dos parâmetros (*ModelParameters.dat*), de reconciliação (*DataReconciliation.dat*) e do algoritmo de otimização (*NLPSolver.out*). Finalmente, as informações que não estão nos arquivos acima deverão ser depositadas no arquivo de diagnóstico (*DAOP.diag*). Com estas informações depositadas em arquivos, será possível realizar o diagnóstico e até mesmo reproduzir e analisar detalhadamente o caso problemático.

4.1.2.7 Módulo de monitoração dos resultados

A monitoração dos resultados é uma atividade onde o engenheiro pode acompanhar os resultados do otimizador rodada a rodada. Através da monitoração, é possível detectar ocorrências do otimizador e informar os resultados e indicadores de robustez, desempenho e qualidade da solução do otimizador. O acompanhamento pode ser feito de forma gráfica ou em tabelas de dados históricos (vide apêndice J). Estes dados podem ser exportados para outras aplicações. Neste módulo devem-se ter as seguintes facilidades:

- Seleção de dados históricos;
- Monitoração dos eventos do otimizador;
- Monitoração das trajetórias dos estados e controles;
- Monitoração dos indicadores de robustez, desempenho e qualidade da solução;
- Monitoração dos indicadores técnicos do problema de otimização.

4.1.2.8 Módulo de diagnóstico e sintonia

A ferramenta de diagnóstico e sintonia do otimizador tem a função de auxiliar o engenheiro na solução de problemas ocorridos com o otimizador ou melhorar a robustez,

desempenho e qualidade das soluções do mesmo. O diagnóstico e sintonia é uma atividade que pode ser feita simplesmente pela análise de informações fornecidas pelo otimizador, bem como a realização de testes específicos com o sistema ou até mesmo rodar o otimizador de forma *offline* com os dados do caso problemático e obter informações mais detalhadas sobre o problema ocorrido com o otimizador (vide apêndice K). Neste módulo devem-se ter as seguintes facilidades:

- Análise dos estados e falhas do otimizador;
- Análise das características do problema de otimização;
- Análise dos resultados do otimizador;
- Análise de sensibilidade dos parâmetros de sintonia do otimizador;
- Investigação das causas de falhas ocorridas em diferentes níveis de detalhes;
- Testes específicos de verificação de hipóteses sobre o problema ocorrido;
- Diagnóstico e solução de problemas ocorridos com o otimizador.

4.1.3 Formas de uso do *DRTO*

O sucesso das aplicações de *DRTO*, assim como as de automação industrial, depende do bom uso deste tipo de ferramental. O ciclo de utilização de um sistema de *DRTO* tem fundamentalmente quatro etapas, são elas: O projeto e implementação do sistema de *DRTO*, a operação e supervisão dos aplicativos, gestão de resultados, e análise de problemas e oportunidades e diagnóstico de problemas (vide Figura 4.19).

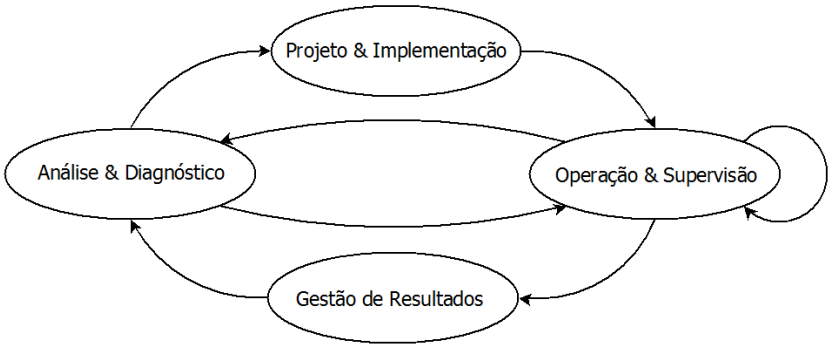


Figura 4.19 – Ciclo de utilização de um sistema de *DRTO*.

O projeto e implementação consistem da definição das funções de otimização dinâmica, da configuração e teste do sistema de *DRTO*. A operação e supervisão se referem à utilização do sistema em tempo real, interagindo com a planta, e a monitoração dos resultados do otimizador. A gestão de resultados consiste do processo de decisão baseado nos resultados econômicos e indicadores técnicos de desempenho da planta. A análise e diagnóstico se referem ao processo de solução de problemas e busca de novas oportunidades de otimização. Para que este ciclo funcione adequadamente, é necessário que diferentes atores executem suas tarefas.

Como foi dito anteriormente, o sistema de *DRTO* consiste de uma aplicação de otimização dinâmica *offline* e um conjunto de aplicações *online*.

A aplicação de otimização *offline*, que é executada de forma desconectada da planta, é utilizada pelos engenheiros de automação e de processos na construção do modelo de

otimização dinâmica, calibração do modelo desenvolvido, estudo de casos de otimização dinâmica, simulação e visualização da solução do otimizador *online* e no diagnóstico do otimizador *online*.

O sistema de otimização *online*, que é executada em tempo real e de forma integrada à planta, é utilizado pelos operadores. Estes operadores interagem com o sistema de *DRTO* de duas formas: em malha aberta ou malha fechada.

Na utilização do sistema em malha aberta (modo de aconselhamento - *Advisor*) o sistema resolve um problema real da planta, mas não implementa os resultados. Isto é, realiza a aquisição de dados da planta, resolver o problema, e somente apresenta os resultados para a engenharia e operação. Este modo de operação pode ser acompanhado de um mecanismo de aceitação total ou parcial dos resultados do otimizador. Quando o operador aceita a solução ótima, o sistema de *DRTO* implementa as ações de controle ótimo automaticamente, uma vez que for dada esta autorização.

Esta forma de uso tem o propósito de auxiliar o engenheiro no diagnóstico do processo, quando houver uma deterioração ou operação anormal na planta. Permite fazer auditorias de benefícios do sistema, através da realização de comparações da solução ótima obtida com um *benchmark*. Serve como ferramenta de suporte à decisão, onde auxilia o engenheiro a definir quando parar algum equipamento por baixo desempenho, ou a sugerir mudanças de receitas ao operador. Ou até mesmo rodar o otimizador em paralelo com o *DRTO* em malha fechada para avaliar o sistema ou buscar oportunidades escondidas.

Na operação em malha fechada, o sistema resolve o *DAOP* conectado à planta, obtendo dados dos instrumentos e implementando os resultados do otimizador automaticamente na planta. O *DRTO* conduz continuamente a planta para a política ótima de controle e responde às alterações no processo (ex.: perturbações no processo e alterações de configuração do *DAOP*).

Diferentes classes de profissionais estão envolvidas na atividade de otimização dinâmica do processo. Os perfis dos profissionais envolvidos são estabelecidos para implementar, operar, manter e gerenciar o sistema. Além disso, há a parte da infra-estrutura de informações do processo que interage diretamente com o sistema e por sua vez com os profissionais envolvidos. Estas partes são os sistemas digitais de controle e as bases de dados de informações. Temos, portanto, os seguintes atores:

Engenheiro de automação – é o profissional que implementa o projeto de otimização dinâmica (define objetivos e restrições, constrói o modelo de otimização, configura e mantém o sistema, monitora as soluções e a implementação dos resultados do otimizador, e resolve os problemas encontrados durante a obtenção da solução pelo otimizador. Este profissional acompanha os indicadores de desempenho do sistema de *DRTO*.

Engenheiro de processos – é o profissional que monitora, avalia e gerencia os resultados do otimizador. Além disso, resolve os problemas relativos à utilização do sistema, isto é, resolve questões relativas à solução dos problemas de otimização dinâmica (como objetivos, restrições e parâmetros adequados do problema). Este profissional também executa os estudos de casos de otimização e auxilia na análise de fidelidade do modelo de

otimização dinâmica. Ele também acompanha os indicadores de desempenho do sistema de *DRTO*.

Gerente da planta – é o profissional que analisa os indicadores de desempenho e resultados do otimizador dinâmico e demanda por intervenções no sistema. Estes resultados podem ser financeiros como de desempenho dos ativos de produção.

Operador da planta – é o profissional que conduz o sistema de *DRTO*, isto é, faz a gestão direta do uso do sistema de otimização *online* (como ligar/desligar, incluir/remover/mover restrições, acompanhar as ações de controle do otimizador). Este profissional demanda por intervenções dos engenheiros de automação e de processos quando detecta algum problema com a utilização do sistema de *DRTO*.

Sistema digital de controle – é a infra-estrutura de controle de processos. Trata-se do *SDCD* (Sistema Digital de Controle Distribuído), que controla a planta e fornece informações de processo ao operador e à interface de comunicação do *SDCD* com as aplicações supervisórias (como o *DRTO*). Normalmente estas interfaces são servidores *OPC* (*OLE for Process Control*).

Base de Dados de Informações – é uma base de dados que deposita as informações de processo e de negócios envolvidos na otimização da operação da planta. É uma base de informações da instrumentação da planta, de qualidades de matérias primas e produtos, inventários, dados econômicos de produção, e indicadores de desempenho e resultados do sistema de *DRTO*.

Implementar o projeto de *DRTO*

Implementar o projeto significa construir o modelo de otimização (*DAOP*). Esta etapa consiste em construir modelo de simulação dinâmica do processo e o problema de otimização dinâmica. Deve-se calibrar este modelo e estabelecer as condições iniciais consistentes do processo para teste do sistema. Estas atividades são executadas utilizando o otimizador dinâmico *offline*. Além disso, do lado da operação do sistema, é necessário configurar e testar o sistema de *DRTO*. Isto significa realizar: a configuração da comunicação com o sistema de controle (configurar os itens *OPC* dos instrumentos do processo), popular a base de dados de informação, construir as telas de operação do sistema e testar o sistema *online* em malha aberta.

Esta atividade é praticamente toda realizada pelo engenheiro de automação, contando com apoio do engenheiro de processos e operadores.

Utilizar o sistema de *DRTO*

Utilizar o sistema de *DRTO* consiste em manter a efetividade do uso do sistema pelos operadores. Para isso, é necessário acompanhar os resultados do otimizador através de sistema de monitoração do *DRTO*. Ao detectar algum mau funcionamento, resolver os problemas que aparecerem com o sistema de *DRTO*. Isto passa pelo uso de algum ferramental de diagnóstico e sintonia do sistema.

Durante o processo de acompanhamento e diagnóstico, poder ser necessário utilizar o otimizador *offline* para visualizar, em detalhes, os resultados de determinadas rodadas do otimizador *online*, importando os dados do caso inteiro ou partes. Pode-se repetir uma rodada do otimizador *online* através do otimizador *offline* e analisar problemas. Pode-se rodar no modo *offline* um caso problemático e analisar iteração-a-iteração do otimizador, além de poder resolver o *DAOP*, flexibilizando as restrições do processo.

Esta atividade é compartilhada entre o engenheiro de processos, que está mais focalizado no acompanhamento dos resultados do otimizador, e o engenheiro de automação, que se dedica mais à solução de problemas ocorridos com o sistema. Ambos os engenheiros e os operadores participam de todo o processo, mas com intensidade e responsabilidade diferenciadas.

Gerenciar os resultados do *DRTO*

Esta atividade tem dois focos principais: a gestão dos resultados econômicos do sistema e a gestão técnica do sistema de *DRTO*. Na gestão de resultados econômicos, o gerente da planta é um ator importante que motiva o uso do sistema e justifica o investimento feito. Para isso, o mesmo precisa acompanhar e analisar indicadores econômicos e técnicos de desempenho do otimizador. Esta atividade tem muita interação com o engenheiro de processos, que por sua vez interage com o engenheiro de automação e operadores para manter o retorno do capital.

Na gestão técnica, o engenheiro de processos realiza análises dos resultados do otimizador para solucionar problemas e buscar novas oportunidades de otimização. Isto passa por analisar e manter a fidelidade do modelo de otimização e realizar estudos de casos com novos cenários de oportunidades. Estas atividades também contam com a colaboração do engenheiro de automação e dos operadores.

Nesta atividade, o engenheiro pode recorrer à execução do otimizador *offline* para buscar oportunidades escondidas. Isto pode passar por alterações na estrutura do *DAOP* a ser resolvido e explorar resultados possíveis em outras condições operacionais. Além disso, pode-se executar a estimação de estados no modo *offline*. Isto permite avaliar mais adequadamente a fidelidade do modelo de otimização. Esta tarefa também pode ser executada para diagnóstico e sintonia do sistema *online*.

Operação do sistema de *DRTO*

A operação do sistema de *DRTO* consiste na condução e solução do *DAOP* no modo *online*. A operação pode ser feita em malha aberta ou malha fechada. Ligar/ desligar, alterar as restrições do problema, monitorar os resultados do otimizador, analisar e criticar os resultados do otimizador. Analisar as soluções do otimizador junto à engenharia de processos. Solicitar a intervenção da engenharia de automação quando há problemas na operação do sistema.

Esquematicamente podemos apresentar estes atores e suas relações no seguinte diagrama (Figura 4.20):

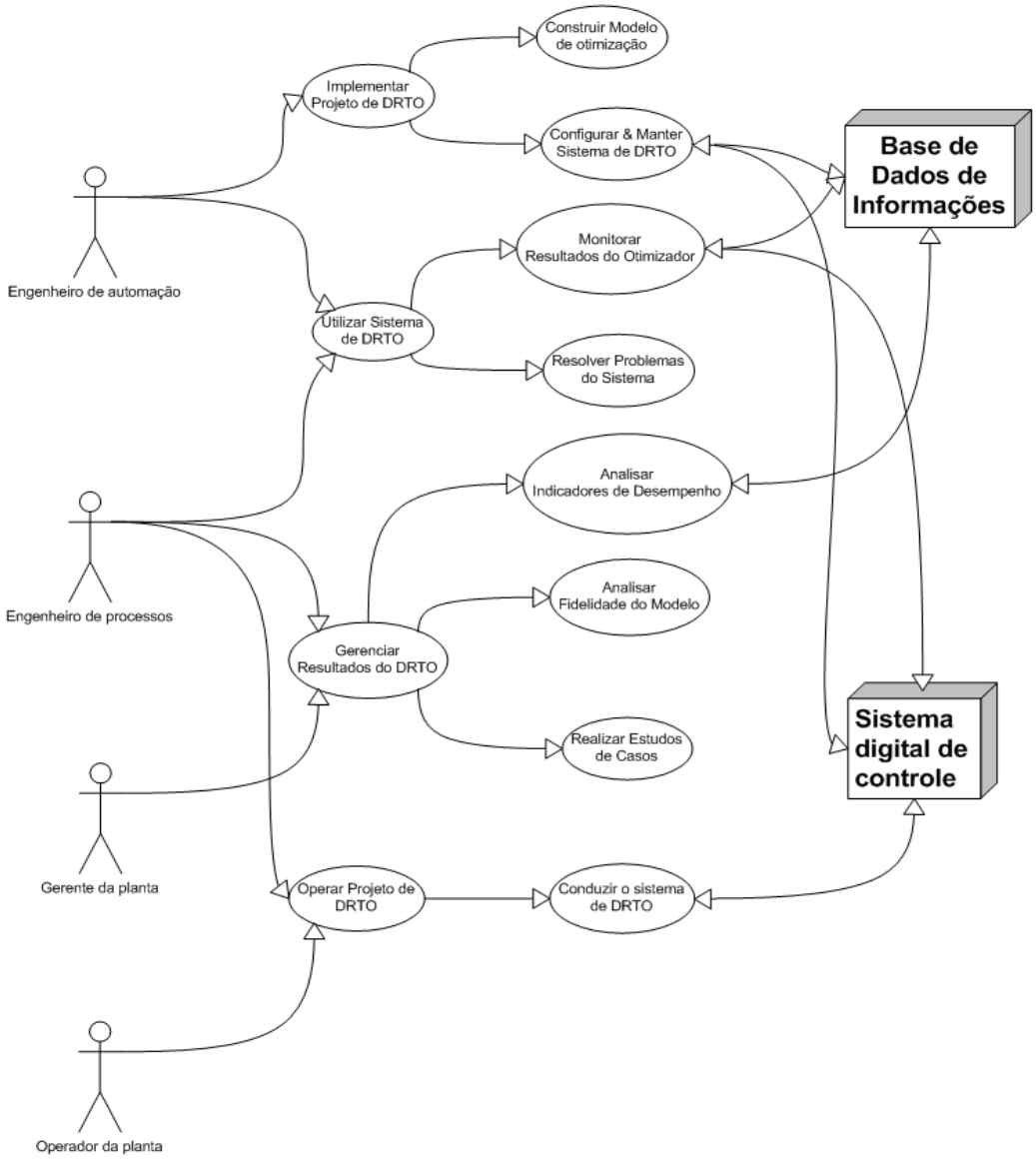


Figura 4.20 – Casos de usos do sistema de *DRTO*.

4.2 Proposta de ferramenta de monitoração e diagnóstico de DRTO

O sucesso na aplicação de uma ferramenta de otimização dinâmica é atingido quando os requisitos dos usuários são atendidos. O atendimento das suas necessidades e expectativas faz com que a solução seja robusta e eficaz, e resultando assim na satisfação do usuário. Com isso, o ciclo de vida da solução é prolongado. A ferramenta de *DRTO* deverá ter habilidade de proporcionar melhorias na produção e contribuir para atender às necessidades do usuário. Porém, quando ocorrem instabilidades ou inadequações nos resultados do otimizador, o desempenho do processo é afetado negativamente, aumentando o custo de produção e gerando insatisfação dos clientes. Além disso, tanto os engenheiros quanto os operadores devem estar atentos à obtenção de novas oportunidades e melhorias no processo produtivo. Para isso, é muito importante que os mesmos tenham mecanismos

para identificar estas deficiências e oportunidades no processo, e assim atingir os seus objetivos de melhoria de produção.

Muitas vezes é difícil apresentar e analisar os problemas e oportunidades de forma objetiva. Os problemas e pontos críticos do processo não aparecem claramente ou são mascarados por transições e instabilidades no processo. Por isso, é importante ter mecanismos eficientes e eficazes de solução de problemas que resultam na diminuição ou eliminação da incidência de erros, incertezas, deficiências e falhas nas soluções propostas pelo otimizador. Isto permite o estabelecimento de uma conexão entre solução de problemas produtivos e os objetivos globais da organização.

4.2.1 Metodologia e requisitos de monitoração e diagnóstico

Em função dessas necessidades e expectativas, propõe-se o projeto de uma ferramenta de diagnóstico e solução de problemas na utilização de um *DRTO*. Esta ferramenta tem por objetivo auxiliar aos usuários na redução e eliminação de falhas e deficiências no uso do sistema de *DRTO*. Para isso, é necessário colocar todo o conhecimento de otimização nesta ferramenta, sem perder a essência e a aplicabilidade do sistema, focalizando nos aspectos práticos e interpretativos da ferramenta. Com isso, os benefícios da solução de problemas de aplicações de *DRTO* são evidenciados através do aumento na confiança na aplicação, aumento da potencialidade de otimização do processo, na satisfação do usuário e na melhoria da qualidade da tomada de decisão. A robustez desta ferramenta possibilita um sucesso sustentável do sistema de *DRTO* que é função da sua capacidade de encontrar e eliminar as causas básicas de erros ou falhas do otimizador, focalizando nos pontos relevantes para os usuários e na sua facilidade de uso.

A ferramenta em si não tem a capacidade de resolver os problemas ocorridos com o otimizador. É fundamental a utilização de metodologias sistemáticas de solução de problemas. Estas metodologias auxiliam na obtenção de melhorias contínuas e na solução de problemas, como por exemplo, o *DMAIC* (*Define* – Definir, *Measure* – Medir, *Analyze* – Analisar, *Improve* – Melhorar, *Control* – Controlar). Esta metodologia utiliza uma abordagem sistêmica e focada em dados para analisar e resolver problemas que comprometem desempenho global das empresas. Ela é usada para promover a melhoria de processos existentes, quando a solução não atende aos requisitos do usuário ou não desempenha adequadamente.

De modo geral, para solucionar problemas ocorridos com o uso de aplicações de *DRTO*, é importante descobrir e mostrar "o que", "o porquê" e "o como" de tais problemas. Esta metodologia procura responder as perguntas a respeito do desempenho do processo:

- Como funciona o processo?
- Quais são os procedimentos utilizados?
- Quais são as fontes conhecidas de instabilidade e qual é seu efeito sobre a qualidade e o desempenho do sistema?
- Qual é o desempenho do processo?
- Quais são os desvios no desempenho?
- A capabilidade atende as necessidades do usuário?
- Os sistemas de medição de processos são capazes de detectar o que incide na qualidade dos resultados observados?

4.2.1.1 Metodologia DMAIC para DRTO

Propõe-se aqui o projeto de um sistema de monitoração, diagnóstico e sintonia de sistemas de *DRTO*. Este sistema procura contemplar todas as etapas do *DMAIC*. Relativo ao item DEFINIR, deve-se especificar o escopo do problema de diagnóstico e sintonia de *DRTO*. Com esse sistema, deve-se ser capaz de acompanhar os resultados do otimizador e corrigir as falhas ocorridas no uso do sistema *DRTO*. Para que seja feito um diagnóstico adequado do problema encontrado no uso do otimizador, é necessário avaliar os resultados do *DRTO* em função do histórico das rodadas. Além disso, deve-se analisar a aplicabilidade, a confiabilidade e robustez dos resultados do otimizador. O alvo principal do sistema proposto é o fornecimento de todas as informações do otimizador para que a operação e engenharia tenham a capacidade de identificar e corrigir falhas ocorridas, auxiliar o usuário na exploração de novas oportunidades de otimização e promover a melhoria de desempenho do otimizador dinâmico. Além disso, pode auxiliar aos desenvolvedores de sistemas a eliminar deficiências ou até mesmo aprimorarem os algoritmos utilizados.

No que se refere ao MEDIR e ANALISAR propõe-se aqui o projeto de uma ferramenta de diagnóstico de *DRTO* e estabelecer uma sistemática de solução de problemas de otimização. Esta infra-estrutura e metodologia buscam encontrar as soluções baseadas nos resultados falhos e deficiências do otimizador, como mostra a Figura 4.21.

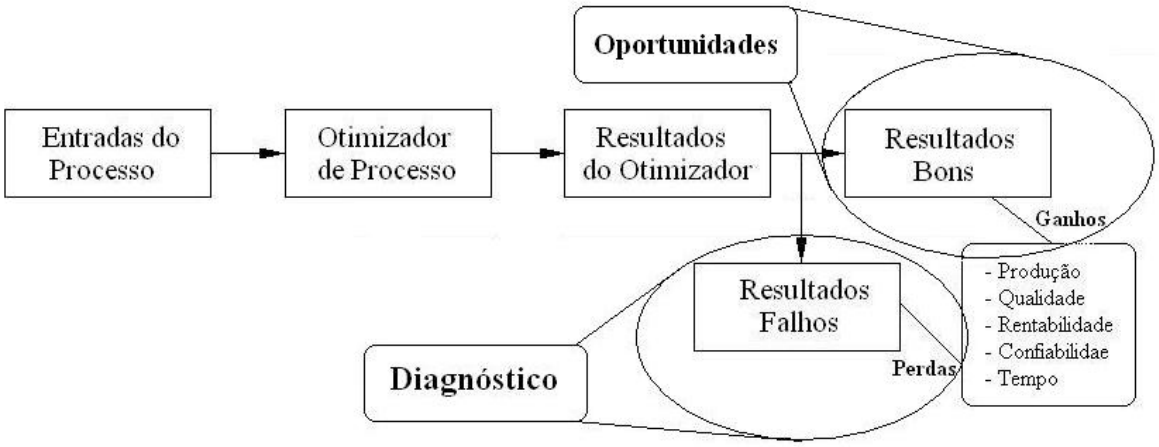


Figura 4.21 – Diagnóstico de falhas e busca de oportunidades do *DRTO*.

Baseado nas informações de entradas do processo, nos resultados e no rastreamento sistemático, através da utilização de métodos matemáticos e heurísticos, as causas de falhas e origens de instabilidades na solução do otimizador podem ser identificadas. Com isso, pode-se fazer um diagnóstico e busca de oportunidades de forma adequada.

O sistema de monitoração e diagnóstico de *DRTO* deve ter indicadores que expressam a qualidade dos resultados do otimizador. Para isso, é necessário que se estabeleça uma medida da qualidade da solução do problema de otimização dinâmica. Esta métrica envolve os diferentes aspectos deste problema. Pode-se destacar:

Distância do ótimo real. Utilizando um *benchmark*, com solução conhecida, mede-se a capacidade de obter a solução ótima. Isto deverá ser feito comparando-se os valores obtidos para a função objetivo e os perfis das variáveis em cada caso.

Obediência às restrições. Verificar se as restrições do problema foram adequadamente respeitadas. Nesta análise, procura-se avaliar os resíduos das restrições.

Consistência da solução. Partindo de diferentes condições iniciais, o algoritmo deve obter perfis ótimos coerentes com os resultados esperados. Alterando os parâmetros de sintonia do otimizador, deve-se verificar se os resultados são consistentes.

Desempenho do otimizador. O desempenho do otimizador é avaliado comparando-se vários aspectos, por exemplo: o tempo de *CPU* gasto para resolver o problema, o número de avaliações da função objetivo e das restrições, o número de iterações no *loop* externo, dentre outros.

Robustez do otimizador. A robustez do otimizador é avaliada medindo a quantidade de problemas resolvidos com sucesso e a dependência da solução com a estimativa inicial e com os parâmetros do modelo e de sintonia do otimizador.

Fazer um diagnóstico adequado de uma falha ocorrida com um problema de otimização muitas vezes não é uma tarefa fácil. Várias questões relevantes precisam ser respondidas, tais como:

- Como analisar o problema de otimização quando o algoritmo falha?
- Como fazer um diagnóstico adequado?
- Como ajustar adequadamente os parâmetros de sintonia do otimizador dinâmico?
- Como corrigir o ajuste (reparametrizar)?

Para se avaliar o problema de otimização dinâmica, deve-se partir de uma dada condição inicial e um perfil de controle proposto pelo otimizador de uma determinada situação. Com isso, integra-se o modelo para verificar se não há instabilidade ou problemas numéricos na integração. Além disso, avalia-se a função objetivo e as violações das restrições. Estas informações levam a concluir que a falha se deve ao problema de otimização ou não.

Para avaliar o erro de quadratura, utilizam-se os valores obtidos por integração e por quadratura. Com isso, calcula-se a integral dos resíduos na quadratura. Se este resíduo for significativo, a precisão da quadratura está baixa e pode prejudicar a obtenção da solução ótima.

Para avaliar o algoritmo de *NLP*, devem-se analisar os efeitos de cada parâmetro do algoritmo e da topologia do problema em uma determinada situação. Isto leva à identificação da origem do problema e indicação de uma possível solução.

Uma vez que se obtenha alguma solução do *DAOP*, deve-se fazer uma análise critica da solução obtida. Neste momento surgem algumas questões como:

- Como analisar os resultados do otimizador?
- Como analisar as oportunidades e cenários?

Para dar suporte a estes questionamentos, é interessante fazer uma análise de sensibilidade do problema de otimização com relação às variáveis de controle e/ou às restrições, além dos efeitos causados pela remoção de restrições de uma determinada situação proposta. Isto pode ser feito pela observação da matriz de sensibilidade do modelo e dos multiplicadores de Lagrange do problema.

Finalmente, relativo aos itens MELHORAR e CONTROLAR deve-se testar a solução proposta, monitorar os resultados do otimizador e avaliar continuamente o desempenho do sistema de *DRTO*. Para isso, é necessário que a ferramenta proposta permita que se verifique se a solução proposta para resolver o problema é efetiva. Desta forma, devem-se fazer testes utilizando a ferramenta de diagnostico e até mesmo rodar novamente o caso problemático de modo *offline*, com as ações corretivas das falhas ocorridas ou com uma nova sintonia do sistema. Para que a solução do problema ocorrido seja sustentável, é necessário que o sistema seja monitorado continuamente. Esta monitoração possibilitará, aos envolvidos no problema, um acompanhamento das soluções do otimizador e verificar se a ação corretiva foi eficaz. Além disso, permite avaliar o desempenho e históricos dos resultados para definir o nível de consistência dos resultados e de satisfação do usuário. A avaliação contínua deve ser acompanhada de um procedimento sistemático e objetivo de analise de processo através de reuniões de trabalho onde se discute os resultados do otimizador e se define o grau de conformidade com os resultados esperados. Estes fatores auferem maior grau de satisfação por parte do usuário e maior robustez por parte do sistema de *DRTO*.

4.2.1.2 Conceitos de monitoração e diagnóstico de DRTO

As ferramentas de monitoração e diagnóstico são aplicativos que tem o objetivo fornecer informações ao usuário sobre os resultados do otimizador. Estes resultados se referem às falhas, robustez e desempenho do sistema de *DRTO*.

A atividade de monitoração se dedica a obter informações rodada a rodada do otimizador com o objetivo de permitir ao usuário acompanhar os resultados do otimizador e estabelecer mecanismos de gestão do uso de tal aplicativo. A atividade de diagnóstico visa fornecer ao engenheiro um ferramental de solução de problemas em otimização dinâmica. Este aplicativo permite identificar causas de falhas, desempenho inadequado e a busca de melhores oportunidades de otimização.

Monitoração é uma ferramenta que tem três partes: uma aplicação que gera as informações de monitoração e roda em tempo real, um módulo de geração de relatórios pelos diferentes módulos do *DRTO* e um visualizador e gerenciador de resultados do otimizador. O objetivo destas três partes é fornecer as informações das rodadas do otimizador de forma organizada a permitir uma ágil e precisa tomada de decisão.

A aplicação que roda em tempo real visa fornecer informações atualizadas do processo. Esta ferramenta é executada de forma cíclica, *online* e coleta informações do processo em tempo real. Nesta aplicação, são monitoradas as condições da planta, as ações do otimizador, a estrutura do problema de otimização e as ações do operador. A parte de geração de relatórios consiste em reportar os resultados do otimizador, informações para geração de indicadores de desempenho, status da rodada do otimizador, dentre outros. A

parte de visualização consiste de um ambiente de navegação onde há um painel de controle com os indicadores de desempenho de caráter gerencial, onde se tem uma visão geral e rápida da qualidade dos resultados do otimizador.

A atividade de diagnóstico do *DRTO* consiste na solução de problemas encontrados no sistema, e é baseado nas informações geradas por cada modulo do sistema. Com a disponibilidade das informações e procedimentos de análise de problemas, é possível encontrar a causas básicas do problema ocorrido ou de mau desempenho do sistema. Além disso, permite identificar em qual parte do sistema de *DRTO* em que a ação corretiva deve ser feita. A ferramenta de diagnóstico pode ser usada de duas formas. A primeira é uma investigação das causas do problema ocorrido baseado nas informações geradas pelo otimizador *online* no momento em que o problema apareceu. A segunda forma é re-executar a rodada problemática, com dados gerados pelo otimizador *online*, com um maior nível de detalhes de informações do módulo de otimização dinâmica.

Na ferramenta de diagnóstico e sintonia do otimizador tem-se uma aplicação onde é efetuada a leitura de dados de uma determinada situação passada pela aplicação *online* e reproduzir esta rodada vivida com o objetivo de identificar problemas ou falhas específicas do otimizador. Na realização desta atividade, pode-se interromper o otimizador em pontos específicos e analisar uma determinada iteração, identificando a fonte do problema que pode ser devido à natureza do problema, do processo de discretização ou do algoritmo de otimização.

O módulo de diagnóstico e sintonia consiste de uma aplicação que é executada de forma *offline* e independente da aplicação *online* de otimização dinâmica. O módulo *online* de otimização arquiva as condições iniciais da última rodada ou da rodada em que houve problema na obtenção da solução ótima. Com isso, pode-se executar a aplicação *offline* do otimizador dinâmico e parar no momento em que houve o problema. A paralisação do algoritmo de otimização permite ao usuário realizar uma análise do problema sobre os três aspectos citados acima. Esta ferramenta possibilita alterar os parâmetros de sintonia do otimizador de forma a obter uma melhor solução do problema. Além disso, esta mesma ferramenta permite identificar possíveis fragilidades dos algoritmos, possibilitando a revisão de seu projeto.

Esta aplicação *offline* tem duas funções básicas: realizar estudos de casos e diagnóstico e sintonia do otimizador. Na execução de um otimizador dinâmico, normalmente se tem problemas devido a três fatores: (1) a estrutura do problema de otimização ou modelo; (2) a qualidade da discretização do problema de otimização dinâmica; (3) as características do próprio algoritmo de otimização.

A atividade de estudo de casos consiste em estabelecer um determinado cenário, executar a otimização dinâmica e analisar os resultados e suas oportunidades na planta. Esta atividade pode ser executada utilizando o próprio otimizador, onde o problema original de otimização dinâmica pode ser alterado ou até mesmo se alterar alguns parâmetros de sintonia do simulado/otimizador. Além disso, pode-se fazer uma análise de sensibilidade do problema à remoção de determinada restrição.

O sistema de diagnóstico e sintonia de *DRTO* deve proporcionar a solução de problemas por alguns pontos de vistas, são eles: ponto de vista da estrutura do sistema de *DRTO*, ponto de vista do algoritmo de otimização e ponto de vista do tipo de falha ocorrida no uso do sistema de *DRTO*. Este sistema deve realizar o diagnóstico através da monitoração dos resultados do otimizador, da realização do diagnóstico seguindo os níveis de detalhamento do problema investigado e na busca de melhores desempenhos em função da monitoração e diagnóstico do *DRTO*.

Geralmente, a dificuldade de se encontrar o ponto ótimo de um problema de otimização pode ter as seguintes causas básicas:

- Problema inviável;
- Existem restrições redundantes na matriz de restrições ativas;
- Problema tem solução finita correspondente a restrições que não estão ativas na iteração considerada;
- Problema com infinitas soluções
- Problema ardiloso (quando pelo menos uma restrição de desigualdade é linearmente dependente de outra restrição) e viável. O algoritmo poderá oscilar entre duas ou mais soluções;
- Problema não ardiloso e crítico (quando o algoritmo não convergiu para uma solução estacionária em um número finito de iterações). A solução não converge e o algoritmo tende a oscilar entre soluções inviáveis.

A análise de algoritmos de otimização consiste basicamente da análise de convergência, robustez e desempenho computacional. O êxito de um algoritmo está ligado à eficiência na geração da direção de busca. Deseja-se que esta direção seja finita e descendente com relação à função objetivo. Esta solução pode não ser única, ou o *NLP* pode não ser convexo. No último caso é mais fácil encontrar um mínimo local do que o mínimo global. E a capacidade de encontrar eficientemente o perfil ótimo global define a qualidade de um algoritmo de otimização.

4.2.2 Processo de monitoração e diagnóstico de DRTO

O processo de monitoração e diagnóstico consiste num conjunto de práticas para avaliar os resultados do otimizador, bem como resolver problemas quando estes aparecem. Normalmente, os operadores utilizam o sistema e obtém vantagens econômicas do seu uso. Quando estes resultados não são satisfatórios, estes operadores demandam por intervenções da engenharia para fazer com que o sistema obtenha estes resultados esperados. Além disso, o sistema poderá apresentar falhas ou mau funcionamento causado por erros operacionais, problemas de processo ou problemas com o otimizador.

Os gestores, por outro lado, acompanham os resultados com enfoque mais gerencial e através de um sistema de monitoração do *DRTO*. Uma vez que estes resultados não sejam satisfatórios, os mesmos demandam por intervenções por parte da engenharia.

A engenharia, por sua vez, rotineiramente deve acompanhar os resultados do otimizador, de forma a identificar o mau funcionamento ou falhas, problemas de desempenho e coletar informações para relatar os benefícios obtidos pelo otimizador. Para executar tais tarefas, o engenheiro deve utilizar um sistema de monitoração do *DRTO*.

Uma vez que o engenheiro detectou algum problema a ser resolvido, através do sistema de monitoração, o mesmo fará um diagnóstico diretamente (quando o problema for de fácil identificação), ou deverá utilizar alguma ferramenta de diagnóstico para identificar e resolver o problema encontrado. Portanto, o processo de solução de problemas é realizado na seqüência "Monitora e realiza o diagnóstico".

Para realizar a contento estas atividades, é necessária a utilização de ferramentas de monitoração e diagnóstico. A seguir, serão apresentadas as estruturas destas duas ferramentas.

4.2.2.1 Na ferramenta de monitoração do DRTO

A ferramenta de monitoração tem uma estrutura de acompanhamento de informações históricas do sistema de *DRTO* com o objetivo de fornecer ao usuário uma visão da evolução das soluções obtidas pelo otimizador. Basicamente esta ferramenta procura dar uma visão geral das rodadas do otimizador utilizando indicadores de desempenho, e apresentando estas informações na forma gráfica e de relatórios.

Este sistema tem dois enfoques principais, são eles: o aspecto gerencial e o aspecto técnico.

Acompanhamento gerencial da otimização dinâmica

O acompanhamento gerencial consiste da utilização de indicadores de gestão que podem ser apresentados na forma de relatório e de gráficos de tendência. Normalmente tem uma forma de resumo gerencial, e um pequeno nível de detalhe. Os indicadores gerenciais são: a robustez, o desempenho, a qualidade da solução, o fator de serviço, a conformidade com a receita.

Robustez - a robustez se refere ao percentual das otimizações executadas foram bem sucedidas. Um índice elevado (de 99 e 100%) pode ser considerado muito bom, e um índice baixo (inferior a 80%) é considerado muito ruim. Uma redução neste índice pode ser causada por problema de modelagem do *DAOP* (incluindo o modelo do processo), dados ruins do processo, algum problema numérico ou de algoritmo.
Desempenho - o desempenho se refere ao percentual de tempo de *CPU* gasto, tendo como referência um *benchmark* baseado nos resultados históricos (Dolan e Moré, 2001).

Qualidade da solução - refere-se à capacidade de obter a solução ótima, tendo como referência um *benchmark* (com solução conhecida), Comparam-se os valores da função objetivo e os perfis das variáveis em cada caso. Nem sempre a qualidade pode ser medida, neste caso, terão menos valores de qualidade do que o número de rodadas bem sucedidas do otimizador. A quantidade depende da freqüência com que o *DAOP* é alterado (o problema e as condições do processo).

Fator de serviço - este fator fornece a informação da disponibilidade do sistema para os operadores. É a percentagem do tempo em que o sistema está rodando e disponível para otimizar. Este indicador está relacionado à confiabilidade do sistema. Um fator de serviço

baixo remete a uma análise da confiabilidade do sistema e provavelmente a realização de um diagnóstico desta não disponibilidade.

Conformidade com a receita - este indicador mostra a diferença entre a receita realizada e a planejada. A conformidade é o percentual das diferenças entre os perfis realizados e os perfis de controle e estados fornecidos pelo otimizador, tendo como referência os valores propostos pelo otimizador. A não conformidade da receita indica a não aceitação das receitas propostas ou dificuldade de realizá-las, e isto deverá ser diagnosticado.

Fator de Utilização - este indicador fornece a informação da utilização do sistema por parte dos operadores. É a percentagem do tempo em que o sistema está ligado e atuando na planta. Este indicador está relacionado à robustez do sistema e da aceitação por parte dos usuários. A não utilização do sistema remete a uma análise do problema e provavelmente a realização de um diagnóstico desta não utilização.

Neste ambiente, o gestor poderá acompanhar os dados históricos dos indicadores, escolhendo o período de análise e algum momento em foco. Para mais detalhes, vide apêndice J.

Acompanhamento técnico da otimização dinâmica

Este acompanhamento se refere à monitoração e análise das condições em que o otimizador deverá resolver o problema (*DAOP*) e das atualizações do modelo do processo. Para isso, deverá se monitorar a atualização de parâmetros do modelo, a coleta de dados e a reconciliação das medidas dos instrumentos da planta e estimação de estados do processo. Além disso, também deverão ser acompanhados os eventos relacionados ao problema de otimização (vide apêndice J).

No acompanhamento destes tópicos, tem-se uma uma visão das rodadas dos módulos do otimizador e dos resultados destas rodadas. Portanto, podem-se organizar os indicadores na seguinte forma:

- Acompanhamento da atualização do modelo do processo;
- Acompanhamento da reconciliação de dados do processo;
- Acompanhamento da estimação de estados do processo;
- Acompanhamento da aquisição e validação de dados;
- Acompanhamento dos eventos da planta;
- Acompanhamento da solução do problema de otimização dinâmica;
- Acompanhamento implementação dos resultados do otimizador;

O acompanhamento da atividade de atualização de parâmetros é efetuado através da visualização dos indicadores específicos e de gráficos de tendência. Na visão dos resultados, deve-se acompanhar: a significância das alterações nos parâmetros (número e variações significativas); valores das incertezas absolutas e relativas das predições das restrições do *DAOP* (Intervalo de confiança das predições); a variabilidade (média e desvio padrão) e as tendências dos parâmetros (crescente, decrescente ou estável).

Se a qualidade do ajuste for ruim, o coeficiente de correlação for baixo e/ou alguns parâmetros estiverem nos seus limites, provavelmente há problemas com os dados

experimentais. Neste caso, os valores históricos dos parâmetros podem explicar a tendência, ou remete-se ao diagnóstico. Também se deve realizar o diagnóstico, caso a estimação não tenha sido bem sucedida e este fato não é explicado pelos dados.

Na visão da reconciliação, deve-se acompanhar: a significância das alterações nos bias dos instrumentos (número e variações significativas); a variabilidade dos biases (média e desvio padrão); as tendências dos bias (crescente, decrescente ou estável); e as informações dos instrumentos considerados com erros grosseiros.

Se o desvio do balanço material for muito grande, provavelmente há problemas de calibração de instrumentos de processo. Se não for este o problema e a reconciliação estiver sendo efetuada simultaneamente com o ajuste de parâmetros, pode haver problemas com os dados experimentais ou ajuste do otimizador (no caso reconciliação e ajuste de parâmetros). Neste caso, os valores históricos dos *biases* podem explicar a tendência ou oscilação. No caso de tendência, pode ser perda de calibração ou então, remete-se ao diagnóstico. Também se deve realizar o diagnóstico, caso a reconciliação não tenha sido bem sucedida e este fato não seja explicado pelos dados.

Na visão da estimação de estados, deve-se acompanhar deve-se acompanhar: a significância das alterações nos estados estimados (número e variações significativas) - diferença estimação x integração do último estado estimado (no mesmo instante de tempo); e análise das inviabilidades nas condições iniciais (violações das restrições nas condições iniciais - estados e controles).

Se houver desvios significativos em relação aos valores obtidos na solução do sistema NLA do otimizador, provavelmente há problemas com os dados experimentais ou de modelagem. Neste caso, os valores históricos destes desvios podem explicar a tendência, ou remete-se ao diagnóstico. Também se deve realizar o diagnóstico, caso a estimação não tenha sido bem sucedida e este fato não é explicado pelos dados.

Ao monitorar a aquisição e validação de dados, deve-se acompanhar: valores históricos do status da comunicação OPC (indica a disponibilidade do dispositivo de aquisição de dados) e % de dados falhos ou inválidos na última coleta (indica a intensidade de falhas de comunicação). Esta informação remete ao diagnóstico.

Na visão dos eventos da planta, deve-se acompanhar: os valores históricos da % das execuções do otimizador que foram concluídas. Com esta informação, tem-se o nível de intervenção e de perturbação na planta. Toda vez que o módulo de otimização estiver resolvendo o *DAOP* e algum evento de interrupção ocorre na planta ou uma perturbação sensível acontece, a obtenção da solução é interrompida e iniciada um novo processo de solução com o novo cenário.

Na visão da rodada, deve-se acompanhar: as mudanças de receitas do otimizador, por inclusão e exclusão de restrições de controle e estado ou alterações nos valores das restrições de controle e estado (indica o nível de intervenções por parte do operador da planta); as perturbações inesperadas e significativas na planta (indica o nível de estabilidade da dos controles da planta); e as mudanças na ordem de produção - final da

receita ou nova receita (indica o nível de intervenções da área de planejamento e programação da produção).

Ao encontrar problemas na atualização dos estados da planta (atualização de parâmetros, reconciliação de dados e/ou estimação de estados), há uma grande chance da causa ser modelo, parametrização ou dados experimentais inadequados. Neste caso, a solução do problema ocorrido deve ser auxiliada pelo uso da ferramenta de diagnóstico. Também pode haver problemas na aquisição de dados. Neste caso, o diagnóstico normalmente é direto e pode explicar o problema na atualização do estado da planta. Os eventos ocorridos na planta também podem explicar os problemas de fidelidade do modelo. Mudanças na receita, perturbações no processo e mudanças nas ordens de produção podem explicar a freqüência dos disparos do otimizador e a não otimalidade, não rastreamento ou problemas na obtenção da solução do otimizador. Neste caso, também deve ser utilizada a ferramenta de diagnóstico para buscar as causas dos problemas encontrados.

O acompanhamento das soluções dos problemas de otimização se refere à monitoração e análise das soluções do problema (*DAOP*) e das circunstâncias em que o mesmo é resolvido. Para isso, deverá se monitorar a o problema de otimização dinâmica e os resultados do otimizador dinâmico.

Na monitoração do problema de otimização dinâmica, deve-se acompanhar: valores históricos dos erros nas especificações das restrições de estado - limites máximo, mínimo e de domínio (indica quais restrições estão especificadas de forma errada); os erros nas especificações dos controles (indica quais variáveis de controle estão especificadas de forma errada) e tempo final (indica erro na sua especificação); e erros de discretização - qualidade da discretização (número e distribuição dos elementos) - erro de quadratura (indica a precisão e adequação da discretização e adaptação da malha discreta).

Na visão da rodada, deve-se acompanhar: as datas, horários das soluções do otimizador (indica a sua freqüência de disparo); datas, horários das soluções bem sucedidas (indica a obtenção de soluções ótimas); e os tempos gastos nas otimizações (indica o seu desempenho);

Na visão dos resultados, deve-se acompanhar: os estados de solução ótima ou aceitável e falhas do otimizador por problemas de convergência ou divergência, solução inviável ou ilimitada, problemas numéricos, de construção do modelo e de codificação do programa (indica sucesso ou natureza da falha); evolução das soluções do otimizador - ex.: perfis ótimos calculados, variações absoluta e relativa (indica a evolução dos perfis obtidos nas soluções)

Os dados desta monitoração são obtidos do sistema de informações históricas do processo e dos arquivos de relatórios do otimizador. A cada rodada do otimizador, o mesmo deverá criar uma versão do arquivo de resultados com a data e hora do disparo. Com isso, têm-se as receitas de rodadas passadas.

Ao encontrar erros grosseiros nas especificações das restrições de estado e controle explicam diretamente o insucesso do otimizador. Neste caso, pode-se optar por corrigir tais erros diretamente ou flexibilizar as restrições quando for justificável. Os problemas na

qualidade da discretização explicam o não rastreamento das soluções do otimizador, devido à sua imprecisão e por ser impraticável. Este fato remete ao diagnóstico da adaptação da malha, da escolha e parametrização dos métodos de solução de *DAOP* e *NLP*. As violações das restrições e restrições ativas justificam o não cumprimento da receita por soluções não aceitas pela operação, incertezas nos perfis de controle obtidos pelo otimizador, modelo inadequado ou devido ao problema mal resolvido. Estes problemas remetem à necessidade de se fazer um diagnóstico mais preciso com o uso de ferramenta adequada. Além disso, o otimizador pode ter desempenho fraco, sendo necessário melhorar seu desempenho com o auxilio de ferramentas de diagnóstico. Há outros problemas na obtenção da solução que indicam fortemente o uso de ferramentas de diagnóstico. Sendo fundamental quando houver problemas de convergência ou divergência, inviabilidade, problemas numéricos, ou instabilidades nas soluções sem mudanças no problema de otimização.

Referente à monitoração e análise das implementações das soluções do problema (*DAOP*). Para isso, deverá se monitorar os aspectos das transformações dos resultados do otimizador dinâmico em receitas de controle de processo, como mostra a Figura J.8 (Apêndice J).

Na visão das implementações, deve-se acompanhar: os valores históricos das significâncias das alterações nas receitas ótimas (solução proposta é significativamente diferente da solução anterior); da otimalidade das soluções (Hamiltoniano constante); e análise do *NCO* - arcos singulares e não singulares

As percentagens de implementações, conformidades nas soluções e significâncias das alterações indicam a normalidade nas soluções do otimizador. Por outro lado, problemas de otimalidade (Hamiltoniano) e nas estruturas dos arcos (*NCO*) indicam a fuga das condições de otimalidade ou incapacidade de implementar os perfis ótimos. Para resolver estes problemas, devem-se utilizar ferramentas de diagnóstico.

4.2.2.2 Na ferramenta de diagnóstico e sintonia do DRTO

Durante a utilização do sistema de *DRTO*, podemos encontrar uma série de problemas de diferentes naturezas e complexidades de soluções. Para resolver estes problemas, propõe-se a criação de uma estrutura de aplicativo de diagnóstico e sintonia para o otimizador dinâmico. Este aplicativo é executado de forma *offline*, onde o sistema coleta informações de saídas e relatórios do sistema de *DRTO*.

A ferramenta de diagnóstico e sintonia é um ambiente que visa analisar uma rodada específica em que houve falha ou obteve uma solução sub-otimizada. Além disso, pode-se fazer uma análise de uma solução otimizada sob demanda. Isso pode ser feito com o objetivo de verificar se a solução obtida provavelmente está no ponto ótimo ou não. E também fazer um estudo de oportunidades com um planejamento determinado, isto é, fazer uma análise da sensibilidade da solução à movimentação de restrições ou de mudança de receita ou planejamento de produção.

Esta ferramenta deve ter a capacidade de avaliar detalhadamente e identificar os fatores de insucesso de um algoritmo de otimização. Esta análise pode ser não invasiva, feita através do uso de indicadores associados a determinados pontos do algoritmo, ou invasiva, feita

através da paralisação do algoritmo em determinados pontos e analise de alguns parâmetros de depuração do algoritmo.

Na análise não invasiva, os problemas encontrados durante a solução do otimizador podem ser analisados em três níveis de detalhes (vide apêndice K). Na análise invasiva, o sistema de diagnóstico pode rodar novamente o caso problemático de forma *offline*, utilizando o ambiente *EMSO* com os dados de entrada do caso problemático. A análise pode ser efetuada, rodando-se o otimizador *offline*, parando a aplicação temporariamente e verificando os resultados e informações do algoritmo a cada iteração do otimizador. Esta análise pode ser feita também com alterações de alguns parâmetros do problema um até mesmo com outra sintonia.

Voltando à análise não invasiva, esta ferramenta tem diferentes níveis de diagnósticos e diferentes aspectos. Podemos citar os seguintes aspectos:

- Análise da formulação do problema de otimização dinâmica;
- Análise da solução do problema de otimização dinâmica;
- Análise de sensibilidade da solução em relação aos parâmetros do otimizador;
- Análise das mensagens do otimizador

Na análise da formulação do *DAOP*, o objetivo principal é prover ao usuário informações relativas às dimensões do problema contínuo, da consistência das condições iniciais, da viabilidade inicial do problema de otimização e uma idéia da complexidade e não-linearidade do modelo de otimização. Estas informações são obtidas, seguindo o caminho da ferramenta de diagnóstico a partir da visão geral do problema de otimização.

Na análise da solução do *DAOP*, o objetivo principal é verificar as causas de problemas de inviabilidade, não convergência, falta de robustez e desempenho computacional baixo. O êxito de um algoritmo está ligado à capacidade solução da viabilidade do problema, na habilidade da definição da direção de busca e progresso para a solução ótima, e na sua eficiência computacional. As dificuldades de conseguir cumprir com os quesitos acima citados devem levar a uma investigação criteriosa. A investigação das causas de problemas ou mau desempenho começa com a obtenção de informações do algoritmo na visão de alarmes e eventos do otimizador, e principalmente pela navegação da visão da solução do problema de otimização (nível 1 de diagnóstico), chegando até a entrar no mérito do funcionamento do algoritmo de otimização se for necessário (nível 3 de diagnóstico).

Na análise da sensibilidade da solução do *DAOP*, o objetivo principal é verificar se é possível resolver o problema encontrado na investigação da solução através de mudanças na configuração do problema como um todo. Isto pode ser feito tanto com mudanças na formulação do problema, no modelo do processo, e/ou nos parâmetros de sintonia dos algoritmos utilizados. Neste último, pode-se planejar uma análise de sensibilidade dos parâmetros dos algoritmos através da visão de sensibilidade dos parâmetros da ferramenta de diagnóstico.

Na análise das mensagens do otimizador, o usuário pode visualizar de forma organizada e em diferentes graus de detalhes as informações geradas durante a obtenção da solução do *DAOP*. Estas informações servirão como ponto de partida para o processo de investigação

das causas do problema encontrado. Esta visualização é efetuada através da visão de alarmes e eventos na ferramenta de diagnóstico.

A seguir, será apresentada a estrutura desta ferramenta de diagnóstico nos seus diferentes níveis de detalhes. Esta ferramenta pode ser utilizada com qualquer tipo de algoritmo de otimização. Ao entrar cada vez em mais detalhes do processo de solução, inevitavelmente deverão ser investigados os comportamentos dos algoritmos. Ao realizar as investigações neste nível de detalhes, a estrutura da ferramenta de diagnóstico deverá permitir que se obtenham informações específicas de cada algoritmo utilizado. A idéia é fazer a primeira parte da navegação com telas comuns a todos os algoritmos e outras específicas dos algoritmos utilizados. Portanto, deverão ter telas para os algoritmos de solução de *DAOP*, como também as telas para algoritmos de *NLP* (no caso, para *SQP* e pontos interiores - *IP*, e para os algoritmos *IPOPT*, *OPT++*, *SNOPT* e *NPSOL*). A idéia básica é utilizar as informações dos arquivos de saída dos algoritmos de códigos fechados e criar mecanismos de reportagem nos algoritmos de códigos abertos. Também se procura utilizar o máximo de telas configuráveis, onde a sua estrutura é definida por arquivo específico de cada solver. Em último caso, constroem-se telas específicas dos algoritmos. Neste trabalho, será apresentada a ferramenta de diagnóstico focada no método de *DAOP* de colocação em elementos finitos com o algoritmo de *NLP* sendo o *IPOPT*.

Na descrição da ferramenta em questão, serão apresentados os seguintes tópicos:

- Nível 1 – Visão geral
  - Visão da formulação do *DAOP* (problema de otimização dinâmica)
  - Visão da solução do *DAOP*
  - Visão dos parâmetros dos algoritmos (de *DAOP* e *NLP*)
  - Visão dos alarmes e eventos do otimizador
- Nível 2 – Visão Detalhada
  - Referente à formulação do *DAOP*
    - Visão das variáveis e seus limites
    - Visão das restrições do *DAOP*
    - Visão da função objetivo do *DAOP*
    - Visão da discretização do *DAOP*
  - Referente à solução do *DAOP*
    - Visão das iterações do solver de *NLP*
    - Visão do desempenho do otimizador - estatísticas de tempos
  - Referente aos parâmetros dos algoritmos
    - Análise de sensibilidade dos parâmetros
- Nível 3 – Diagnóstico
  - Referente à solução do *DAOP* (*IPOPT*)
    - Visão das fases do *NLP*
    - Visão da verificação da convergência
    - Visão da atualização da Hessiana
    - Visão da atualização do parâmetro de barreira
    - Visão da definição da direção de busca
    - Visão do cálculo do comprimento do passo
    - Visão do julgamento do ponto tentativa

Nível 1 – Visão geral

O objetivo da análise neste nível é fornecer ao usuário uma visão geral do problema resolvido e das informações geradas na solução deste problema. Com estas informações, tem-se uma idéia da natureza do problema e de sua dimensão. Além disso, obtêm-se informações gerais sobre a solução do problema, do desempenho do otimizador e das mensagens de alarme e eventos gerados pelo otimizador. Com estas informações, podem ser identificadas as causas básicas do problema ocorrido se a mesma for de simples verificação. Também pode ser feita uma avaliação dos resultados do otimizador (perfis de controle), mesmo que a solução tenha sido bem sucedida. Na aba de parâmetros, os mesmo podem ser visualizados e testados.

Na visão geral do problema de otimização, temos as dimensões do problema de otimização contínuo (*DAOP*), onde são apresentados os números de variáveis de estado diferenciais e algébricas, variáveis de controle, parâmetros e restrições e limites adicionais do problema de otimização. Também são apresentados os números de restrições de igualdade (número de equações do modelo) e desigualdade do problema (vide apêndice K). Na seção de discretização, têm-se os parâmetros usados para transformar o problema contínuo em discreto. Neste caso, são apresentados o método escolhido e as definições básicas de discretização. Os métodos possíveis são *single-shooting*, *multi-shooting* e discretização total por colocação ortogonal em elementos finitos.

Além disso, tem-se a quantidade de variáveis envolvidas na função objetivo. A qualificação do tamanho do problema, aqui proposta, é definida com base no número de variáveis discretas no problema de *NLP*. O tamanho pode ser classificado em (Vide Tabela 4.2):

Tabela 4.2 – Classes de tamanhos de problemas de *NLP*.

Número de Vars (m)	***Classe de Tamanho***
1 – 10	*Minúsculo*
11 – 100	*Pequeno*
101 – 10000	*Médio*
10001 – 100000	*Grande*
100001 – 1000000	*Muito Grande*
> 1000000	*Ultra Grande*

Se o usuário desejar obter informações mais detalhadas sobre a dimensão do problema, ele pode clicar no botão **detalhes** localizado na parte inferior da seção de qualificação de problema. Quando o tamanho do problema se torna muito grande ou ultra grande, é importante obter mais detalhes sobre o mesmo.

O principal objetivo desta visão geral da solução do *DAOP* é concentrar-se nas informações de resultados da solução geral (vide apêndice K). Esta visão fornece uma idéia do status da solução, dos principais problemas e do desempenho do sistema. Nesta visão, visualizam-se as informações do estado de saída do otimizador. O algoritmo de otimização retorna um estado que pode ser classificado como na Tabela 4.3, onde o algoritmo (no caso *IPOPT*) reporta ao final de sua execução. Cada algoritmo tem seu conjunto específico de mensagens, que devem ser interpretados de acordo com o algoritmo utilizado.

Tabela 4.3 – Estados de saída possíveis da solução do *NLP*.

Estado da Solução	***Mensagem***
SUCCESSFUL	Solução ótima encontrada.
CONVERGED_TO_ACCEPTABLE_POINT	Convergia a um ponto aceitável.
STOP_AT_TINY_STEP	Parou com um tamanho do passo muito pequeno.
USER_REQUESTED_STOP	Parando otimização no ponto atual por solicitação do usuário.
LOCAL_INFEASIBILITY	Convergiu para um ponto inviável local. Problema pode ser inviável.
MAXITER_EXCEEDED	Número máximo de iterações ultrapassado.
ERROR_IN_STEP_COMPUTATION	Erro na etapa computacional (regularização se torna muito grande?)
INVALID_NUMBER_DETECTED	Detectado um número inválido na função ou de derivada do *NLP*.
RESTORATION_FAILURE	Falha de restauração!
DIVERGING_ITERATES	Iterações divergentes; problema pode ser ilimitado.
INTERNAL_ERROR	Erro interno: Estado de retorno do solver desconhecido valor.

Na seção de resultados da visão da solução, têm-se as informações gerais sobre os critérios de convergência do otimizador. Mostra os valores iniciais (iteração 0), valores finais e qualificação de cada critério da solução final. Estes são: a função objetivo, inviabilidade dual (definida como: $Inf\_du = max(|\nabla_x L|_\infty, |\nabla_s L|_\infty)$, onde L é o Lagrangiano da função objetivo), violação da restrição (definida como: $c_{viol} = \max\left(|c(x)|_\infty, \left|d^L_{viol}\right|_\infty, \left|d^U_{viol}\right|_\infty\right)$, onde $d^L_{viol} = \max\left(0, d^L - \left(P^L_d\right)^T d(x)\right)$ e $d^U_{viol} = \min\left(0, d^U - \left(P^U_d\right)^T d(x)\right)$), da condição de complementaridade (definida como: $Compl = \max\left(\left|S_{x_L} z_L\right|_\infty, \left|S_{x_U} z_U\right|_\infty, \left|S_{S_L} v_L\right|_\infty, \left|S_{S_U} v_U\right|_\infty\right)$) e do erro global do *NLP* (que é o máximo valor das normas infinitas da inviabilidade dual, violação de restrição e complementaridade, definida como: $E = \max\left\{\left\|Inf\_du / s_d\right\|_\infty, \|Inf\_pr\|_\infty, \left\|Compl / s_c\right\|_\infty\right\}$, sendo s_d e s_c seus fatores de normalização). As suas qualificações, aqui propostas, são baseadas em suas respectivas tolerâncias (*dual_inf_tol, constr_viol_tol* e *compl_inf_tol*) conforme a Tabela 4.4.

Tabela 4.4 – Classes de inviabilidade primal, dual e complementariedade.

Inf_du, Inf_pr e Compl	***Classe***
$0 - 10^{-3} \times tol$	*Muito Bom*
$10^{-3} \times tol - tol$	*Bom*
$tol - 10^{3} \times tol$	*Satisfatório*
$10^{3} \times tol - 10^{6} \times tol$	*Ruim*
$> 10^{6} \times tol$	*Muito Ruim*

Se o usuário desejar obter informações mais detalhadas sobre a solução do problema, ele pode clicar em detalhes localizados na parte inferior da seção de resultados.

Na seção de desempenho do sistema, Nesta seção têm-se as informações gerais sobre o desempenho do otimizador. É apresentado o número de iterações para convergir ou terminar a otimização. A qualificação do número de iterações baseia-se para o número ideal de iterações e está relacionada com o tamanho do problema. Esta referência poderia ser estabelecida, armazenando o menor número de iterações (n), com bons resultados, da

configuração atual. A qualificação, aqui proposta, pode ser definida como $2^{Categoria} \times n$, como mostra a Tabela 4.5. Isso pode ser feito clicando no botão de novo *benchmark*.

Tabela 4.5 – Classes de número de iterações do otimizador.

Categoria	***Iterações***	***Classe***
0	*1 – n*	*Muito Bom*
1	*n+1 – 2n*	*Bom*
2	*2n+1 – 4n*	*Satisfatório*
3	*4n+1 – 8n*	*Ruim*
4	*> 8n*	*Muito Ruim*

Tempo de *CPU* é o tempo de máquina para convergir ou terminar a otimização. No aplicativo de *DRTO* o usuário precisa obter uma série de soluções durante um determinado período de tempo (para processos químicos, geralmente pelo menos 6 horas / dia). Precisa-se obter alguma solução de forma ágil quando algo é alterado no processo. O usuário precisa estabelecer um valor de referência para este desempenho e deve considerar o tempo que a operação pode aguardar por uma nova receita de *DRTO* (geralmente 30 min para processos químicos atuais). Por conseguinte, este será o tempo máximo aceitável de *CPU*. Com base nisso, sugere-se aqui a seguinte escala de qualificação (este é um ponto de partida).

A qualificação do tempo da *CPU*, aqui proposta, baseia-se no melhor desempenho de uma determinada configuração de problema. Esta referência poderia ser estabelecida, guardando o menor tempo de *CPU*, com bons resultados, da configuração atual. A qualificação pode ser definida como tempo de porcentagem do desempenho *benchmark*, como mostra a Tabela 4.6. Este *benchmark* é obtido por análise do engenheiro ou utilizando a metodologia proposta por Dolan e Moré (2001).

Tabela 4.6 – Classes de desempenho do otimizador - Tempo de *CPU*.

Tempo de CPU (min)	***Classe***
0 – 10% CPU_{bench}	*Muito Bom*
10 – 20% CPU_{bench}	*Bom*
20 – 50% CPU_{bench}	*Satisfatório*
50% – 100% CPU_{bench}	*Ruim*
> 100% CPU_{bench}	*Muito Ruim*

Se o usuário desejar obter informações mais detalhadas sobre o desempenho do sistema *DRTO*, ele pode clicar em detalhes localizados na parte inferior da seção de resultados.

Na visão geral dos parâmetros do otimizador, o principal objetivo é se concentrar na análise dos seus parâmetros e na realização da análise de sensibilidade dos mesmos (vide apêndice K).

Nível 2 – Visão Detalhada

Este nível de diagnóstico procura fornecer ao usuário alguns detalhes sobre uma rodada específica do sistema. Estas informações permitem que o usuário identifique a parte do algoritmo que teve problemas. Neste nível, procura-se mostrar os detalhes do problema em

questão, das informações da solução deste problema e dos parâmetros de sintonia dos algoritmos utilizados para solucionar o problema. Na apresentação dos detalhes do problema, são visualizadas informações das dimensões das variáveis e seus limites. Na apresentação dos detalhes do problema, são visualizadas informações dos resumos das iterações, de desempenho do otimizador, e da sensibilidade dos parâmetros de sintonia dos algoritmos.

Na seção de definições da função objetivo, concentra-se na formulação do problema (*DAOP*). Têm-se as informações sobre a função objetivo. São apresentadas as variáveis que estão na função objetivo, seus tipos e os estágios ou elementos em que a função objetivo é avaliada. Aqui são apresentadas as quantidades e os tipos de restrições do problema contínuo e discreto. Nesta seção, são visualizadas informações do número total de restrições, de igualdades e de desigualdades, de desigualdades com apenas limites inferiores ou superiores, e com ambos os limites.

Na seção de definições da discretização do problema, são visualizadas informações dos detalhes do método de solução de *DAOP*, têm-se as definições do número de elementos finitos iniciais e dos parâmetros de adaptação e agrupamento de elementos. Para o método de discretização total, acrescenta-se a visualização do número de pontos de colocação. Além disso, são visualizadas informações da parametrização dos controles.

Na obtenção de detalhes da solução do problema de otimização dinâmica, tem-se a seção das iterações do algoritmo de *NLP*. Nesta seção, são visualizadas informações das iterações resumidas reportadas durante a solução do problema de *NLP* (vide apêndice K).

Na seção de desempenho nas avaliações de funções, são apresentados os tempos de *CPU* gastos para o algoritmo avaliar as funções do problema (vide apêndice K).

Na seção de desempenho do algoritmo de otimização, são apresentados os tempos de *CPU* gastos para as tarefas de solver. Aqui são visualizadas as informações sobre tempo de *CPU* totais gasto nessas tarefas, porcentagem de tempo gasto para cada tarefa e a qualificação do desempenho dessas tarefas. Estas tarefas são inicialização de iteração, atualização da Hessiana, enviar saída de iteração, atualizar parâmetro barreira, cálculo da direção de busca, computação do ponto aceitável, aceitação de ponto tentativa e verificação de convergência.

Na seção de desempenho total do otimizador, são apresentados o número total de avaliações de funções e o número de iterações do solver. Têm-se as informações sobre tempo de *CPU* total gasto em todas as avaliações de funções e como também para todas as tarefas. São calculadas as porcentagens de tempo gasto nas avaliações de funções e no solver. Tem-se ainda a qualificação dos seus desempenhos, conforme a Figura 4.54.

Esta tela fornece informações para uma análise crítica do desempenho do sistema *DRTO*. O usuário pode identificar qual tipo de avaliação de função ou tarefa precisa ter seu desempenho melhorado.

A avaliação de desempenho do sistema pode ser realizada na forma de gráficos de perfil de desempenho, utilizando a metodologia de Dolan e Moré, 2001 (vide apêndice K).

Nível 3 – Diagnóstico

Este nível de diagnóstico tem o objetivo de fornecer ao usuário mais detalhes sobre uma iteração específica do otimizador, para permitir que o usuário execute diagnóstico nessa iteração. Estas informações permitem que o se identifique a parte do algoritmo que apresentou problemas. Nesta análise, são visualizados os seguintes aspectos do algoritmo de *NLP* (no caso do *IPOPT*): fases do programa, verificação da convergência, atualização da Hessiana, atualização do parâmetro de barreira, direção de busca, comprimento do passo e definição do ponto tentativa.

A visão da atualização da Hessiana fornece alguns detalhes da atualização da matriz Hessiana da Lagrangeana da função objetivo. Ao escolher uma determinada iteração, o usuário verifica as características da matriz Hessiana. Das informações da matriz, destacam-se: as dimensões da matriz, o seu posto, seu número de condicionamento, sua positividade, singularidade e inércia. Além disso, verifica-se o autovalor máximo e mínimo desta matriz. Problemas com estes fatores podem fazer com que o otimizador tenha dificuldades de definir a direção de busca do otimizador. As correções destes fatores podem ser desde uma modificação no escalonamento das variáveis ou até mesmo reescrever o modelo do processo de outra forma.

Com esta análise se conclui a investigação das causas do problema encontrado ao solucionar o *DAOP*.

O processo natural de investigação consiste primeiramente em verificar as mensagens do otimizador. Se esta informação não for suficiente, o usuário poderá verificar a estrutura do problema, para verificar o grau de dificuldade do *DAOP* em questão (dimensão e complexidade). Após esta verificação, inicia a investigação da falha pela análise da solução num primeiro nível de diagnóstico. Neste nível, a atenção principal está na análise dos critérios de convergência do algoritmo de otimização, com foco na verificação da viabilidade do problema (violação das restrições). Se este nível de informações não for suficiente para encontrar as causas básicas da falha, recorre-se ao segundo nível de diagnóstico. Neste nível, obtêm-se informações mais detalhadas sobre os mesmos aspectos citados acima. Neste nível e no anterior, também pode ser realizada uma análise de desempenho do otimizador, quando o mesmo estiver com desempenho fraco. Nesta análise são identificadas as tarefas que estão comprometendo mais fortemente o desempenho do sistema.

Se o segundo nível de diagnóstico não for suficiente para identificar as causas da falha ocorrida, recorre-se ao terceiro nível de diagnóstico. Neste nível, são investigadas todas as iterações do algoritmo de *NLP*, procurando identificar quais tarefas do otimizador apresentaram dificuldades ou até mesmo falharam. Neste ponto, entra-se no mérito das funcionalidades de cada tarefa do algoritmo de *NLP*. Os pontos principais a serem analisados são: os critérios de convergência do algoritmo, a viabilidade do problema, o progresso da função objetivo, a definição da direção de busca do ponto ótimo e o passo do otimizador. Se esta análise não for suficiente, então se investiga em detalhes cada tarefa do algoritmo.

4.2.3 Formas de Uso da Ferramenta de Diagnóstico

O diagnóstico do sistema de *DRTO* pode ser feito através da análise das informações fornecidas pelo próprio otimizador. Estas informações devem ser inspecionadas à luz das características do *DAOP* e da qualidade da solução do otimizador. Com esta análise e da realização de testes específicos é possível fazer um diagnóstico da falha ou mau funcionamento do sistema.

A solução de problemas ocorridos durante a solução de um *DAOP* é realizada fundamentalmente executando três atividades, utilizando uma metodologia e um ferramental adequado. As atividades básicas do *DMAIC* aqui consideradas são: análise, diagnóstico e ação corretiva da falha ou mau funcionamento. A ferramenta que realiza a análise e diagnóstico tem dois enfoques de navegação, são eles o enfoque na estrutura do sistema de otimização dinâmica (monitoração), e na natureza e informações da solução do problema de otimização dinâmica.

Há duas formas de conduzir a solução de problemas (conforme mostra da Figura 4.22). Pode-se realizar a análise de falhas, onde deve ser efetuado o diagnóstico da falha ocorrida baseado no status do otimizador e utilizando as informações geradas pelo mesmo durante o processo de otimização. A segunda forma é a realização da análise de processo do otimizador, utilizando o otimizador *offline* na plataforma *EMSO*. Nesta análise, são avaliadas a robustez, o desempenho e oportunidades de otimização. Na análise de robustez e desempenho, são verificados os efeitos dos fatores que os afetam. Para isso, são realizados testes de sensibilidades aos fatores em foco no *EMSO*. Já a análise de oportunidades consiste na realização de estudos de casos utilizando o otimizador *offline* no *EMSO*, onde o usuário propõe mudanças no problema de otimização *online* (ex.: analisar o efeito da eliminação ou flexibilização de uma determinada restrição do *DAOP* nos resultados econômicos do processo).

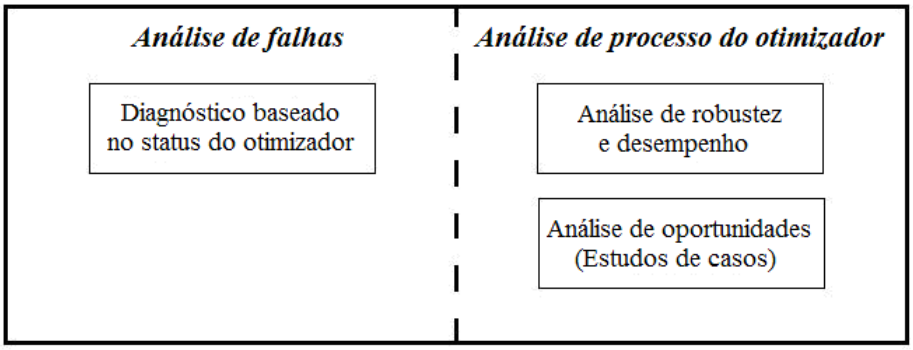


Figura 4.22 – Classificação básica das análises de processo e de falhas na otimização.

4.2.3.1 Análise de falhas

Na solução de falhas ou mau funcionamento do sistema há diferentes formas de diagnosticar o sistema *DRTO*, pode-se concentrar na falha do otimizador ou problema encontrado (com base no status de saída do otimizador) ou concentrar-se aos processos de *DRTO* (com base no desempenho do otimizador e em dados históricos dos resultados do otimizador). Normalmente, o processo de diagnóstico começa com a monitoração do sistema para indicar a natureza do problema ocorrido durante a solução do otimizador e

posteriormente se recorre ao uso da ferramenta de diagnóstico para identificar as causas deste problema.

Ao resolver um *DAOP*, podem-se encontrar problemas com diferentes complexidades a serem diagnosticados e solucionados, são eles: problemas simples, medianos e complexos. Nos problemas simples (ex.: falhas nos dados da planta), os diagnósticos são fáceis de fazer e as soluções normalmente são obvias e rápidas de serem implementadas. As causas de problemas dessa natureza podem ser encontradas facilmente através do uso da ferramenta de monitoração ou de diagnóstico no nível 1 de análise de problemas (como apresentado anteriormente). A identificação de falhas em problemas de dificuldades medianas (ex. inviabilidade em problemas pequenos) é fácil de realizar e usualmente efetuada através da ferramenta de monitoração. Já as causas do problema são normalmente encontradas através da investigação das informações fornecidas pelo otimizador. Neste caso, utiliza-se a ferramenta de diagnóstico, navegando nos níveis 1 e 2 de diagnóstico, como apresentado anteriormente. Nos problemas complexos, há maiores dificuldades em encontrar as causas da falha e normalmente exigem que seja realizada uma análise sistemática e até mesmo alguns testes, para encontrar as suas causas básicas. Neste caso, provavelmente exigirá a investigação das causas através do diagnóstico no nível 3 da mesma ferramenta.

A análise do problema ocorrido, utilizando a ferramenta de diagnóstico já apresentada, normalmente se inicia pela visualização dos alarmes e eventos gerados pelo otimizador. Uma das informações mais importantes é o estado de saída do otimizador, onde se verifica se houve falha na obtenção da solução e a sua natureza. Genericamente, o estado de saída do otimizador pode assumir os seguintes valores: ótimo encontrado, ótimo não encontrado, erro de processamento e parado pelo usuário. Quando o ótimo for encontrado, a solução pode ser ótima ou o algoritmo pode ter convergido a um ponto aceitável. Quando o ótimo não é encontrado, o problema pode ser inviável, não convergiu (por ter atingido o número máximo de iterações ou não consegue progredir - melhorar a função objetivo) ou então divergiu (violações dos critérios de convergência do *NLP* se tornaram muito grandes). O erro de processamento pode ser devido a um erro de codificação, problema numérico ou até mesmo causado por um dado de entrada indevido. Quando é parado pelo usuário, o algoritmo pode ter sido interrompido pelo comando do usuário ou pelo gerenciador do *DRTO*, devido a mudanças no problema de otimização (a rodada é descartada).

Na seção de alarmes e eventos, além das informações geradas pelo algoritmo de otimização, também são apresentadas informações dos outros módulos do sistema de *DRTO*. Os problemas básicos dos algoritmos de atualização dos estados da planta, modelo e comunicação com o sistema de controle da planta podem ser classificados em: condições iniciais inviáveis ou inconsistentes; valores dos estados e parâmetros, qualidades e falhas da atualização parâmetros do modelo, da reconciliação de dados e estimação de estados; falhas na aquisição de dados; e falhas na análise e implementação de resultados.

O diagnóstico da atualização dos estados da planta e do modelo do processo começa com a monitoração da atualização dos estados e modelo (atualização dos parâmetros do modelo, da reconciliação de dados, da estimação de estados, da aquisição e validação de dados e dos eventos da planta). Se este acompanhamento remeter ao uso da ferramenta de diagnóstico, a análise de falhas e ocorrências pode ser efetuada da mesma forma adotada

para o caso otimização dinâmica. Lembre que os problemas de estimação de parâmetros, de estados e reconciliação são *DAOP's* com formulações especificas. Portanto, podem sofrer os mesmos males de um problema de otimização dinâmica. É claro que as circunstâncias em que um problema de estimação é resolvido são diferentes da otimização da operação da planta.

Além disso, no processo de validação do ajuste do modelo, deve-se efetuar uma avaliação da dinâmica do processo (ganhos, constantes de tempo, tempos mortos, tempos de estabilizações e entalpias). Também se pode complementar a validação do ajuste do modelo através da realização de uma análise de sensibilidade das variáveis estado com os controles e parâmetros. Estas duas atividades fornecerão informações preciosas da fidelidade do modelo à planta. Há situações na reconciliação de dados onde se pode observar alguma variabilidade nos bias dos instrumentos, isto pode indicar problema na realização desta tarefa, pois os instrumentos não perdem a calibração rapidamente e nem oscilam na sua calibração. Isto pode ser resultado de problemas numéricos com o algoritmo de otimização ou nos dados experimentais. Divergência entre a estimação e a solução do *NLA* também devem ser tratadas no processo de diagnóstico do otimizador. Análise das inviabilidades nas condições iniciais - violações das restrições nas condições iniciais - estados e controles, também devem se resolvido no diagnóstico.

O diagnóstico da solução do problema de otimização tem um processo mais complexo. Ao resolver o *DAOP*, o otimizador pode acusar problemas ou não, e as condições da solução devem ser avaliadas tanto no caso bem sucedido como de falha no algoritmo de otimização *NLP*. No caso de solução bem sucedida, a otimalidade da solução deve ser verificada, e em caso de falha, devem-se analisar suas causas. Normalmente, elas estão relacionadas a erros nas especificações das restrições (limites - máximo, mínimo e domínio), erros nas especificações dos controles e tempo final, erros de discretização - qualidade da discretização (número e distribuição dos elementos) - erro de quadratura, podem ser problemas de violações das restrições e não melhora da função objetivo.

As condições de falhas do algoritmo de otimização normalmente podem ser classificadas em: problemas de convergência, divergência, inviabilidade, ilimitação, numéricos, construção do modelo e programação. Além disso, pode não haver convergência do integrador do modelo. No processo de diagnóstico, também deve ser verificada a otimalidade da solução (Hamiltoniano constante) e análise do *NCO* - arcos singulares e não singulares. Para verificar as causas do mau funcionamento, podem ser realizados testes complementares para se certificar da correção do diagnóstico feito. Estes testes podem até mesmo ser fundamentais na solução dos casos mais complicados. Os testes complementares podem ser: caso visualização - para verificar o grau de diferença entre o planejado e realizado; caso otimalidade - para verificar o valor utópico prático da função objetivo; caso viabilidade - para verificar quando o problema é inviável; caso flexibilidade - para localizar as inviabilidades e solucioná-las; rodada de iteração a iteração - para depurar cada iteração do otimizador.

Dependendo da informação do estado e falha do otimizador e da visão geral da solução do otimizador, o usuário poderá seguir diferentes caminhos no processo de diagnóstico. Aqui, serão apresentadas as rotas de diagnóstico do *DRTO*.

Caso de solução bem sucedida

Quando o otimizador encontra uma solução ótima ou sub-ótima aceitável, esta solução precisa ser verificada. Isto porque esta solução pode não ser ótima. Para se certificar de que esta solução é ótima, devem-se efetuar algumas verificações. Primeiramente, deve-se analisar a consistência dos resultados do otimizador. Para isto, é importante monitorar os resultados históricos e seus índices de conformidade. Se a solução for "conforme", a solução é aceitável e o usuário pode explorar novas oportunidades de otimização, realizando alguns estudos de casos. Caso a solução seja "não conforme", devem-se verificar as condições de otimalidade da solução obtida. Para verificar esta solução, pode-se executar o caso otimalidade. Esta atividade não é mandatória, mas pode fornecer uma visão do valor utópico da função objetivo, ou seja, consegue-se medir a distância deste ponto ótimo, e juntamente com a análise das restrições, ter uma medida do que ainda pode ser melhorado no processo de otimização.

Também devem ser analisadas as *NCO* (*Necessary Conditions for Optimality*) para o problema de contínuo. Com isso, pode-se verificar a otimalidade da solução ótima. Além disso, também pode ser realizada a análise de sensibilidade da solução, para avaliar a robustez da solução. Há outros fatores importantes que podem invalidar a solução obtida como a inconsistência ou divergências nas condições iniciais, a inconsistência dos parâmetros do modelo e dos biases reconciliados e a baixa qualidade da discretização no uso dos métodos diretos. Todos estes fatores devem ser analisados durante a monitoração do sistema e indicar quando ocorrer algum desvio ou perturbação relevante nos resultados do otimizador. Uma vez detectada a existência de problemas, então se deve imediatamente realizar seu diagnóstico. O fluxo deste processo está esquematizado na Figura 4.23.

A verificação da consistência dos resultados do otimizador é efetuada mediante a monitoração dos dados históricos dos índices de conformidade. Estes índices utilizam as mesmas idéias do índice de conformidade definido no item 4.1.2.6 - Gerenciamento e seqüenciamento de tarefas - Monitoração de eventos da planta. Estes índices capturam as alterações ocorridas em duas soluções sucessivas do otimizador (*k* - rodada atual; *k-1* - rodada anterior). Os mesmos avaliam os aspectos referentes às condições iniciais, aos parâmetros do modelo, perfis de controle e ao valor da função objetivo. Este indicador corresponde à medida da distância (norma 2) entre duas rodadas consecutivas (d_k) do otimizador, e representa o nível de mudança da característica em questão. Desta forma os indicadores tomam a seguinte forma:

Com base na condição inicial:

$$I_{Ir} = \frac{1}{nx}\sum_{i=1}^{nx}\frac{d_k\left[x^*_{i,0,k}, x^*_{i,0,k-1}\right]+\varepsilon}{\left|x^*_{i,0,k-1}\right|+\varepsilon} \tag{4.8}$$

onde I_{Ir} é o índice de conformidade relativo das condições iniciais, $x^*_{i,0}$ é a condição inicial da variável de estado *i* (seja otimizado ou não), *nx* é o número variáveis de estado, e ε a precisão da máquina, para evitar a divisão por zero.

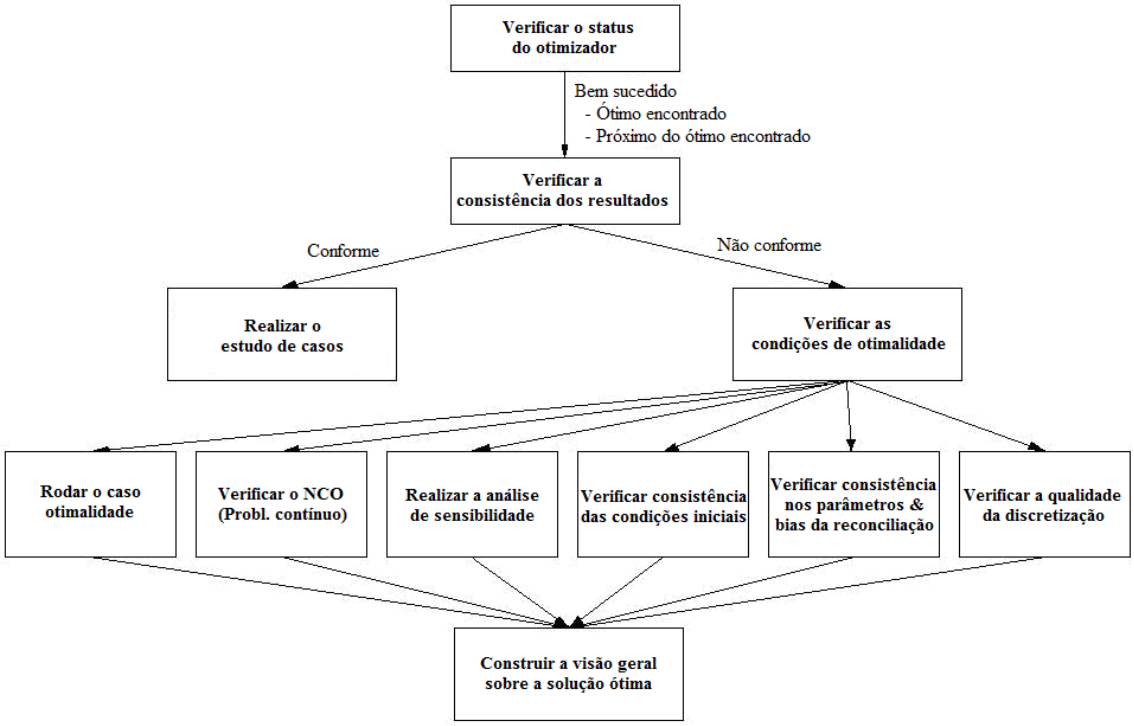


Figura 4.23 - Procedimento de verificação da consistência dos resultados do otimizador.

Com base nos parâmetros de modelo (Atualização do modelo):

$$I_{\mathrm{Pr}} = \frac{1}{np}\sum_{i=1}^{np}\frac{d_k\left[p^*_{i,k}, p^*_{i,k-1}\right]+\varepsilon}{\left|p^*_{i,k-1}\right|+\varepsilon} \tag{4.9}$$

onde I_{Pr} é o índice de conformidade relativo dos parâmetros do modelo, p^*_i é o valor do parâmetro ótimo i do modelo, e np é o número de parâmetros.

Com base nos perfis de controle:

$$I_{Cr} = \frac{1}{nu \times nt}\sum_{i=1}^{nu}\sum_{j=1}^{nt}\frac{d_k\left[u^*_{i,j,k}, u^*_{i,j,k-1}\right]+\varepsilon}{\left|u^*_{i,j,k-1}\right|+\varepsilon} \tag{4.10}$$

onde I_{Cr} é o índice de conformidade relativo dos perfis de controle, $u^*_{i,j}$ é o perfil ótimo da variável de controle ou perturbação i no instante de tempo j do horizonte de otimização, nu é o número de variáveis de controle e nt o número de instantes de tempo dentro do horizonte de otimização.

Com base no valor da função objetivo:

$$I_{Or} = \sum_{i=1}^{no}\frac{d_k\left[\Phi^*_{i,k}, \Phi^*_{i,k-1}\right]+\varepsilon}{\left|\Phi^*_{i,k-1}\right|+\varepsilon} \tag{4.11}$$

onde I_{Or} é o índice de conformidade relativo da função objetivo, Φ^*_i é um determinado objetivo *i*, e *no* é o número de objetivos. O indicador global pode ser escrito como:

$$I_{Gr} = I_{Cr} + I_{Or} \tag{4.12}$$

Lembre que na comparação das condições iniciais e nos perfis de controle devem ser atualizados para a mesma base de tempo. Isto é, as condições iniciais em *k-1* devem ser integradas até *k*, utilizando as ações de controle implementadas neste intervalo de tempo. As ações de controle das rodadas em *k-1* e *k* devem ser interpoladas em intervalos de tempos pequenos e eqüidistantes, e devem ser comparados na mesma base de tempos.

Para decidir se deve verificar a solução obtida é importante analisar o grau de conformidade da solução obtida com a anterior. Se não há alterações significativas, provavelmente a solução é ótima e não há porque avaliar a sua otimalidade, já que é uma atividade consumidora de tempo. A definição deste ponto corte depende de um acompanhamento estatístico das conformidades obtidas anteriormente. Com esta referência, pode-se obter um valor limite para o indicador (*benchmark*). Desta forma, a classificação é também baseada no valor limite e a separação sugerida é a seguinte (vide Tabela 4.7):

Tabela 4.7 – Classificação da conformidade da solução ótima.

Classe	**I_{Gr}**	**Conformidade**
Equivalente	< 10% *Benchmark*	Conforme
Levemente diferente	10% - 100% *Benchmark*	Talvez conforme
Diferente	> 100% *Benchmark*	Não conforme

Nota: São consideradas a rodada anterior ou uma base consistente. A tabela acima deve ser construída com base em I_{Gr}, I_{Ir} e I_{Pr}.

Caso esteja **conforme**, não há qualquer motivação para verificar os resultados, mas o processo pode ter algumas oportunidades de otimização oculto. O usuário pode executar o estudo de casos. Se for **talvez conforme**, o usuário deve avaliar essas informações e considerá-lo "Conforme" ou "Não conforme". Para decidir, o mesmo deve verificar mudanças nas condições iniciais, nos parâmetros de modelo, nas especificações do problema de otimização, nos parâmetros de sintonia do otimizador, ou outro ponto que a monitoração e diagnóstico indicar. E se **não estiver conforme**, o otimizador encontra uma nova solução ótima, podendo ter dois cenários possíveis: houve uma grande mudança no problema de otimização ou a solução não é ótima. Se o problema mudou, a priori, não se tem qualquer referência para decidir sobre os resultados. Se a solução não é ótima, então é necessário fazer um diagnóstico do problema encontrado. Para ambos os casos é importante realizar uma análise de condições de otimalidade. No primeiro caso é recomendável realizar testes de otimalidade (verificação do Hamiltoniano constante e das *NCO* para o problema contínuo). No segundo caso, além do teste de otimalidade, o usuário pode executar diversos testes para obter informações suficientes para diagnosticar o problema. Os testes recomendáveis são: resolver novamente o *DAOP* (*offline*) com o caso otimalidade; verificar o *NCO* para o problema contínuo; realizar a análise de sensibilidade; verificar a qualidade da discretização; além das verificações das consistências das condições iniciais, dos parâmetros do modelo e dos *biases* da reconciliação já efetuadas. Se

estas verificações não explicarem algum falso ótimo, a solução obtida pode ser considerada ótima.

No teste de otimalidade, busca-se conhecer do valor utópico da função objetivo, para obter esta informação é necessário executar o teste de otimalidade da solução (problema praticamente irrestrito). Isto é realizado, eliminando-se os efeitos das restrições que não descaracterize o problema de otimização em questão. Neste caso, são mantidas apenas as restrições de igualdade (modelo do processo), restrições de domínio e algumas restrições inerentes ao problema (ex.: volume máximo de um tanque. O problema não tem sentido se ultrapassar o volume do tanque - isso pode ser considerada restrição de domínio do problema específico).

Na verificação das condições de otimalidade (*NCO*), são analisadas as parcelas referentes à violação das restrições e à parcela da otimalidade da solução. O otimizador dinâmico sempre está buscando ativar uma restrição ou um arco singular (sensibilidade nula do Hamiltoniano com as variáveis de controle). As condições de otimalidade consistem em verificar estes fatores. Como as informações dos multiplicadores de Lagrange são obtidas do problema discretizado e os métodos diretos tendem à solução dos métodos indiretos quando os comprimentos dos elementos finitos tendem a zero, estes multiplicadores de Lagrange podem ser considerados como aproximações das variáveis adjuntas dos problemas dos métodos indiretos. Desta forma, verifica-se o *NCO* para o problema continuo através da simulação da solução ótima encontrada e re-amostrando o horizonte de tempo em tempo pequenos passos (perto do problema contínuo). Também se interpolam os multiplicadores de Lagrange (perfis adjuntos) e verificam quais variáveis de controle estão longe do ótimo. A diferença é que em vez de obter um Hamiltoniano nulo, busca-se um Hamiltoniano constante (devido a esta aproximação).

Na verificação da violação das restrições é importante avaliar as conseqüências do processo de discretização dos métodos diretos. Neste caso, divide-se o horizonte de otimização em uma quantidade de intervalos de tempos tal que o tamanho deste intervalo seja suficientemente pequeno para esta análise (ex.: um centésimo ou milésimo do horizonte de otimização). Com esta simulação, podem ser verificadas as violações das restrições ao longo das trajetórias dos estados e controles.

Para realizar a análise de sensibilidade, perturbam-se cada ação de controle e verificam-se os seus reflexos nas violações das restrições e no valor da função objetivo. Com isso, podem-se avaliar os efeitos das suas incertezas na solução ótima. Na análise de sensibilidade, temos de considerar os seguintes aspectos da solução ótima: Viabilidade – Regular o processo; Otimalidade – otimizar o processo.

A verificação das consistências das condições iniciais leva em conta três aspectos podem perturbar estas condições sob o ponto de vista do otimizador. São elas: modelos de processo diferentes usados pelo estimador de estados e otimizador; método numérico utilizado no estimador de estados não reproduz a tarefa *NLA* do otimizador ou não consideram as equações de conservação na obtenção da solução; e perturbações nos parâmetros do modelo ou na reconciliação de dados. As conseqüências destes fatores é o aparecimento de diferenças nas derivadas iniciais das variáveis de estado e nas posições das variáveis algébricas em relação aos encontrados na solução do *NLA* pelo otimizador.

Isto pode resultar em um problema de otimização inviável. Se a condição inicial não for viável, o otimizador não vai encontrar solução para o problema. Porém, corre-se o risco desta informação ser falsa. Neste caso, deve-se realizar o diagnóstico com mais cuidado. No caso contrário, os estados equivocados também podem perturbar os resultados do otimizador desnecessariamente.

Na verificação das consistências dos parâmetros do modelo e dos *biases* da reconciliação, deve-se levar em conta a existência de parâmetros que podem ser alterados lentamente e outros rapidamente. O usuário precisa para classificá-los, observar as tendências e validar esses valores (acreditar ou não). Com base nesta análise, pode-se decidir se existem parâmetros não-confiáveis. O mesmo acontece com os *biases* da reconciliação de dados. Geralmente, as evoluções de desvios de instrumentos são muito lentas. Apenas os procedimentos de calibração ou alguns incidentes podem desviar instrumentos de forma rápida. Neste tópico, a monitoração dos parâmetros fornece informações valiosas para o diagnóstico do problema.

Finalmente, na verificação da qualidade da discretização, deve-se ter em mente que os resultados de uma discretização inadequada podem afastar os perfis de controle da solução ótima. Uma discretização ruim pode causar a violação de restrições que o otimizador não consegue perceber na solução do problema discreto.

Caso de problema inviável

Para realizar esta tarefa de forma eficiente é necessário primeiramente fazer um diagnóstico do problema utilizando as informações fornecidas pelo otimizador, pois reduzirá o número de alternativas a serem testadas. Para isso, é preciso obter a visão geral sobre a inviabilidade. A inviabilidade pode acontecer devido aos seguintes critérios que não estão satisfeitos: violação de restrição ou condições de complementaridade. Para obter informações sobre violações das restrições, precisamos classificar e definir o cenário na qual ocorreu cada violação de restrição. Define-se a violação de restrição no problema contínuo como a intensidade e a distribuição de qualquer restrição ao longo de todo o horizonte de otimização. Para ser mais preciso nesta avaliação, é necessário que seja executada uma simulação utilizando os perfis ótimos de controle sugeridos ou interpolar os perfis de controles e estados encontrados pelo otimizador. Deve-se então re-amostrar o horizonte de otimização em intervalos de tempo pequenos e eqüidistantes e calcular as violações das restrições em cada instante. Depois são verificados o tipo e a natureza de cada restrição que foi violada. Usando este mapa, o usuário pode decidir sobre o caminho a ser investigado. Às vezes, a principal causa é óbvia e pode ser facilmente diagnosticada, mas há situações em que é quase impossível encontrar a restrição problemática sem o uso de qualquer tipo de ferramenta de diagnóstico.

No passo seguinte, é necessário descobrir se o processo apenas começou inviável ou tornou-se inviável ao longo do horizonte de otimização. Sendo que a viabilidade do problema de otimização dinâmica pode ser comprometida por quatro razões básicas: condições iniciais inconsistentes, condições iniciais inviáveis, condições intermediárias inviáveis e especificações de restrições contraditórias.

As condições iniciais do processo fornecidas por um estimador de estados devem ser consistentes. Esta estimação pode ser feita através da utilização de ferramentas de estimação (ex.: simuladores dinâmicos, estimadores do tipo *MHE* ou *CEKF*) ou até mesmo através do fornecimento direto dos valores das variáveis de estado (diferenciais e algébricos) e de controle. Porém estas condições iniciais podem não ser consistentes do ponto de vista do modelo de processo utilizado pelo otimizador dinâmico. Isto pode ocorrer quando a estimação de estados utiliza um modelo do processo diferente do utilizado pelo otimizador, seja por imprecisão na solução do estimador, por um sistema algébrico-diferencial diferente, ou até mesmo a ferramenta de estimação não necessita respeitar as restrições do sistema algébrico-diferencial para estimar o estado inicial da planta.

Se as condições iniciais consistentes, segundo o ponto de vista do otimizador dinâmico, não são concordantes com as condições e tendências observadas pelo estimador de estados, o otimizador pode estar procurando otimizar um processo diferente da planta real. Este fato pode comprometer a eficácia do otimizador, por começar de um ponto operacional diferente do obtido pelo estimador de estados. Devido a este fato, é importante verificar se as condições iniciais fornecidas pelo estimador de estados são consistentes segundo o ponto de vista do otimizador. Se não forem consistentes, a execução do módulo de otimização fica invalidada ou comprometida. Esta decisão poderá ser tomada pelo usuário do otimizador. Com o objetivo de auxiliar o usuário nesta decisão, a ferramenta de diagnóstico deverá fornecer a informação da não conformidade entre as condições iniciais fornecidas pelo estimador em relação às usadas pelo otimizador. Na realidade esta avaliação não fornece a informação de quais condições iniciais são corretas, mas sim uma analise comparativa entre duas soluções concorrentes.

Há situações onde esta diferença pode não ter importantes conseqüências, porém quando o processo está próximo do seu limite (principalmente após alguns ciclos do otimizador), esta incoerência entre o estimador e o otimizador pode levar à inviabilidade da solução do otimizador, perdendo uma oportunidade de obter os benefícios potenciais de otimização. Duas situações básicas podem levar a esta perda. Quando os valores das variáveis algébricas calculadas pela inicialização do otimizador estiverem indevidamente fora dos limites de operação ou quando as derivadas das variáveis de estado estiverem apontando no sentido de violar as restrições, e levam processo a violar as restrições antes das correções dos controles. Além disso, os valores das variáveis de estado podem estar indevidamente dentro ou fora das restrições devido à incoerência entre a estimação de estados e a consistência das condições iniciais.

A ferramenta de diagnóstico deve evidenciar quando as condições iniciais são viáveis ou inviáveis para os valores do estimador e otimizador, além disso, também deve apresentar as diferenças absolutas e relativas entre os valores, tomando como referência os dados do otimizador. Também devem ser apresentados os valores das derivadas das variáveis de estado e sua tendência a violar as restrições ou não. Com estas informações, o usuário pode decidir se estas condições iniciais são aproveitáveis ou não.

Independentemente da existência de incoerências entre as condições iniciais dadas pelo estimador e as calculadas pelo otimizador, as condições iniciais podem ser inviáveis ou tendem inexoravelmente à inviabilidade. Se isto acontecer no início do horizonte do

problema de otimização, certamente não existirá solução viável para tal problema e o otimizador só poderá ser executado com sucesso quando tal situação for extinta. Esta é uma situação que pode ocorrer numa planta real, e pode se transformar em um fator de insucesso da utilização de um otimizador dinâmico, pois há situações onde se altera o problema de otimização e o mesmo passa a partir de um problema inviável.

A ferramenta de diagnóstico deve evidenciar quando as condições iniciais são viáveis, e devem acusar quando as variáveis de estado ou sua derivada leva a um problema inviável. Com estas informações, o usuário pode decidir quando estas condições iniciais se tornam inviáveis.

Há casos onde a violação de restrições deve ser admitida por um curto intervalo de tempo. Isto pode ser resolvido utilizando uma formulação de problema de otimização em múltiplos estágios onde o objetivo é regular o processo, trazendo as variáveis para dentro das restrições no mínimo tempo, e minimizando a integral das violações das restrições. Isto é, o problema passa a ser de controle no primeiro estagio do otimizador. Neste caso, as restrições passam a ser *soft constraints*. Uma vez que o processo estiver regulado, passa-se a otimizar normalmente o processo.

Também há situações onde as condições iniciais são viáveis, porem as direções das derivadas iniciais leva o processo inexoravelmente para uma condição inviável. Isto é, não há manipulações possíveis das variáveis de controle que evitem as violações das restrições do processo ao longo do tempo. Neste caso, deve-se utilizar a mesma estratégia do caso de violação das condições iniciais.

A detecção da inviabilidade inicial será feita através da comparação das posições das variáveis de estado e de controle em relação aos limites de especificação estabelecidos para os mesmos. Quanto à tendência à violação, analisam-se as direções das variáveis de controle que evitam a inviabilidade e simula-se o processo utilizando a máxima manipulação possível.

Uma maneira eficiente efetuar o diagnóstico da inviabilidade é a solução do problema de otimização flexibilizando as restrições (de forma *offline*). Neste caso, reformula-se o problema relaxando as restrições e resolvendo este novo *DAOP* nas mesmas condições que o otimizador *online* falhou. Desta forma, o otimizador *offline* deverá resolver o problema, minimizando as alterações nas restrições do problema e fornecendo uma visão clara das restrições problemáticas da rodada em questão. Com isso, fica mais fácil imprimir uma ação corretiva no problema. Lembre também, que esta atividade também é custosa e deve ser realizada quando o diagnóstico não é óbvio. Se o problema continuar inviável, constata-se que o problema seria definitivamente inviável. O coração da análise de inviabilidades é a solução do problema relaxado, sendo que as outras atividades aqui citadas são utilizadas para analisar o problema e conduzir a este teste e dar sustentação à solução flexibilizada, através da verificação da coerência da solução relaxada proposta com as informações fornecidas pelo algoritmo de otimização no problema original. Este assunto é discutido em detalhes na parte 2 desta tese.

Há situações onde a receita do processo é alterada entre um estágio e outro, ou ocorre uma perturbação importante, que obriga a operação alterar abruptamente a receita do processo.

Neste momento, o problema pode se tornar inviável (intencionalmente ou não). Portanto, da mesma forma que seria tratada a inicialização do problema em condições inviáveis, no momento do horizonte de otimização onde esta forte descontinuidade ocorrer (ex.: transição de especificação de produção) um novo estágio é criado onde o objetivo também é regular o processo, trazendo as variáveis para dentro das restrições no mínimo tempo.

Do mesmo modo, há também situações onde as condições intermediárias são viáveis, porem as direções das derivadas nas transições de estágios leva o processo inexoravelmente para uma condição inviável. Neste caso, também se deve utilizar a mesma estratégia do caso de violação das condições iniciais.

A forma de evidenciar tais inviabilidades momentâneas seria idêntica a adotada na inicialização do otimizador. Ou seja, ter-se-ia um mapa da inviabilidade do problema ao longo do horizonte de otimização. Para chegar a esta conclusão é necessário re-executar o otimizador em dois modos, são eles: modo simulação e modo viabilidade dos perfis de controle. No primeiro modo, simplesmente simula-se o processo utilizando os perfis iniciais de controle fornecidos. No segundo, executa-se o otimizador anulando a função objetivo, procurando alguma solução viável alternativa.

Há casos onde ocorrem conflitos entre as especificações de processo, isto é, as restrições concorrem entre si, fazendo com que o problema nunca haja solução viável. Neste caso, pode-se ter a situação de uma restrição inadmissível (quando for impossível respeitar as equações ou pelos limites de controle - ex.: evaporação da água à temperatura máxima de $40^{o}C$ na pressão atmosférica) ou restrições conflitantes (quando for impossível satisfazer algumas restrições ao mesmo tempo). Sendo que, as restrições conflitantes não são fáceis de serem identificadas, principalmente em problemas de grandes dimensões. A detecção é feita da mesma forma que o problema de inviabilidade inicial e intermediária. Uma vez que seja constatado que não é problema de inviabilidade momentânea.

Além disso, da mesma forma anterior, pode-se executar o otimizador anulando a função objetivo, procurando alguma solução viável. Se a solução for viável, então não há restrições concorrentes. Caso contrário, uma análise conjunta das soluções do problema sem restrições adicionais e do problema de viabilidade permite identificar as restrições problemáticas. Com estas informações, o usuário pode analisar o problema de otimização do ponto de vista da engenharia de processos e eliminar os conflitos entre restrições.

Neste caso, deve-se priorizar e abandonar algumas restrições visando eliminar a incompatibilidade das restrições. Uma alternativa é flexibilizar as restrições do problema de otimização, da mesma forma da inviabilidade inicial ou intermediária. Devem-se estabelecer as prioridades entre as restrições concorrentes, alterando o objetivo das restrições concorrentes menos prioritárias. Lembre que não há um caminho único para realizar o diagnóstico da otimização dinâmica. A natureza do problema encontrado é que irá levar à escolha pela estratégia de relaxamento das restrições ou não.

Para encontrar as causas principais da inviabilidade, pode-se executar um conjunto de testes (vide Figura 4.24). São elas: verificar a viabilidade de inicial e intermediária; executar o caso de viabilidade; realizar a análise de sensibilidade para restrições; verificar a consistência nas condições iniciais; verificar a consistência nos parâmetros e bias da

reconciliação; verificar os procedimentos de discretização, classificar as restrições problemáticas e rodar o caso com restrições relaxadas.

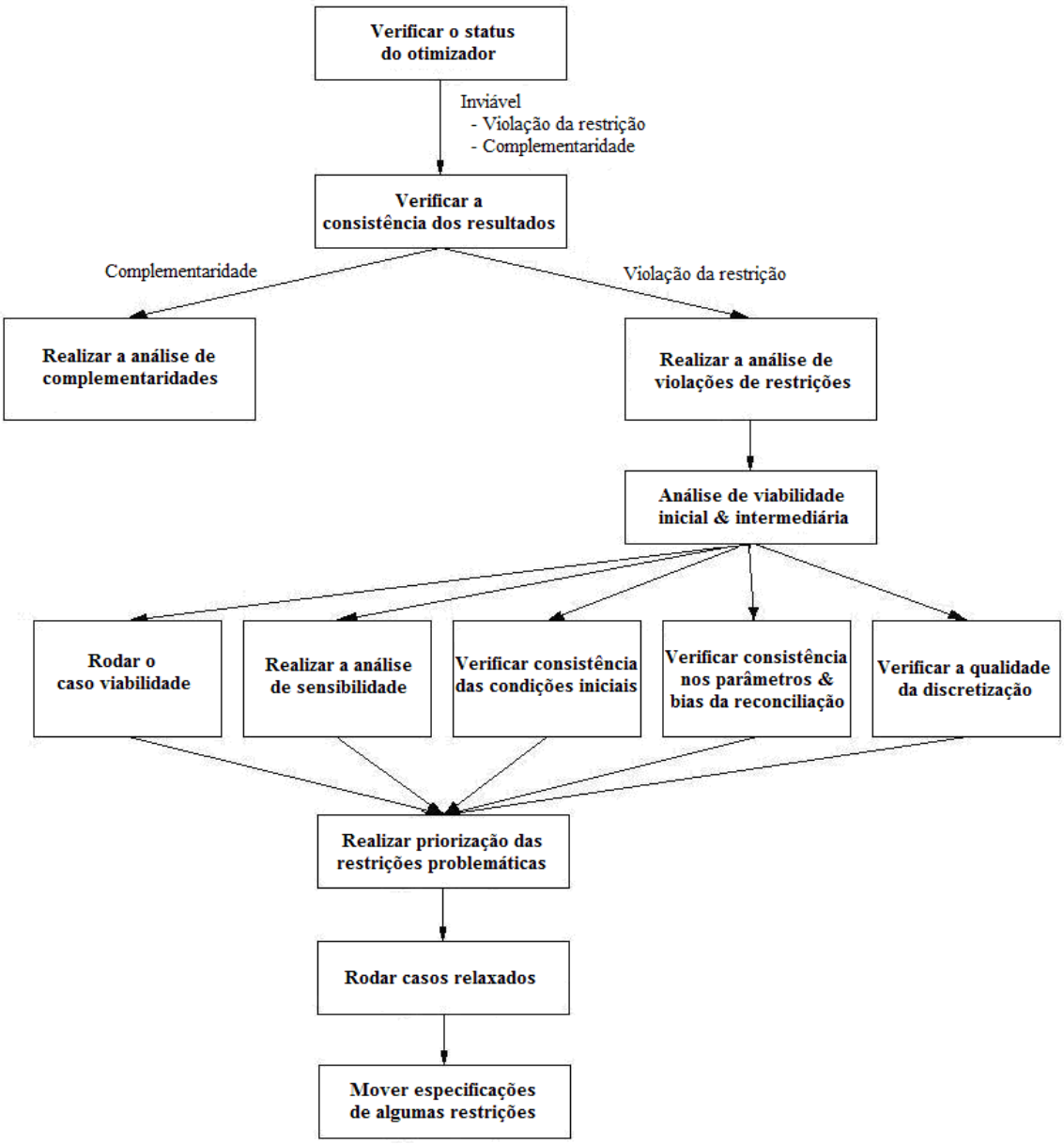


Figura 4.24 – Procedimento de verificação da inviabilidade do problema de otimização.

Para analisar a viabilidade de inicial e intermediária, deve-se verificar se a condição inicial está fora da região viável. Verificar se é possível evitar a inviabilidade utilizando toda a faixa de operação de variáveis de controle. Detectar descontinuidades intermediárias nas especificações de restrição e verificar se as condições nesses pontos intermediários são viáveis ou tendem necessariamente para a inviabilidade. Também pode ser executado o caso de viabilidade. Com o objetivo de encontrar alguns problemas relacionados à função objetivo ou convexidade, pode-se anular a função objetivo e verificar se é possível regular o processo, ou seja, transforma um problema de otimização em um problema de controle regulatório. Se obtiver uma solução bem sucedida, então não haverá restrições conflitantes

ou especificações equivocadas. Neste caso, podem estar ocorrendo problemas de convergência.

Se não conseguir solução bem sucedida, provavelmente têm-se especificações equivocadas ou restrições conflitantes. Esta hipótese pode não ser verdade, sendo necessário reduzir a chance de falsos negativos. Para este caso, é importante executar o problema de otimização (de forma offline) flexibilizando as restrições do processo, sendo necessário reformular o problema na sua forma relaxada, como apresentado na parte 2 desta monografia. Outra opção é a realização da classificação das restrições problemáticas e a identificação das causas de tal problema. Se não for possível podem-se executar casos de viabilidade alternativos, removendo as restrições com base no ranking acima para obter uma solução viável.

Quando se encontrar uma solução viável, podem-se inspecionar essas prováveis restrições problemáticas. Neste caso, pode-se realizar uma análise de sensibilidade e tentar encontrar uma solução viável, uma possível movimentação da especificação de restrições. Obtendo sucesso, podem-se colocar novamente as outras especificações de restrições e verificar se continua a obter a solução bem-sucedida. Se não, deve-se de repetir o procedimento acima para obter uma solução viável com novas especificações. Depois de concluir este procedimento, pode executar novamente o caso de otimização com a nova especificação do problema. Se obtiver uma solução ótima, deve-se ter o caso bem sucedido. Caso contrário, a causa é provavelmente o problema de "convergência".

Este é um processo exaustivo que nem sempre vale a pena ser realizado. Isto é eficaz para problemas pequenos, pois não exigem muito esforço do usuário. Porém, se esta questão for fundamental para resolver o problema encontrado, este procedimento deve ser executado mesmo assim.

Caso de problemas de convergência

Problemas de convergência são inerentes aos algoritmos de otimização, porém *DAOP's* mal postos podem causar este tipo de problema. Às vezes é difícil distinguir problemas de convergência da inviabilidade, pois o processo de convergência do algoritmo de otimização passa por eliminar as violações das restrições.

Para diagnosticar os problemas de convergência, é necessário avaliar as condições de execução das tarefas deste tipo no algoritmo *NLP* (vide apêndice B). Para avaliar as condições da solução do problema de *NLP*, é necessário analisar cada critério de convergência do algoritmo, bem como todas as tarefas realizadas pelo algoritmo que levarão a esta convergência. Nesta monografia é considerada a utilização do algoritmo *IPOPT* (aqui são apenas consideradas as utilizações dos métodos de otimização determinísticos baseados em gradiente). Os algoritmos determinísticos executam as seguintes tarefas ao resolver um problema de otimização: cálculo da direção de busca, cálculo do tamanho do passo e verificação de convergência.

A seguir são apresentados de forma resumida os critérios de convergência e tarefas executadas pelo algoritmo *IPOPT*. De forma resumida, deve-se verificar a convergência do algoritmo através da análise da inviabilidade *primal* (violação das restrições), inviabilidade

dual (violação das restrições das variáveis de folga), complementaridade (condição a serem satisfeitas para as desigualdades) e erro global do *NLP* (norma infinita destes critérios juntos). Das tarefas executadas pelo algoritmo, que levam a esta convergência, pode-se citar: a atualização da Hessiana (matriz de *KKT - Karush-Kuhn-Tucker* - condição de otimalidade), atualização do parâmetro de barreira, cálculo da direção de busca, cálculo do tamanho do passo, e da avaliação do ponto tentativa (proposto). Basicamente, deve-se verificar se a matriz Hessiana é não singular, positiva definida, ou bem condicionada (pouca inércia). O parâmetro de barreira deve se reduzir ao se aproximar da solução ótima. No cômputo da direção de busca, as inviabilidades devem ser reduzidas. No cálculo do tamanho do passo e aceitação do ponto tentativa, as violações das restrições devem ser reduzidas e deve haver suficiente progresso no valor da função objetivo. As verificações de todas estas tarefas devem ser efetuadas utilizando o nível 3 da ferramenta de diagnóstico.

Há casos onde o comportamento da Hessiana está contaminado pela estrutura do modelo do processo e isso deve ser corrigido. Uma opção para resolver este problema seria mudar a estrutura do modelo do processo, ou reformular as restrições do problema de otimização. Outra opção interessante é a adoção de algoritmos de otimização que utilizam a Hessiana reduzida, pois minimizam os fatores relativos ao modelo do processo, ficando somente as parcelas referentes à otimização do processo.

4.2.3.2 Análise de processo do otimizador

Durante o procedimento de diagnóstico, pode-se complementar esta atividade com a realização de algumas análises de sensibilidades para avaliar a robustez da solução ou até mesmo aprimorar a sintonia do otimizador. Para realizar a análise de robustez do otimizador, o usuário pode realizar testes de sensibilidades da função objetivo e das violações das restrições com perturbações nas variáveis de controle, condições iniciais, nas próprias restrições e nos parâmetros do modelo. Nestes dois últimos inclusive podem ser identificados as restrições e os parâmetros que mais afetam a solução do problema, e assim mitigar seus efeitos (ex.: melhorando a precisão de um determinado parâmetro). Estes testes podem ser efetuados de duas formas, simplesmente simulando o processo mediante as perturbações planejadas ou perturbar os fatores (ex.: parâmetro, posição da restrição) e resolver *offline* o *DAOP* e analisar seus efeitos na solução ótima. Para prover maior robustez e desempenho ao otimizador, pode-se aprimorar sua sintonia, e realizar a análise de sensibilidade dos indicadores de desempenho do otimizador com alterações dos seus parâmetros de sintonia. A escolha do par adequado deve ser acompanhada de uma análise prévia do desempenho comprometido e da escolha do possível parâmetro de sintonia mais eficaz para resolver tal problema.

A análise das informações do otimizador, através da ferramenta de diagnóstico, pode ser melhorada resolvendo novamente o *DAOP* no *EMSO* (de forma *offline*). Da mesma forma do caso anterior, no lugar de simplesmente simular o processo, executa-se novamente a otimização dinâmica com a mesma base de informações da otimização *online*. Com isso, pode-se verificar cada iteração através de paradas momentâneas da otimização e visualizar os resultados e as informações do otimizador. Após a parada, o analista poderá continuar até a próxima iteração ou até mesmo encerrar a rodada. Lembre que esta é uma atividade muito árdua e demorada. Portanto deve ser realizada somente em casos de difícil diagnóstico.

Análise de robustez e desempenho

Lembre que os métodos diretos de solução de *DAOP* passam por uma discretização e posterior solução de um *NLP*. Portanto, todas as informações fornecidas pelo otimizador são referentes ao *NLP* (que são informações muito úteis), e sofrem as conseqüências das aproximações realizadas. E no caso de discretização total, acrescenta o fato de a solução viável poder ser obtida somente no final do processo de otimização. Com isso, a atividade de visualização dos resultados do otimizador através do *EMSO*, passa a ser uma ferramenta complementar importante no processo de diagnóstico. Nesta visualização, o *EMSO* faz a leitura das informações do *DAOP* em questão, das condições iniciais e os perfis de controle gerados pelo otimizador. Com isso, o processo pode ser simulado, podendo ser verificada a viabilidade do problema contínuo. Além disso, também podem ser avaliados os erros de discretização, no uso do método *single-shooting*, *multi-shooting* e discretização total.

Análise de oportunidades

A visualização dos resultados do otimizador através do *EMSO* também pode ser utilizada para avaliar as diferenças entre a receita proposta pelo otimizador e a receita executada na operação. Com esta ferramenta, pode-se verificar e até mesmo antecipar a detecção de violações de restrições na planta real. Além disso, pode-se medir o grau de conformidade da operação com a receita do otimizador, e com isso, fazer uma análise crítica de seus resultados e da qualidade da otimização da planta.

A análise da qualidade da solução do *DAOP* pode ser efetuada através de perturbações nas entradas e verificar como a solução ótima é afetada pela perturbação nos estados, parâmetros e restrições. Esta análise é efetuada através realização da simulação *offline* e utilizando a solução proposta pelo otimizador, procuram-se as entradas mais sensíveis com a função objetivo e restrições com as entradas do sistema. Estas entradas são as possíveis incertezas e perturbações a serem analisadas, verificando-se os seus efeitos. As incertezas e perturbações podem ser originadas da estimação de estados (condições iniciais consistentes), da estimação de parâmetros e da implementação da solução (perfis de controle).

Outra modalidade de análise de sensibilidade é a verificação dos efeitos das incertezas sobre a função objetivo, viabilidade das restrições e perfis ótimos de controle. Isto é efetuado através da realização da otimização dinâmica, considerando incertezas nas entradas do problema, tais como: perturbações no estado inicial, nos parâmetros e nas restrições.

4.3 Análise crítica e contribuições para os sistemas de DRTO

É proposta neste trabalho, uma estrutura de *DRTO* aplicável à otimização dinâmica processos em bateladas e semi-bateladas, e que possibilita uma maior integração da camada de controle e otimização com as áreas de planejamento e programação de produção. Este ferramenta tem a capacidade de otimizar dinamicamente diferentes receitas de produção, processos de transições de processo na produção (campanhas), alterações de ordens de produção e quando há manobras operacionais. Esta arquitetura proposta permite

a realização da otimização dinâmica tanto no modo *online* como *offline* de forma integrada, onde se pode resgatar qualquer rodada *online* e repeti-la de forma *offline* sem qualquer esforço adicional. Como não se tem conhecimento da existência deste tipo de ferramenta, conclui-se que as soluções projetadas até hoje consistem de sistemas com funcionalidades parciais e são adaptadas para cada caso aplicado. A única solução comercial que tem algumas facilidades comuns a esta proposta é o sistema da *IPCOS* (empresa Holandesa de *NMPC* para processos em batelada). Porém, ela só tem o módulo de controle *online* adaptado para processos em batelada. Apesar de esta solução ser utilizada como *NMPC*, neste modo, a mesma não apresenta sensíveis vantagens em relação às soluções existentes para esta função.

A seguir é efetuada uma análise das seguintes contribuições na proposta do sistema de *DRTO*:

1) Otimização dinâmica *online* e *offline* integrados, com funcionalidades comuns;
2) Resgate e repetição qualquer rodada *online*;
3) Estimação de estados genérica (Ajuste de parâmetros, reconciliação, estimação de estados) com o mesmo solver de otimização dinâmica;
4) Disparador agrega eventos, análise de resultados (não executa sem necessidade), usa critério de otimalidade e viabilidade;
5) Interrupção da otimização no meio da execução (re-início sem terminar a execução);
6) Utilização de toda a infra-estrutura de modelagem do *EMSO* e incorporação de uma linguagem de *DAOP* de modelagem orientada a objetos;
7) Interpretador transforma qualquer formato de modelagem em *DAOP* padrão automaticamente;
8) Visualização dos resultados do *DAOP* no formato estruturado no ambiente *EMSO* (Visualização numérica e gráfica);
9) Atribuição de variáveis para reduzir as dimensões do *DAOP* e redução de não linearidade;
10) Manipulação adequada das variáveis de controle de forma automática e segura (suave e com controle de taxa de variação);
11) Execução da otimização independentemente do ajuste de parâmetros e reconciliação de dados (diferente dos *RTO's*);
12) Realização do ajuste de parâmetros e reconciliação de dados de forma seqüencial ou simultânea que pode ser alterado *online*;
13) Agregação da análise estatística, diagnóstico e discriminante dos ajustes de parâmetros;
14) Utilização da avaliação de resultados adaptado para otimização dinâmica (semelhante à do *RTO*);
15) Inclusão do uso da análise de otimalidade baseada no Hamiltoniano constante;
16) Opções de métodos de solução de *DAOP single-shooting*, *multi-shooting* e colocação em elementos finitos, com escolha *online*;
17) Montagem modular das avaliações de funções para o *NLP*, podendo agregar funcionalidades como: adaptação de malhas (elementos finitos moveis e *wavelets*), problemas simples e multi-estágios, detecção da estrutura da solução e agrupamento de elementos;
18) Ferramenta de monitoração e diagnóstico de *DRTO*, com diferentes níveis de detalhes para análise de falhas, inviabilidades e desempenho (incluindo novas métricas);
19) Mecanismo de depuração da solução *offline* iteração-a-iteração e avaliação do erro de discretização do *DAOP*;
20) Metodologia de diagnóstico;

A estrutura do *DRTO* proposto aqui contém os módulos de construção do *DAOP*, atualização de modelos, solução de *DAOP's*, avaliação e implementação de resultados, gerenciamento e seqüenciamento de tarefas, e monitoração e diagnóstico do sistema de *DRTO*. Esta estrutura é diferente das estruturas já propostos anteriormente, pois as soluções alternativas tratam de forma genérica a estimação de estados, o mecanismo de disparo só leva em conta os critérios de otimalidade e viabilidade do *DAOP*, não contemplam a análise de resultados do otimizador e nem as interações do *DRTO* com os operadores (só permitem uma estrutura fixa do *DAOP*).

Na estrutura de *DRTO* proposto, grande parte das funcionalidades do sistema *online* é comum à aplicação *offline*, de forma que seja possível reproduzir no modo *offline*, situações vividas no modo *online*. Isto permite diagnosticar algum problema na aplicação *online*.

O *EMSO*, com a sua *IHM*, é utilizado na construção do modelo de otimização dinâmica. Para isso, foi proposta um acréscimo na *EML*, de forma a lidar com modelagem de *DAOP's*. Esta linguagem permite que se formule o problema de otimização na forma de memória de cálculo em engenharia química, ou seja, apresentar as equações e declarações que permita escrever o modelo de *DAOP* de forma intuitiva e fácil de interpretar. Isto porque se propõe a construção de um interpretador que transforma o *DAOP* da forma escrita pelo engenheiro para um formato padrão, onde o otimizador interpreta o problema. Com isso, o usuário não precisa se restringir a apresentação do *DAOP* no formato padrão. Outros softwares como o *gPROMS*, *DyOS* e *DynoPC* são restritivos em relação a este ponto. Esta linguagem proposta permite que seja construída toda a diversidade de *DAOP's*, contemplando problemas com simples e múltiplos estágios, com mais de um objetivo, e restrições de caminho, pontos interiores e terminais, assim como tempos finais livres para serem otimizados. Estes últimos pontos, também podem ser encontrados no *DyOS* e *DynoPC*.

Na atividade de construção de modelos, o otimizador permite que sejam importados dados do otimizador *online*, tais como: parâmetros atualizados do modelo do processo, condições iniciais da planta, informações da última reconciliação de dados, estrutura do *DAOP* atualizada (com a receita a ser otimizada). Desta forma, o otimizador pode reproduzir um caso real com dados obtidos diretamente do otimizador *online*. As soluções existentes atualmente realizam estas operações de forma parcial, sendo necessário um grande esforço de edição no modelo de otimização para reproduzir um caso real da planta. Além disso, esta facilidade permite que sejam visualizadas e simuladas com precisão as receitas ótimas propostas pelo otimizador *online*. Neste caso, são utilizados os mesmo recursos de navegação, visualização de dados e representação gráfica de todas as variáveis do *DAOP* em questão.

A estrutura proposta também contempla algumas facilidades simples, mas proporcionam benefícios sensíveis ao sistema. Pode-se utilizar um mecanismo de atribuições de variáveis (*assign*) com o objetivo de reduzir o número de variáveis e equações do *DAOP* efetivamente resolvido. Isto faz com que o esforço computacional para resolver o problema seja menor, além de poder proporcionar a redução de não linearidades por agregação de equações do modelo. Outra facilidade simples, porém eficiente é a utilização das

manipulações dos controles da mesma forma efetuadas na operação de plantas reais (através de ações de controle em degraus). Em muitos casos, as ações de controle devem partir do ponto de operação corrente da planta e tem suas velocidades de alterações limitadas. Isto é reproduzido no otimizador através de manipulações de Δu e transformando as manipuladas reais do processo em variável de estado u. Isso equivale a um perfil linear por partes na variável manipulada real da planta. Como resultado, tem-se uma política realista de otimização, além de suavizar a solução ótima. Também relacionado às definições das variáveis de controle, foi conceituada uma forma de análise de especificações das variáveis de controle, baseado nas informações de singularidade do sistema *DAE*, bem como na utilização de análises de grafos e uso do algoritmo de Tarjan adaptado para auxiliar na escolha das variáveis de controle do *DAOP* (não foi proposto método).

A estrutura proposta contempla a calibração do modelo construído, através da *SE_SOLVER.DLL*. Esta *DLL* realiza o ajuste de parâmetros do modelo do processo, detecção e eliminação de erros grosseiros e reconciliação de dados, e estimação de estados do processo. A proposta é a utilização da formulação do *MHE* na forma de *DAOP*. Desta maneira, pode ser utilizada uma formulação única para ajuste de parâmetros, reconciliação e estimação de estados. Sendo que estas tarefas podem ser executadas utilizando o próprio otimizador dinâmico. Periodicamente, os parâmetros, os *biases* dos instrumentos e as condições iniciais devem ser atualizados com base nas informações da planta real. Sugere-se aqui, uma estrutura que permita que esta tarefa seja realizada seqüencial ou simultaneamente. A decisão é dependente do problema. Esta solução proporciona maior flexibilidade ao sistema, pois as correções podem ser efetuadas em tempos diferentes. E dependendo do caso, proporciona mais robustez e desempenho ao sistema.

Como parte do processo de calibração do modelo, é efetuada a análise estatística da calibração para avaliar a fidelidade do modelo à planta. Nesta análise, é avaliada a qualidade do ajuste (através da matriz de covariância dos parâmetros), a qualidade do modelo (através dos intervalos de confianças das predições), discriminação entre modelos (para avaliar se os parâmetros devem ser atualizados), e efetuado o diagnóstico da atualização do modelo (utilizando o método de condicionamento da matriz de covariância dos parâmetros desenvolvido por Miletic e Marlin, 1998).

A maior virtude desta proposta é a realização, de forma automática, deste tipo de análise e fornecer estas informações ao usuário e ao módulo da avaliação de resultados do otimizador. Da mesma forma, a análise dos *biases* das reconciliações são fornecidos ao gerenciador do *DRTO*, para as atualizações de estados e implementações das ações de controle compensadas pelos *biases*. Além disso, esta estrutura proposta permite avaliar a avaliar as significâncias das mudanças nos parâmetros do modelo e verificar as coerências das estimações, e também a coerência entre o estimador de estados com as condições iniciais do *DAOP*.

Na otimização *online*, esta tarefa é executada por uma aplicação *SE_DRTO.EXE*, que resolver periodicamente o problema de atualização de estado. Esta tarefa pode levar alguns minutos para ser concluída, e também pode falhar. Em ambos os casos, o gerenciador deve atualizar as condições iniciais para o momento do disparo do módulo de otimização. Desta forma, o mesmo necessita integrar o modelo com os últimos dados confiáveis da planta.

Isto faz com que não seja obrigatório estimar o estado de forma sincronizada com o módulo de otimização. Caso a estrutura do problema seja radicalmente alterada, a rodada seguinte do otimizador fica condicionada à estimação de estados. Isto é coordenado pelo gerenciador de tarefas.

Apesar destas técnicas serem bem difundidas, não se tem conhecimento (na literatura aberta) desta forma de utilização e automação da análise. Isto faz com que esta aplicação proporcione mais robustez e confiabilidade ao *DRTO*.

O mecanismo de monitoração da planta e de disparo do otimizador são tarefas importantes na otimização *online*. Esta tarefa é realizada pelo gerenciador do *DRTO* (*DRTO_Manager.EXE*), que é executado de forma cíclica e com freqüência bem mais elevada do que as aplicações de estimação de estados (*SE_DRTO.EXE*) e de otimização dinâmica (*OPT_DRTO.EXE*). Esta aplicação monitora o processo e verifica a ocorrência de eventos (ex.: alterações na estrutura do problema e perturbações na planta) que invalidam a última receita ótima. Estes tipos de eventos devem disparar as atividades de atualização do estado e solução do problema de otimização devido às mudanças na estrutura do *DAOP* ou por alterações na solução ótima (perda de otimalidade ou viabilidade). O diferencial desta proposta é que a abordagem existente (Kadam et al., 2003) só considera a otimalidade da solução, sendo esta, função da violação da tolerância da variação da solução do problema *DAOP* linearizado no ponto. Este critério é relativo e menos robusto do que a alternativa proposta, utilizando o critério proposto por Tanartkit e Biegler (1997) de Hamiltoniano constante ao longo do horizonte de otimização, que é um critério absoluto.

No módulo de solução de *DAOP*, propõem-se a utilização dos três métodos diretos de solução de *DAOP* mais populares (método *single-shooting*, *multi-shooting* e discretização total - colocação em elementos finitos). Isto foi proposto porque não há um consenso na literatura aberta sobre qual método é mais eficiente. Na realidade a escolha é dependente do problema, e deve ser feita considerando os pontos apresentados na análise comparativa da revisão bibliográfica. A montagem dos vetores de resíduos das funções, Jacobianas e Hessianas foram feitas de forma modular no processo de discretização (*DAOP2NLP.DLL*), onde as estruturas no âmbito do problema de *NLP* e no estágio são as mesmas para todos os métodos, diferenciando-se apenas no elemento finito. Com esta modularidade, fica mais fácil implementar e mudar de algoritmo.

Propõe-se neste módulo (*DAOP_SOLVER.DLL*) a inclusão de algumas facilidades para refinar a solução obtida pelo otimizador. A primeira delas é a adaptação de malhas discretas. Há duas abordagens distintas na literatura aberta. Uma tradicional, baseada em elementos finitos móveis e a outra baseada em *wavelets*. Ambas as abordagens requerem um custo computacional elevado, e o usuário deve decidir se deseja utilizar ou não. Biegler et al. (2001) propuseram um método mais eficiente e robusto de adaptação com elementos finitos móveis e a abordagem de *wavelets* adotada por Marquardt (Binder et al., 1997) tem características interessantes, porém o critério de parada é fraco (variação da solução ótima). Apesar desta fragilidade, na prática tem fornecido soluções boas. Por este motivo, propõe-se aqui dispor dos dois métodos de adaptação, e em função do desempenho computacional e das qualidades das soluções obtidas, define-se o melhor método para o problema em questão.

Outras facilidades importantes são o agrupamento de elementos, e a detecção de estrutura da solução ótima, reformulação e re-otimização do *DAOP*. Estas facilidades são graus de liberdade adicionais para refinar a solução encontrada.

Aqui não se propõe métodos novos, mas sim a utilização das melhores facilidades de soluções proposta anteriormente. Isto fornece um sistema mais robusto e eficiente, pois agregam os valores de cada iniciativa individualmente. Além disso, fornece opções para os temas em que ainda não há consenso, sendo ainda dependente do problema e a escolha o método passa a ser em função do desempenho e robustez no problema em questão.

Depois de resolver o *DAOP*, propõe-se a utilização da análise de resultados e implementação de resultados. Na análise dos resultados do otimizador, propõe-se a verificação das condições de otimalidade da solução e análise discriminante dos perfis de controle. É verificado se a solução é ótima (Hamiltoniano é constante) e se é diferente da ultima solução implementada (efetuado de forma semelhante ao *RTO*, adaptado para *DAOP*). Com esta análise é possível decidir quando a receita otimizador deve ser implementada na operação da planta.

Uma vez que a solução passou na avaliação de resultados, devem-se implementar as ações de controle. Neste momento, é criada uma agenda de ações em batelada e executada a implementação conforme o procedimento (tempo, ação). Os valores das posições enviados para o sistema de controle da planta devem ser compensados pelos *biases* definidos na reconciliação de dados, isto para que a ação de controle real seja a esperada pelo otimizador.

Esta análise e implementação de resultados representam uma vantagem interessante, pois evita mudanças desnecessárias de receitas ótimas. Isto melhora os resultados do otimizador e a satisfação dos usuários, pois perturba menos a planta e proporciona um retorno financeiro mais consistente do otimizador.

Uma análise complementar à análise de resultados é a realização da análise de sensibilidade da solução (restrições e função objetivo) com as ações de controle. Esta análise auxilia na avaliação das decisões do otimizador. Esta análise permite verificar se a solução é muito sensível aos perfis de controle. Este fato indica baixa robustez da solução, ou seja, incertezas no modelo e entradas fornecem soluções muito diferentes. Neste caso, deve-se tomar providencias para melhorar esta robustez. Esta facilidade é uma ferramenta muito boa para auxiliar este processo de decisão, juntamente com a ferramenta de monitoração e diagnóstico do *DRTO*.

Além do processo de solução do *DAOP*, a monitoração e diagnóstico do *DRTO* são fatores críticos de sucesso deste tipo de aplicação. A capacidade de avaliação e diagnóstico contribui para a solução de problemas com a aplicação e conferem maior robustez e aceitação deste tipo de otimizador. Como parte deste processo, o otimizador fornece as informações necessárias às ferramentas de monitoração e diagnóstico a cada rodada de cada uma das aplicações do sistema de *DRTO*. Estas aplicações geram informações históricas para a monitoração dos eventos do otimizador, monitoração dos perfis dos

estados e controles, monitoração dos indicadores de robustez, desempenho e qualidade da solução e monitoração dos indicadores técnicos e gerenciais do problema de otimização

Considerando as atividades executadas de forma *offline*, há duas atividades projetadas para este sistema que agregam muito em qualidade ao sistema de *DRTO*. São elas: facilidades de estudos de casos de otimização dinâmica e simulação e visualização da solução do otimizador *online*. Nos estudos de casos, pode-se repetir o mesmo caso resolvido pelo otimizador *online*, realizar análises de sensibilidade da solução do problema de otimização (movendo restrições ou alterando incertezas do modelo), estudar casos de mudanças de estrutura do *DAOP* (ex.: excluir ou incluir restrições), e estudar os efeitos das alterações dos parâmetros de sintonia do otimizador. Além disso, é possível executar iteração a iteração, depurar a solução e visualizar os resultados parciais. Tanto na solução do *DAOP* como na simulação de uma solução ótima, pode-se visualizar a solução do otimizador *online*, verificar violações indevidas ou não das restrições, e as condições de otimalidade

Estas facilidades da aplicação *offline* representam um diferencial em relação às aplicações conhecidas no mercado e literatura aberta. Principalmente a capacidade de depurar e realizar os estudos de casos previstos neste capitulo, representam uma solução até então inexistente para este tipo de aplicação.

Relativo à monitoração e diagnóstico de *DRTO*, são propostas aqui ferramentas de monitoração e diagnóstico de *DRTO*, assim como o processo de solução de problemas, baseado nos preceitos da metodologia *DMAIC*. São apresentados os conceitos de monitoração e diagnóstico de *DRTO*, as estruturas das ferramentas de monitoração (com todos os seus aspectos), de diagnóstico (nos seus 3 níveis de profundidade). A ferramenta propõe a monitoração de aspectos gerenciais e técnicos para cada módulo do otimizador (robustez, desempenho, qualidade da solução, fidelidade do modelo, dentre outros). A ferramenta de diagnóstico visualiza as informações do problema de otimização, de sua solução, dos algoritmos utilizados e de desempenho e sintonia do sistema. O diagnóstico pode ser não invasivo (com as informações geradas pelo otimizador *online*) ou invasivo (refazendo a rodada de forma *offline* ou através de testes específicos).

No processo de diagnóstico, pode-se destacar também a avaliação dos erros de discretização. Esta facilidade é simples estabelecida conceitualmente, mas não utilizado com o propósito de diagnóstico. Esta análise é fundamental na validação de uma solução do *DRTO*.

Além das ferramentas de monitoração e diagnóstico, também é proposta uma metodologia de diagnóstico e sintonia de *DRTO*, onde é apresentada uma metodologia de análise de falhas do *DRTO* e de análise de processo do otimizador, onde são avaliadas a robustez e desempenho do sistema e realizada a busca de oportunidades de otimização.

Parte 2 – Solução de problema de inviabilidade

5 Introdução

O sistema *DRTO* geralmente é executado automaticamente. O operador só deverá conduzir o otimizador para resolver o problema de otimização de forma correta. No entanto, os *softwares* de otimização estão sujeitos a falhas. Ao tentar resolver um problema de otimização num sistema de *DRTO* é possível encontrar falhas na obtenção da solução ótima, que podem ocorrer quando o problema for inviável. Não é raro encontrar situações em que as condições iniciais são inviáveis ou inexoravelmente tendem a inviabilidade. Nesses casos, não há qualquer possibilidade do otimizador encontrar uma solução. A academia tem gasto muito esforço para resolver os problemas de inviabilidade de diferentes maneiras.

Usualmente duas ações podem ser tomadas: eliminar as restrições que foram violadas ou relaxar estas restrições para posições onde o problema se torna viável. Diferentemente da otimização estacionária, a viabilidade da solução depende das suas condições iniciais e das direções das derivadas no tempo das variáveis de estado. Uma determinada variável de estado pode estar numa posição viável, mas a sua derivada pode apontar para uma inviabilidade, levando o processo a violar as restrições antes das correções possíveis das ações de controle, como será mostrado no Capítulo 7 (ver Figura 7.14d, do caso 2, otimização dinâmica de um reator semi-batelada não-isotérmica). Esta inviabilidade ocorre quando o processo está próximo do seu limite. Além disso, os valores das variáveis algébricas calculadas pela inicialização do otimizador podem estar indevidamente fora dos limites de operação ou até mesmo tender inexoravelmente à inviabilidade. Há também situações onde a receita do processo é alterada entre um estágio e outro, ou ocorre uma perturbação importante, que obriga a operação a alterar abruptamente a receita do processo. Neste momento, o problema pode se tornar inviável (intencionalmente ou não). Portanto, da mesma forma que seria tratada a inicialização do problema em condições inviáveis, no momento do horizonte de otimização onde esta descontinuidade ocorrer (ex.: transição de especificação de produção) um novo estágio é criado com o objetivo de regular o processo, trazendo as variáveis para dentro das restrições no mínimo tempo. Estas situações podem ser tais que a solução do otimizador seja inviável, como mostra a Figura 5.1.

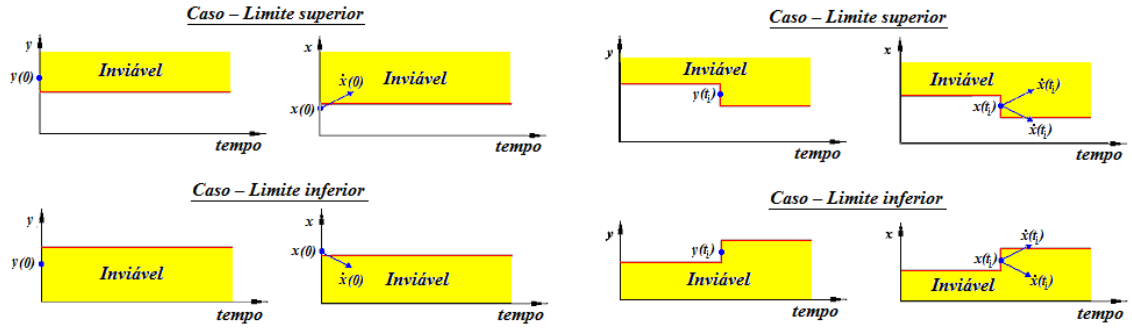


Figura 5.1 – Casos de inviabilidade de variáveis de estado.

(a) Tendência e inviabilidade inicial de variáveis de estado e (b) tendência e inviabilidade intermediária de variáveis de estado.

Outro problema comum é a presença de conflitos de especificações. Não é raro encontrar situações no uso de otimização em tempo real onde os operadores estabelecem posições das restrições que passam a competir entre si. Isto comumente leva a uma solução inviável do problema de otimização dinâmica. E as mesmas medidas acima citadas usualmente são adotadas e com as mesmas conseqüências. A tendência à inviabilidade inicial e intermediária, e os conflitos de especificações podem resultar em um fator de insucesso da utilização de um otimizador dinâmico.

Para resolver eficientemente os problemas de inviabilidade em problemas de otimização dinâmica, propõe-se neste trabalho uma metodologia baseada na relaxação de restrições de um *DAOP* através da solução de um problema de otimização dinâmica multiobjetivos (*MODAOP*).

6 Revisão bibliográfica

Os pacotes de otimização dinâmica utilizam normalmente uma formulação padronizada onde as restrições são impostas apenas às variáveis de estado, de controle e parâmetros do modelo de otimização dinâmica. Neste caso, restrições de desigualdade são incorporadas como equações algébricas pela introdução de variáveis novas limitadas de acordo com a restrição original. Desta forma, os problemas de otimização dinâmica são usualmente apresentados na forma padrão abaixo:

$$\begin{aligned} &\min_{u(t),p} \quad \phi\left(x\left(t_f\right)\right) \\ &s.t. \\ &\quad F\left(\dot{x}(t), x(t), y(t), u(t), p, t\right) = 0; \quad x\left(t_0\right) = x_0 \quad t \in \left[t_0, t_f\right] \\ &\quad x^L \le x(t) \le x^U \\ &\quad y^L \le y(t) \le y^U \\ &\quad u^L \le u(t) \le u^U \\ &\quad p^L \le p \le p^U \end{aligned} \tag{6.1}$$

onde $\phi(x(t_f))$ é o objetivo, $F(\bullet) \in \Re^{nx+ny}$ é o sistema *DAE*, $x(t) \in \Re^{nx}$ são as variáveis de estado, $y(t) \in \Re^{ny}$ as variáveis algébricas, $u(t) \in \Re^{nu}$ as variáveis de controle, e $p \in \Re^{np}$ os parâmetros do modelo independentes do tempo. As restrições do processo podem ser de caminho, instantes de tempo no intervalo ou de tempo final (t_f).

Nos métodos diretos o *DAOP* é transformado em um problema de *NLP* após a parametrização das variáveis. Em todos os casos, os *NLP's* resultantes são normalmente apresentados na forma:

$$\begin{aligned} &\min_{z} \quad f(z) \\ &s.t. \\ &\quad c(z) = 0 \\ &\quad z^L \le z \le z^U \end{aligned} \tag{6.2}$$

onde $f(z)$ é a função objetivo, $c(z)$ as restrições do sistema, e z são as variáveis do *DAOP* discretizado.

A identificação da inviabilidade deve ser percebida o mais rápido possível pelo solver de *NLP*. Iremos considerar aqui os comportamentos dos algoritmos *IPOPT* e *SNOPT*, pois têm sido usados neste trabalho e representam bem os algoritmos típicos de pontos interiores (*IP's*) e *SQP's*. No *IPOPT*, que é um algoritmo de ponto interior do tipo *infeasible path*, a detecção da inviabilidade é feita através do uso do método de filtro *linesearch* que procura resolver um problema de otimização bi-objetivo, tendo com alvos a minimizar a função objetivo do problema de barreira $\varphi(z,\mu) = f(z) - \mu \Sigma \ln(z_i)$ e, onde μ é o parâmetro de barreira, e da violação da restrição $\theta(z) = \|c(z)\|$. O balanceamento entre estes

dois objetivos é feito através da utilização da função de mérito $M(z,\mu,\eta) = \varphi(z,\mu) + \eta\theta(z)$ (Wächter e Biegler, 2003; Nocedal e Wright, 2006). O ponto tentativa z é aceito se $\varphi(z,\mu)$ ou $\theta(z)$ é melhorado. Se não houver suficiente progresso em z, o algoritmo entra na fase de restauração da viabilidade, abandonando temporariamente a minimização da função objetivo $\varphi(z,\mu)$ e minimiza a violação da restrição. Caso o *IPOPT* não consiga obter uma solução viável, o algoritmo retorna à última posição obtida das variáveis de estado e de controle nos pontos discretos do problema de *NLP* original, e calcula um novo ponto tentativa.

No *SNOPT*, quando a inviabilidade é identificada, o algoritmo de otimização força a busca de um ponto viável. Se não for possível encontrá-lo, o algoritmo entra no modo elástico, onde adiciona variáveis de folga e resolve o problema de viabilidade, que é equivalente à minimização da violação das restrições dentro da função objetivo:

$$\begin{aligned} &\min_{x,v,w} \; f(z) + \rho e^T (v + w) \\ &\text{s.a.} \\ &\qquad l \le \begin{pmatrix} z \\ c(z) - v + w \end{pmatrix} \le u \\ &\qquad v \ge 0; \quad w \ge 0 \end{aligned} \tag{6.3}$$

onde e é um vetor unitário, ρ peso elástico, v e w variáveis elásticas.

Desde a década de 70 há preocupações em identificar o conjunto de restrições que tornam o problema de otimização inviável. Na época, o foco era identificar tal conjunto em problemas de otimização linear (*LP's*). A forma de identificar o conjunto de restrições inviáveis era utilizando a localização direta usando heurísticas, tais como: limite inferior maior do que o superior, temperatura de topo maior do que a de fundo de uma coluna de destilação.

Na década de 80 começaram a surgir métodos mais sistemáticos para identificação do conjunto de restrições inviáveis e localização da causa de tal inviabilidade, também com foco em *LP's* (Greenberg, 1993). Nesta abordagem, identifica-se um conjunto irredutível de inviabilidade (*IIS*) (Van Loon, 1981), onde qualquer subconjunto escolhido torna o problema viável. Em um problema de otimização pode haver mais de um *IIS* e esta detecção é combinatória, elevando o custo computacional para encontrar os *IIS*. O fato de existir mais de um *IIS* em um problema de otimização nos remete a uma questão que o método não resolve. Qual a melhor escolha do *IIS* para resolver o problema de otimização? Há um conjunto *IIS* onde a função objetivo tem o melhor valor, porem é necessário testar os *IIS* para encontrar o melhor. Os métodos de identificação de *IIS* são: a utilização de filtro de deleção, de adição, elástico e de sensibilidade (Chinneck, 2008). Além disso, podem-se combinar os filtros e agrupar restrições para acelerar o seu processo de identificação. Esta abordagem tem sido usada no *MINOS* (Chinneck, 1994) e *CONOPT* (Drud, 1994).

Mais recentemente surgiram os métodos de consenso das restrições para identificar inviabilidade de *NLP's*. Onde se calculam as distâncias viáveis, gerando os vetores de

viabilidade (Chinneck, 2004). As direções ruins destes vetores não devem entrar num vetor chamado de vetor de consenso, que representa a viabilidade média para as restrições violadas. Esta abordagem é limitada para a identificação de inviabilidade de problemas de otimização dinâmica devido aos seguintes motivos: o número de combinações possíveis de *IIS's* é relativamente grande devido ao grande número de restrições no domínio do tempo discreto; uma restrição no domínio do tempo contínuo gera um número de restrições equivalente ao número de pontos de discretização do sistema. No domínio do tempo contínuo, uma restrição na variável de estado diferencial em um instante de tempo está correlacionada com os valores nos outros instantes de tempo. Isto faz com que uma violação da restrição seja sensível às ações de controle nos instantes anteriores ao tempo em referência. A sensibilidade maior está ligada às ações de controle tomadas no início do horizonte de otimização. Este comportamento é semelhante ao de uma operação de convolução, onde uma ação de controle no início do horizonte de otimização tem uma forte influência nas variáveis de estado diferenciais num instante bem adiante, isto faz com que seja mais eficaz alterar as ações de controle num tempo distante para corrigir uma inviabilidade. Este tipo de problema torna difícil o uso eficiente das técnicas descritas acima, pois encontram conjuntos de restrições inviáveis, pois o número de combinações possíveis de *IIS* é muito grande.

Tamiz et al. (1996) propuseram o uso da programação de metas (*goal programming*) para a solução do problema de inviabilidade, formulado na forma de um problema de otimização *NLP* multiobjetivos (Equação 6.4a). Nesta formulação, introduzem-se variáveis elásticas (s_i) que relaxam o problema de otimização da seguinte forma:

$$\begin{aligned} &\min_{z,s} \ \{s_1, s_2, \ldots, s_m\} \\ &s.t. \\ &g_i(z) \le s_i \quad i = 1, \cdots, m \\ &s_i \ge 0 \end{aligned} \tag{6.4a}$$

$$\begin{aligned} &\min_{z,s,\gamma} \quad \gamma \\ &s.t. \\ &w_i s_i \le \gamma \\ &g_i(z) - s_i \le \gamma \quad i = 1, \cdots, m \\ &s_i \ge 0 \end{aligned} \tag{6.4b}$$

onde z são as variáveis originais do problema, γ é a função objetivo, $\sum_{i=1}^{m} w_i = 1$ e $w_i \ge 0$.

Neste caso, não se obtém um conjunto ótimo de Pareto, mas uma solução única onde os pesos w_i são escolhidos segundo um critério heurístico (Yang, 2008). Esta abordagem identifica a inviabilidade através da solução do problema de otimização de *NLP*, tornando mais eficaz a solução do problema de inviabilidade.

Os métodos até então conhecidos procuram localizar e eliminar inviabilidades de problemas de *LP* e *NLP*. Estes métodos não têm sido aplicados para identificar e resolver

inviabilidades em *DAOP's*. As técnicas que classificam e eliminam restrições em problemas de *LP* e *NLP* não são efetivos para resolver casos com inviabilidades ou tendências à inviabilidade de condições iniciais ou intermediárias, pois são problemas *NP* (*Non-Deterministic Polynomial time*), de custo fatorial ou combinatório em função de sua complexidade do problema de otimização dinâmica. Além disso, o uso inerente de heurísticas na obtenção da solução pode conduzir a planta para condições não muito lucrativas ou até mesmo levar a planta para condições menos seguras. Diferentemente da otimização estacionária, uma solução viável de sistemas dinâmicos dependem de suas condições iniciais e das direções de suas derivadas no tempo. Usualmente a planta opera próximo dos limites. Isto acontece após algumas rodadas do otimizador em tempo real. Para resolver eficientemente os problemas de inviabilidade descritos acima, é proposta uma metodologia baseada na relaxação de restrições de um *DAOP* através da solução de um problema de otimização dinâmica multiobjetivos, que é descrito na seção a seguir.

Existem algumas técnicas usadas para suavizar as restrições em *MPC*. A primeira abordagem adotada foi a conversão da restrição desigualdade original ($g(z) \leq 0$) em restrição de igualdade, adicionando variáveis de folga s ($g(z) + s^2 = 0$) e uma função de custo ($s^T Q s$) (Jacobson e Lele, 1969; Camacho e Bordon, 1999). Outra técnica usada para lidar com restrições de caminho de desigualdade em problemas de controle ótimo baseia-se na medição de violação de restrição, adicionando uma nova variável diferencial como $\dot{w}(t) = \left[\max\left(0, g\left(z(t)\right)\right)\right]^2$ e uma restrição de ponto final $w(t_f) \leq \varepsilon$ onde ε é um valor positivo muito pequeno (Vassiliadis et al., 1994). Em ambos os casos as restrições não são movidas, tal como é proposto neste trabalho.

7 Proposta de solução de inviabilidade

O sistema *DRTO* deve ser capaz de otimizar dinamicamente o processo, que significa a capacidade de obter automaticamente uma solução ótima do *DAOP*. Os principais requisitos deste tipo de aplicações são sua eficiência e robustez para encontrar uma solução ótima interessante. Uma das questões mais importantes encontradas durante a solução *DAOP* é a inviabilidade do problema. Como vimos antes, geralmente existem duas opções para resolver essas questões de inviabilidades: classificação e eliminação ou identificação e relaxamento das restrições violadas.

As técnicas de resolução de problemas de inviabilidades em *NLP* existentes na literatura científica não são soluções práticas aplicáveis a um *DAOP's* para uso em tempo real. Neste caso, estas técnicas podem ser usadas manualmente e, em parte, usando somente poucas combinações possíveis, onde é feita a escolha do sistema de subconjuntos inviáveis, utilizando a experiência do usuário.

Ao resolver um problema de otimização dinâmica, o usuário pode decidir pela busca da solução do problema original ou do problema relaxado. Na solução do problema original, caso o problema seja inviável, deve-se verificar se a falha é devido às condições iniciais inviáveis ou se os valores das derivadas das variáveis de estado tendem a violar as restrições. Além disso, é necessário analisar a viabilidade ao longo de todas as trajetórias das variáveis de estado e de controle, pois pode haver uma inviabilidade intermediária ou de instante final.

Com base nestes conceitos, podemos imaginar um processo de busca manual para solução viável (Figura 7.1), que adota as técnicas já mencionadas na revisão bibliográfica. Neste procedimento, o *DAOP* original é então resolvido e se não for possível, devemos adotar uma estratégia de eliminação ou relaxamento de restrição em um subconjunto de restrições inviáveis. Como as informações do *NLP* têm sido usadas para resolver inviabilidades, uma determinada restrição pode ser violada em um intervalo de tempo, mas não em outro. O processo para obter a melhor solução viável passa a ser um processo iterativo, através de sucessivas modificações ou flexibilizações do *DAOP* original, que resulta num maior esforço computacional.

Observe que é muito difícil estabelecer um procedimento automático para resolver as inviabilidades em *DAOP*. Para lidar com este problema, propomos aqui um procedimento que pode ser usado como ferramenta de diagnóstico ou como sistema de solução robusta de *DAOP* em tempo real.

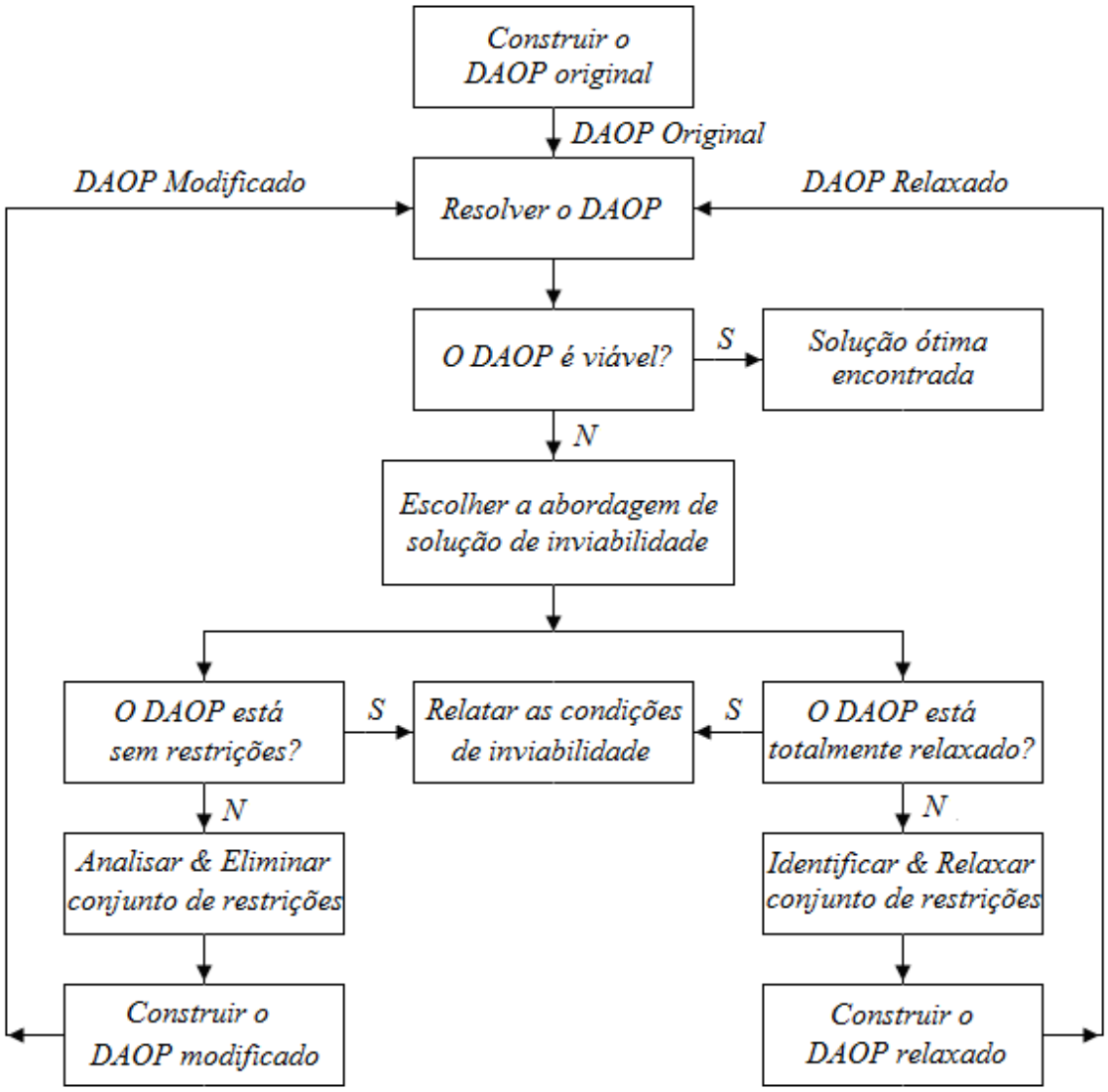


Figura 7.1 – Procedimento manual para resolver inviabilidades em *DAOP*.

Para resolver este problema, propõe-se a seguinte metodologia que pode ser utilizada para obter uma solução viável com relaxamento de restrições ou para executar um diagnóstico de forma a identificar as restrições problemáticas do *DAOP* original. A solução automática de inviabilidades em *DAOP* é mais vantajosa quando a estrutura do problema se torna mais complexa e o processo está freqüentemente sujeito a alterações na receita de operação ou quando há transições constantes no processo, que podem resultar em um problema inviável. Neste caso, o relaxamento automático das restrições se torna um fator crítico para resolver problemas de otimização dinâmica de forma robusta. O relaxamento automático de restrições resulta num procedimento mais simples e numa técnica menos iterativa que o procedimento manual, conforme mostrado na Figura 7.2. Neste caso, o usuário cria o problema original de *DAOP* e define as regras para o relaxamento de restrições (por exemplo: conjunto de restrições que podem ser relaxadas, máxima variância permitida para um relaxamento desejado e a importância relativa de cada restrição). Uma vez que o problema original é construído, o usuário pode inicialmente escolher resolver o *DAOP* original, ou resolver o *DAOP* total ou parcialmente relaxado.

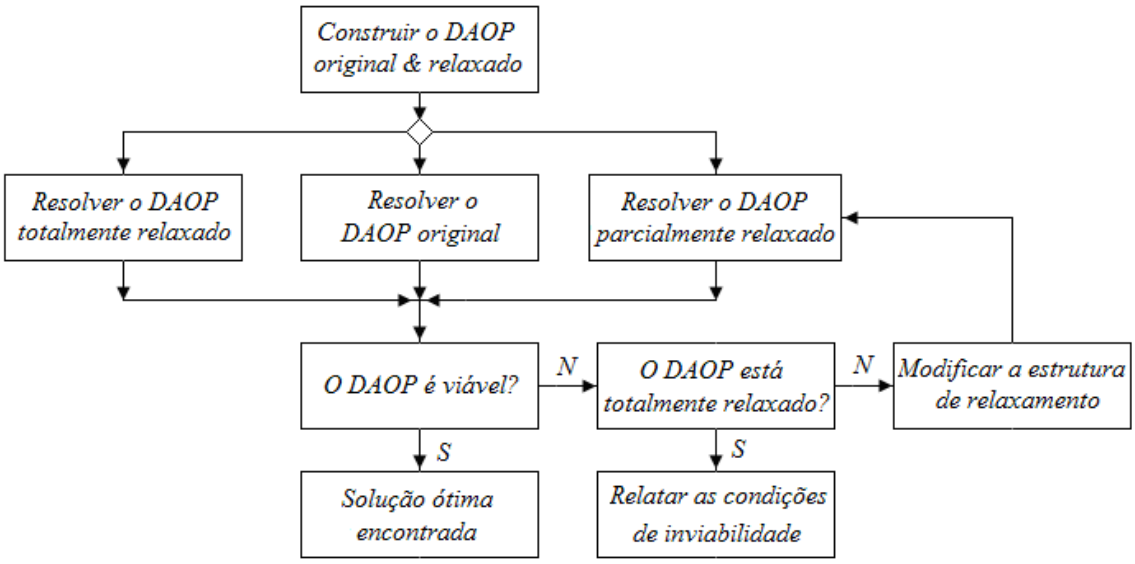


Figura 7.2 – Procedimento automático para resolver inviabilidades em *DAOP*.

Uma vez executado o algoritmo de solução de *DAOP* e encontrado um problema inviável, é então obtido um mapa preciso da flexibilização necessária para o problema original. O usuário pode decidir pela a execução automática do problema relaxado ou até mesmo pela modificação do problema de otimização com base em informações precisas dadas pela solução relaxada. Se o problema totalmente relaxado for inviável, há fortes evidências sobre os problemas estruturais no *DAOP*.

Esta metodologia pode ser usada como ferramenta de diagnóstico, evidenciando quando a solução é inviável, localiza as causas básicas de tal inviabilidade e pode propor uma solução para esta problemática. Para isso, é necessário também encontrar uma nova estrutura de problema viável e sua política ótima para todas as trajetórias das variáveis de estado e de controle.

Os detalhes do método proposto são mostrados abaixo, onde o otimizador identifica inviabilidades no *DAOP* original conforme as situações descritas anteriormente. Sendo que, esse método pode ser usado como uma ferramenta de diagnóstico e como um aplicativo de otimização dinâmica em tempo real (*DRTO*), proporcionando mais robustez para este tipo de aplicação e resultando numa maior aceitação pelos usuários do sistema.

7.1 Métodos de solução do problema multiobjetivos

Há uma série de abordagens para resolver problemas de otimização multiobjetivos. Dentre estes, podemos destacar os algoritmos evolucionários (Coello *et al.*, 2005) tais como *VEGA* (Schaffer, 1984), *NSGA* (Srinivas e Deb, 1994), *NPGA* (Horn *et al.*, 1994), *SPEA* (Zitzler e Thiele, 1998/1999) e *NSGA-II* (Deb *et al.*, 2000/2002). Estes métodos são baseados em algoritmos genéticos (Konak *et al.*, 2006) onde se encontra um conjunto ótimo de Pareto (Marler e Arora, 2004). Neste caso, não temos uma solução, mas sim um conjunto de soluções ótimas concorrentes. Numa segunda abordagem, têm-se os algoritmos baseados na soma ponderada dos objetivos conflitantes (Marler e Arora, 2004). Neste caso, os objetivos concorrentes são transformados em um único objetivo que

pondera cada objetivo individual. O mecanismo de busca pode ser associado a um algoritmo genético ou até mesmo a um determinístico. Uma terceira classe de métodos são os algoritmos de funções objetivo limitadas (Marler e Arora, 2004). Este método estabelece um dos objetivos concorrentes como objetivo principal que será colocado como uma função objetivo simples do novo problema, e o restante dos objetivos concorrentes passa a ser restrições limitadas pelos dois lados por um fator ε, que é modificado a cada iteração para cada objetivo secundário. Com isso, constrói-se uma frente ótima de Pareto onde se tem um conjunto de soluções concorrentes do problema. Uma quarta abordagem é a utilização de alvo dado por um valor utópico (Nemhauser,1988). O valor utópico é o ponto formado pelo conjunto de valores das funções objetivo obtidos pela otimização de cada objetivo individualmente. Este é um valor inatingível, pois não levam em conta os outros objetivos, mas serve como ponto de referência para minimizar as distâncias dos objetivos individuais do valor utópico.

Para a solução de problemas de otimização em tempo real, é necessário obter uma solução numérica única a ser aplicada na planta em operação. Este fato faz com que os métodos evolucionários não sejam recomendados, pois gera um conjunto de soluções (conjunto ótimo de Pareto) e não uma solução proposta. Para resolver este problema, poder-se-ia adotar o método de soma ponderada dos objetivos, porem este método se limita à solução de superfície convexa dos objetivos. Uma abordagem possível seria a utilização de algoritmos de funções objetivo limitadas, porém é necessário eleger o objetivo mais importante. Em otimização dinâmica é difícil dizer qual objetivo é mais importante, pois sua importância depende do problema e das magnitudes das violações das restrições. A utilização do algoritmo de otimização multiobjetivos baseado em uma utopia (Logist *et al.*, 2009) se mostra mais interessante na solução de problemas de otimização dinâmica com solução simultânea da viabilidade das restrições, pois não apresenta os problemas acima citados.

7.2 Proposta de solução de inviabilidades em problemas de otimização dinâmica

O método proposto consiste na solução de um *DAOP*, onde o problema é reformulado como otimização multiobjetivos com o relaxamento das restrições. Este relaxamento é feito através da inclusão de variáveis de folga e dos alvos utópicos no *DAOP*. A solução deste problema leva à obtenção da mínima relaxação das restrições problemáticas. Como características básicas desta abordagem temos ao mesmo tempo *soft-constraints* e *hard-constraints*, que leva à solução do problema otimização e flexibilização das restrições ao mesmo tempo. Isto torna mais fácil para decidir sobre a flexibilização de restrições. Uma das principais características desse método é o fato do usuário não precisar escolher quais restrições devem ser flexibilizadas, a menos que seja problema claramente definido. Na verdade, o algoritmo de otimização toma a decisão de quais restrições devem ser relaxadas. Na solução do problema, obtém-se uma indicação qualitativa e quantitativa de quais restrições e quanto as mesmas devem ser flexibilizadas.

A idéia básica é reformular o *DAOP* original como um problema multiobjetivos para resolver as inviabilidades, onde são minimizados o objetivo original e qualquer movimento das restrições problemáticas. Este mínimo movimento é definido como a integral das relaxações, como mostrado na Figura 7.3.

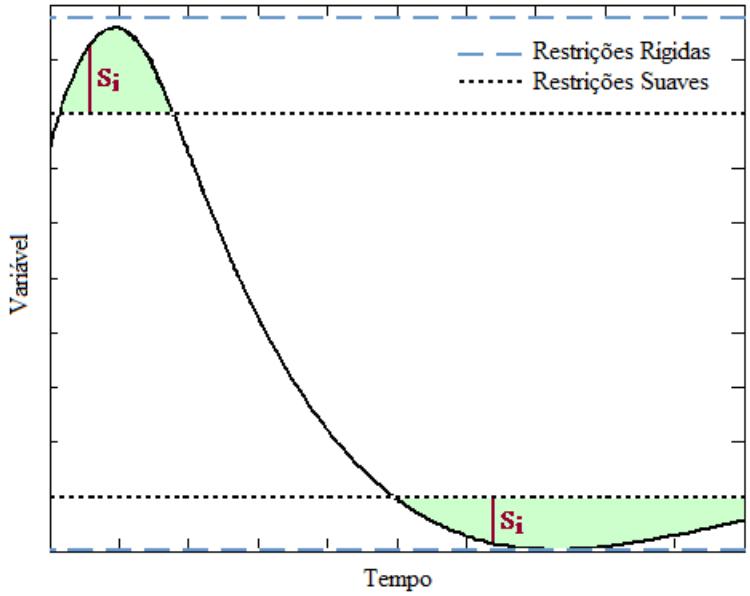


Figura 7.3 - Minimização de relaxamento de restrições usando integral de variáveis de folga das restrições do problema (S_i).

A formulação matemática da função objetivo como problema de otimização multiobjetivos com relaxamento das restrições é descrita na Equação 7.1a.

$$\min_{u,s^x,s^y,s^u,p}\left[\phi\left(x\left(t_f\right)\right) \quad \int_0^{t_f}\left(s^x{}_i\left(t\right)\right)^2 dt \quad \int_0^{t_f}\left(s^y{}_j\left(t\right)\right)^2 dt \quad \int_0^{t_f}\left(s^u{}_k\left(t\right)\right)^2 dt\right] \tag{7.1a}$$

onde $i \in \{1,\ldots,nx\}, j \in \{1,\ldots,ny\}$ e $k \in \{1,\ldots,nu\}$.

As flexibilizações das restrições são feitas através da inclusão de variáveis de folga como a seguir:

$$\begin{aligned} x^L \le x(t) + s^x(t) \le x^U \qquad & s_L^x \le s^x(t) \le s_U^x \\ y^L \le y(t) + s^y(t) \le y^U \ \text{ e } \qquad & s_L^y \le s^y(t) \le s_U^y \\ u^L \le u(t) + s^u(t) \le u^U \qquad & s_L^u \le s^u(t) \le s_U^u \end{aligned} \tag{7.1b}$$

A equação a seguir descreve a formulação matemática do problema de otimização dinâmica multiobjetivos com relaxamento das restrições (através da inclusão de variáveis de folga no *DAOP*):

$$\min_{u,s^x,s^y,s^u,p}\left[\phi\left(x\left(t_f\right)\right) \quad \int_0^{t_f}\left(s^x{}_i\left(t\right)\right)^2 dt \quad \int_0^{t_f}\left(s^y{}_j\left(t\right)\right)^2 dt \quad \int_0^{t_f}\left(s^u{}_k\left(t\right)\right)^2 dt\right]$$

s.a. (7.2)

$$F\left(\dot{x}(t), x(t), y(t), u(t), p, t\right) = 0; \qquad x(t_0) = x_0 \quad t \in \left[t_0, t_f\right]$$

$$
\begin{aligned}
& x^L \le x(t) + s^x(t) \le x^U \\
& y^L \le y(t) + s^y(t) \le y^U \\
& u^L \le u(t) + s^u(t) \le u^U \\
& p^L \le p \le p^U \\
& s_L^x \le s^x(t) \le s_U^x \qquad onde \quad s^x(t) \in \Re^{S_x} \\
& s_L^y \le s^y(t) \le s_U^y \qquad onde \quad s^y(t) \in \Re^{S_y} \\
& s_L^u \le s^u(t) \le s_U^u \qquad onde \quad s^u(t) \in \Re^{S_u}
\end{aligned}
$$

onde $i \in \{1,\ldots,nx\}$, $j \in \{1,\ldots,ny\}$ e $k \in \{1,\ldots,nu\}$.

Diferenças básicas para o *DAOP* original é o uso de restrições suaves e rígidas ao mesmo tempo e a solução simultânea do problema de relaxamento das restrições e a otimização do problema original. Para cada restrição relaxada, é acrescentada uma variável de decisão (s_i) e uma restrição para a mesma, no problema original multiobjetivos. Pode-se notar mais adiante (Equação 7.6) que são necessárias duas restrições para cada relaxamento, no DAOP resultante que será resolvido pelo algoritmo de otimização dinâmica. Isto se deve ao fato de acrescentar uma restrição para a variável de folga e outra para controlar o relaxamento da restrição original. O tempo computacional gasto pode variar de levemente superior ao gasto para resolver o problema viável original, ou até mesmo dobrar o mesmo. Isto dependerá da estrutura do problema relaxado.

A solução do *DAOP* multiobjetivos escrito como apresentado acima é obtida como um conjunto ótimo de Pareto. Para as aplicações de *DRTO*, é desejável que se obtenha um único valor ótimo da função objetivo em vez de um conjunto ótimo de soluções. Para obter uma solução única, o problema original pode ser reformulado na forma de problema de programação por metas (*goal programming*) e com relaxamento das restrições na seguinte forma:

$$
\begin{aligned}
& \min_{u,\Delta\gamma,s^x,s^y,s^u,p} \gamma(t_f) \\
& s.a. \\
& F_i\left(\dot{x}(t), x(t), y(t), u(t), p, t\right) = 0 \quad t \in \left[t_0, t_f\right] \\
& \frac{d\gamma}{dt} = \Delta\gamma(t), \quad \gamma(t_0) = 0 \\
& \phi\left(x(t_f)\right) - \phi^{ref} \le \gamma w_0 \\
& \left(s^x{}_i(t)\right)^2 - \left(s^x{}_i^{ref}\right)^2 \le \Delta\gamma(t)\, w^x{}_i \quad onde \;\; i = 1,\cdots,nx \\
& \left(s^y{}_i(t)\right)^2 - \left(s^y{}_i^{ref}\right)^2 \le \Delta\gamma(t)\, w^y{}_i \quad onde \;\; i = 1,\cdots,ny \\
& \left(s^u{}_i(t)\right)^2 - \left(s^u{}_i^{ref}\right)^2 \le \Delta\gamma(t)\, w^u{}_i \quad onde \;\; i = 1,\cdots,nu \\
& x^L \le x(t) + s^x(t) \le x^U \quad onde \;\; s^x(t) \in \Re^{nx} \\
& y^L \le y(t) + s^y(t) \le y^U \quad onde \;\; s^y(t) \in \Re^{ny} \\
& u^L \le u(t) + s^u(t) \le u^U \quad onde \;\; s^u(t) \in \Re^{nu}
\end{aligned}
\qquad (7.3)
$$

Onde foi criada uma nova variável diferencial γ que é a nova função objetivo a ser minimizada. A variável de controle associada diretamente a este objetivo é Δγ. O significado dessa nova variável γ é a integral relacionada à relaxação mínima de cada restrição ponderada por sua variância aceitável e, ao mesmo tempo, a minimização da distância da função objetivo original de seu valor utópico. Além disso, todas as relaxações são restritas e as restrições originais são relaxadas no *DAOP*.

Um bom valor para $s^{x}{}_{i}^{ref} = 0$ seria o relaxamento desejado, ou seja, é desejável uma solução do problema sem relaxamento das restrições. O mesmo é feito para $s^{y}{}_{i}^{ref}$ e $s^{u}{}_{i}^{ref}$. Desta forma, podemos reescrever o problema na forma:

$$
\begin{aligned}
&\min_{u,\Delta\gamma,s^x,s^y,s^u,p} \gamma\left(t_f\right) \\
&s.a. \\
&F_i\left(\dot{x}(t), x(t), y(t), u(t), p, t\right) = 0 \quad t \in \left[t_0, t_f\right] \\
&\frac{d\gamma}{dt} = \Delta\gamma(t), \quad \gamma(t_0) = 0 \\
&\phi\left(x\left(t_f\right)\right) - \phi^{ref} \le \gamma w_0 \\
&\left(s^{x}{}_{i}(t)\right)^2 \le \Delta\gamma(t)\, w^{x}{}_{i} \quad onde \;\; i = 1, \cdots, nx \\
&\left(s^{y}{}_{i}(t)\right)^2 \le \Delta\gamma(t)\, w^{y}{}_{i} \quad onde \;\; i = 1, \cdots, ny \\
&\left(s^{u}{}_{i}(t)\right)^2 \le \Delta\gamma(t)\, w^{u}{}_{i} \quad onde \;\; i = 1, \cdots, nu \\
&x^L \le x(t) + s^x(t) \le x^U \quad onde \;\; s^x(t) \in \Re^{nx} \\
&y^L \le y(t) + s^y(t) \le y^U \quad onde \;\; s^y(t) \in \Re^{ny} \\
&u^L \le u(t) + s^u(t) \le u^U \quad onde \;\; s^u(t) \in \Re^{nu}
\end{aligned}
\tag{7.4}
$$

O peso $w^{x}{}_{i}$ pode passar para o lado esquerdo da desigualdade, resultando em $\left(s^{x}{}_{i}(t)\right)^2 \Big/ w^{x}{}_{i} \le \Delta\gamma$. Esta forma sugere um fator de normalização. Este fator de normalização pode vir da análise de variância e utilizar o desvio padrão aceitável da violação da restrição ($\sigma^{x}{}_{i}$). Aplicando este conceito a este tipo de restrição, obtém-se a seguinte formulação do problema de otimização:

$$\min_{u,\Delta\gamma,s^x,s^y,s^u,p} \gamma(t_f)$$

s.t.

$$F(\dot{x}(t), x(t), y(t), u(t), p, t) = 0; \qquad x(t_0) = x_0 \quad t \in [t_0, t_f]$$

$$\frac{d\gamma}{dt} = \Delta\gamma(t) \qquad \gamma(t_0) = 0$$

$$\phi(x(t_f)) - \phi^L \le \gamma(t_f) w_0$$

$$\left(\frac{s^x{}_i(t)}{\sigma^x{}_i}\right)^2 \le \Delta\gamma(t) \qquad onde \quad i \in S_x = \{1, \cdots, nx\}$$

$$\left(\frac{s^y{}_j(t)}{\sigma^y{}_j}\right)^2 \le \Delta\gamma(t) \qquad onde \quad j \in S_y = \{1, \cdots, ny\}$$

$$\left(\frac{s^u{}_k(t)}{\sigma^u{}_k}\right)^2 \le \Delta\gamma(t) \qquad onde \quad k \in S_u = \{1, \cdots, nu\}$$

$$x^L \le x(t) + s^x(t) \le x^U$$

$$y^L \le y(t) + s^y(t) \le y^U$$

$$u^L \le u(t) + s^u(t) \le u^U$$

$$s_L^x \le s^x(t) \le s_U^x \qquad onde \quad s^x(t) \in \Re^{S_x} \tag{7.5}$$

$$s_L^y \le s^y(t) \le s_U^y \qquad onde \quad s^y(t) \in \Re^{S_y}$$

$$s_L^u \le s^u(t) \le s_U^u \qquad onde \quad s^u(t) \in \Re^{S_u}$$

onde $\gamma(t_f)$ é o novo objetivo geral a ser minimizado, $\Delta\gamma(t)$ o grau máximo de violação das restrições a cada instante de tempo, ϕ^L um valor utópico da função objetivo do *DAOP* original. Esse valor utópico é escolhido com base no seu significado físico. Geralmente ele é estabelecido como limite superior ou inferior do domínio da função objetivo (ex.: se o objetivo é a maximização da fração molar componente, o valor utópico pode ser 1 [(ϕ^L = -1]). O peso de sub-alcance do objetivo w_0 é o fator de peso que prioriza a viabilidade em detrimento da função objetivo original. O domínio é $0 \le w_0 \le 1$. Quando a minimização de violações das restrições é muito importante, pode ser usado um valor de $w_0 = 0.1$. $\left(\sigma_i^z\right)^2$ é a variância admissível da violação da restrição $s_i^z(t)$ de $z_i(t)$, $z \in \{x, y, u\}$, e podem ser obtidos através de estudos estatísticos de $s_i^z(t)$ com dados históricos. s_L^z e s_U^z são, respectivamente, mínimas e máximas violações das restrições permitidas com base na análise de processo do *DAOP*. É importante notar que não é necessário relaxar todas as variáveis $z(t)$, evitando a criação de otimização de grandes problemas. Se as variáveis são delimitadas em seus domínios ou não podem ficar relaxadas por qualquer outro motivo, suas restrições não podem ser reformuladas, incluindo variáveis de folga.

Para deixar a Equação 7.5 na forma padrão da Equação 6.1, podemos reescrever o problema como:

$$\min_{u,\Delta\gamma,s^x,s^y,s^u,p} \gamma(t_f)$$

s.a.

$$F(\dot{x}(t),x(t),y(t),u(t),p,t)=0; \qquad x(t_0)=x_0 \quad t\in[t_0,t_f]$$

$$\frac{d\gamma}{dt}=\Delta\gamma(t) \qquad \gamma(t_0)=0$$

$$\psi^{\phi}(t_f)=\gamma(t_f)w_0-\phi(x(t_f))+\phi^L$$

$$\psi_i{}^x(t)=\Delta\gamma(t)-\left(\frac{s^x{}_i(t)}{\sigma^x{}_i}\right)^2 \qquad onde \quad i\in S_x=\{1,\cdots,nx\}$$

$$\psi_j{}^y(t)=\Delta\gamma(t)-\left(\frac{s^y{}_j(t)}{\sigma^y{}_j}\right)^2 \qquad onde \quad j\in S_y=\{1,\cdots,ny\}$$

$$\psi_k{}^u(t)=\Delta\gamma(t)-\left(\frac{s^u{}_k(t)}{\sigma^u{}_k}\right)^2 \qquad onde \quad i\in S_u=\{1,\cdots,nu\}$$

$$g^x(t)=x(t)+s^x(t)$$

$$g^y(t)=y(t)+s^y(t)$$

$$g^u(t)=u(t)+s^u(t)$$

$$x^L\le g^x(t)\le x^U \qquad onde \quad g^x(t)\in\Re^{S_x}$$

$$y^L\le g^y(t)\le y^U \qquad onde \quad g^y(t)\in\Re^{S_y}$$

$$u^L\le g^u(t)\le u^U \qquad onde \quad g^u(t)\in\Re^{S_u} \qquad (7.6)$$

$$s_L^x\le s^x(t)\le s_U^x \qquad onde \quad s^x(t)\in\Re^{S_x}$$

$$s_L^y\le s^y(t)\le s_U^y \qquad onde \quad s^y(t)\in\Re^{S_y}$$

$$s_L^u\le s^u(t)\le s_U^u \qquad onde \quad s^u(t)\in\Re^{S_u}$$

$$\psi^{\phi}(t)\ge 0; \quad \psi^x(t)\ge 0; \quad \psi^y(t)\ge 0; \quad \psi^u(t)\ge 0;$$

Há duas estratégias de relaxamento, são elas: mover a restrição com o mesmo valor ao longo de todo o horizonte de otimização. Neste caso, estaremos relaxando as restrições em instantes de tempo onde não haveria necessidade. Assim, os perfis de controle também são alterados por este relaxamento. Na formulação do problema de otimização, é equivalente a definir a variável de folga como um parâmetro a ser otimizado. A segunda estratégia considera a variável de folga como uma variável de controle. Neste caso, as restrições são relaxadas nos intervalos de tempo em que o problema se torna inviável. As manipulações das variáveis de folga seguem o critério de mínimo relaxamento que torna o problema viável, como mostra a Figura 7.4. A segunda estratégia é mais complexa e mais adequada, pois altera as restrições somente nos momentos em que resolve a inviabilidade do problema de otimização. Para aplicação em sistemas de *DRTO*, pode-se substituir uma série de pequenos relaxamentos pelo maior relaxamento em um determinado intervalo do horizonte de otimização.

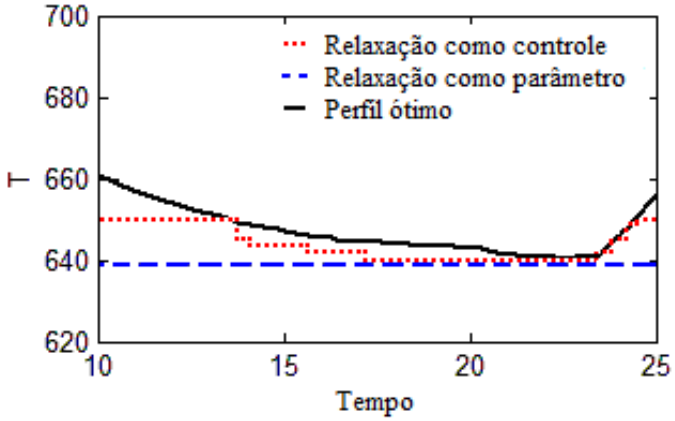


Figura 7.4 - Relaxamento de restrições através de variáveis de folga como parâmetro ou variável de controle.

7.3 Estudos de Casos, Resultados e Comentários

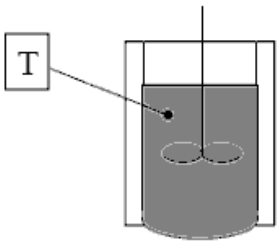
Três casos serão apresentados neste trabalho para mostrar as características da metodologia proposta. Em cada caso, três situações são mostradas, são elas: o problema original (com solução viável), o problema inviável (criando a problemática da inviabilidade pela modificação das posições das restrições) e o problema relaxado viável. Ao resolver o *DAOP* original e o *DAOP* inviável é usada a formulação padrão original apresentada na Equação 6.1. No caso de relaxamento automático, substitui-se o *DAOP* original por reformulação relaxada descrita pela equação 7.5. A mesma formulação relaxada foi usada nos pacotes de otimização dinâmica empregados para a solução dos problemas. Nesses casos foram estudadas as seguintes questões: conflito entre as especificações, inviabilidade inicial, tendência à inviabilidade inicial, e as mesmas situações para inviabilidade intermediária (no caminho).

No presente trabalho, os *NLP's* foram resolvidos usando o método do ponto interior implementado no código *IPOPT* (Wächter e Biegler, 2006) para a abordagem simultânea, através do pacote *DynoPC* (Lang e Biegler, 2007), e um método quasi-Newton implementado no código *SNOPT* (Gill et al., 2005) para a abordagem seqüencial, através do pacote *DyOS* (Brendel et al., 2008). Os estudos de casos foram feitos usando ambos os pacotes *DynoPC* e *DyOS*, e obtiveram-se resultados semelhantes para ambos os pacotes. Aqui apresentamos os resultados utilizando o *DyOS*. O solver de *NLP* e o integrador utilizado nas rodadas do *DyOS* são, respectivamente, *SNOPT* (Gill et al., 2005) e *LIMEX* (Deuflhard et al., 1987). O pacote *DyOS* tem uma estratégia de adaptação de malha baseado em *wavelets* (Binder et al., 2000), e o número inicial de pontos da malha discreta usado em todos os casos foi de oito elementos. Os valores das tolerâncias adotadas para o algoritmo de *NLP* e para o integrador do *DyOS* foram 10^{-6} para ambos.

7.3.1 Caso 1 – Otimização dinâmica de um reator batelada

Considere ao seguinte processo em batelada (vide Figura 7.5) onde temos como variável de controle a temperatura de reação (Ray, 1981; Cervantes et al., 2000). As condições iniciais do processo são: $C_A = 1.0$, $C_B = 0.0$, $C_C = 0.0$, dadas em frações molares dos componentes,

e T = 780 K. O objetivo de produção é maximizar a concentração do componente B (C_B) que é o produto desejado e evitar a produção do componente indesejado C (C_C).



$$A \xrightarrow{1} B \xrightarrow{2} C$$

Figura 7.5 - Caso 1 - Esquema e reações de um reator em batelada.

As equações de balanço de massa são dadas por:

$$\begin{aligned} &\frac{dC_A}{dt} = -k_1(T)\,C_A \\ &\frac{dC_B}{dt} = k_1(T)\,C_A - k_2(T)\,C_B \\ &C_A + C_B + C_C = 1 \end{aligned} \tag{7.7}$$

Sendo $k_1(T) = k_{01}e^{\left(-\frac{E_1}{RT}\right)}$ e $k_2(T) = k_{02}e^{\left(-\frac{E_2}{RT}\right)}$

onde $k_{1,0} = 65.5\,\mathrm{h}^{-1}$; $E_1/R = 5027.7\,\mathrm{K}$; $k_{2,0} = 1970\,\mathrm{h}^{-1}$; $E_2/R = 8044.31\,\mathrm{K}$ são os parâmetros cinéticos das reações 1 e 2 respectivamente. Sendo T a temperatura de reação em K e variável manipulada do reator.

No presente trabalho, este exemplo é dividido em três casos: (1) o problema original, sem restrições de concentração; (2) o problema inviável devido às restrições de temperatura de reação e concentração do componente C; e (3) problema com relaxamento das restrições.

7.3.1.1 Problema Original

O problema original possui solução viável, onde se deseja maximizar a concentração de B com um tempo final (t_f) de 25 horas da seguinte forma:

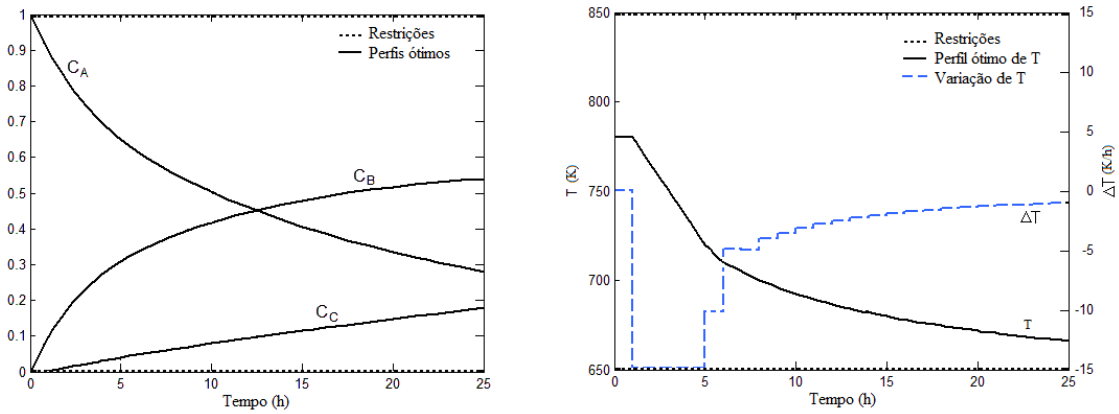
$$\max_{\Delta T(t)} \; C_B\left(t_f\right)$$

s.a.

$$\begin{aligned} &\frac{dC_A}{dt} = -k_1(T)\,C_A && C_A(0) = 1.0 \\ &\frac{dC_B}{dt} = k_1(T)\,C_A - k_2(T)\,C_B && C_B(0) = 0.0 \\ &C_A + C_B + C_C = 1 \\ &0 \le C_A, C_B, C_C \le 1 \\ &\frac{dT}{dt} = \Delta T && T(0) = 780\ K \\ &650 \le T(t) \le 850 \\ &-15 \le \Delta T(t) \le 15 \end{aligned} \tag{7.8}$$

Sendo $k_1(T) = k_{01}e^{\left(-\frac{E_1}{RT}\right)}$ e $k_2(T) = k_{02}e^{\left(-\frac{E_2}{RT}\right)}$

Impondo um perfil linear por partes para a variável de controle ΔT e restringindo o valor de T e de sua variação. Um perfil constante por partes em ΔT corresponde a um perfil linear por partes em T (a variável manipulada original). No mundo industrial, geralmente não é possível iniciar a ações de controle de qualquer ponto inicial, pois a otimização do processo usualmente parte do estado corrente da planta. As variáveis de controle manipuladas pelos *SDCD's* (Sistemas Digitais de Controle Distribuído) são implementadas na forma de pequenos degraus, limitados pelos operadores. Portanto, a melhor maneira de reproduzir a operação do mundo real é representar as variáveis manipuladas como variáveis de estado e impor alterações em degraus sobre a operação do processo. Perfis de ordem superiores não são geralmente implementados no mundo real. A solução ótima deste problema é mostrada na Figura 7.6.



(a) (b)

Figura 7.6 - Caso 1 - Solução ótima do problema original.
(a) Perfis dos estados C_A, C_B e C_C e (b) perfil de T e ΔT.

Uma solução viável foi encontrada, onde a concentração final de B foi de 0.5406 e o limite mínimo da temperatura de reação não foi atingido. Observe também que a concentração final do co-produto C foi de 0.1789, que deve ser a mínima possível.

7.3.1.2 Problema Inviável

Suponha agora que seja imposto um limite de concentração do co-produto indesejável C, onde a concentração máxima permitida é de 0.1. Este fato faz com que as restrições de máxima concentração de C e mínima temperatura de reação sejam concorrentes e conflitantes entre si, pois não conseguem ser satisfeitas ao mesmo tempo. Neste caso, impomos a restrição $0 \leq C_C \leq 0.1$ ao problema original o que resulta em problema inviável e evidencia o conflito de especificações.

Como pode ser visto na Figura 7.7, o otimizador procurou reduzir a temperatura de reação com sua taxa máxima de variação negativa. Note que a concentração final de C (0.1081) violou a restrição, e a temperatura de reação (650 K) atingiu seu limite inferior em alguns trechos. Esta configuração não permite encontrar qualquer solução viável, gerando uma

falha no otimizador. A única solução seria relaxar a restrição de máxima concentração de C, da temperatura mínima de reação ou da máxima variação da temperatura. A solução proposta é exatamente reformular o problema de otimização multiobjetivos onde as duas primeiras restrições citadas acima são relaxadas, já que a terceira é considerada como uma limitação física do processo.

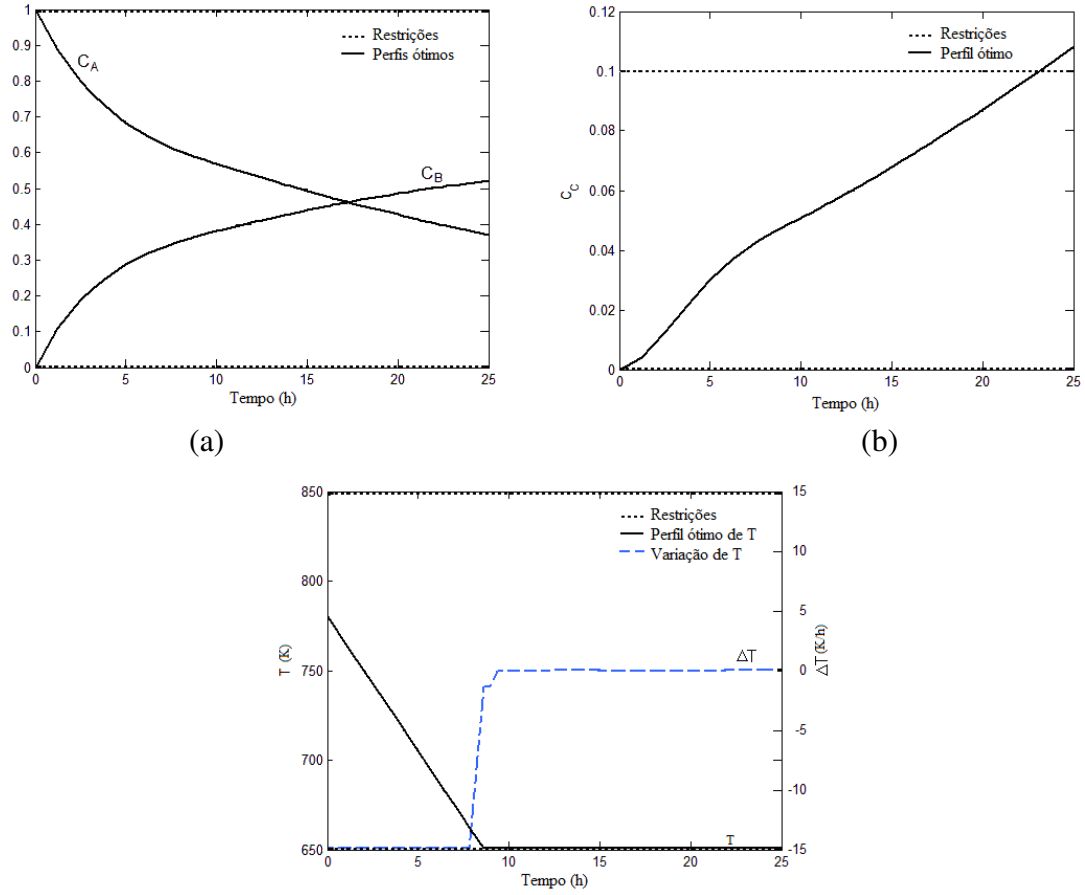


Figura 7.7 - Caso 1 - Melhor resultado do problema inviável.
(a) Perfis dos estados C_A e C_B, (b) perfil do estado C_C e (c) perfil de T e ΔT.

7.3.1.3 Problema Relaxado

Neste caso se resolve o problema de otimização dinâmica multiobjetivos citado acima. Os objetivos que concorrem entre si são: Maximizar C_B do tempo final, minimizar o relaxamento das restrições em C_C e T. O problema multiobjetivos é formulado na sua forma de implementação nos pacotes de otimização dinâmica (Equação 7.9). Nesta formulação, são acrescentadas as variáveis de controle $\Delta\gamma$, s_{CC} e s_T com perfis constantes por partes.

$$\min_{\Delta T,\Delta\gamma,s_T,s_{C_C}} \quad \gamma(t_f)$$

s.a.

$$\frac{dC_A}{dt} = -k_1(T)\,C_A \qquad C_A(0) = 1.0$$

$$\frac{dC_B}{dt} = k_1(T)\,C_A - k_2(T)\,C_B \qquad C_B(0) = 0.0$$

$$C_A + C_B + C_C = 1$$

$$\frac{dT}{dt} = \Delta T \qquad T(0) = 780\ K$$

$$\frac{d\gamma}{dt} = \Delta\gamma \qquad \gamma(0) = 0.0 \qquad (7.9)$$

$$g_{C_C}(t) = C_C - s_{C_C}$$

$$g_T(t) = T - s_T$$

$$\phi(t_f) = -C_B(t_f)$$

$$\Psi_\phi = w^\phi \gamma(t_f) - \phi(t_f) + \phi_L$$

$$\psi_\phi \geq 0$$

$$\psi_{C_C} = \Delta\gamma - \left(\frac{s_{C_C}}{\sigma_{C_C}}\right)^2$$

$$\psi_{C_C} \geq 0$$

$$\psi_T = \Delta\gamma - \left(\frac{s_T}{\sigma_T}\right)^2$$

$$\psi_T \geq 0$$

$$0 \leq C_A, C_B \leq 1$$

$$0 \leq g_{C_C}(t) \leq 0.1$$

$$650 \leq g_T(t) \leq 850$$

$$-15 \leq \Delta T \leq 15$$

$$-0.1 \leq s_{C_C} \leq 0.1$$

$$-10.0 \leq s_T \leq 10.0$$

O otimizador encontrou uma solução viável, como pode ser visto na Figura 7.8, onde procurou minimizar o relaxamento da temperatura mínima de reação e concentração máxima do componente C no final do processamento. Os limites das variáveis de folga são escolhidos com base nas tolerâncias definidas na analise de processo do *DAOP*. Os valores dos desvios padrões para as restrições relaxadas são $\sigma_{C_C} = 0.01$ e $\sigma_T = 1K$. Note que a concentração máxima de C foi de 0.1080, e a temperatura mínima foi de 649.1962 K. Os relaxamentos das respectivas restrições foram mínimos e o mais importante é que não houve falha do otimizador.

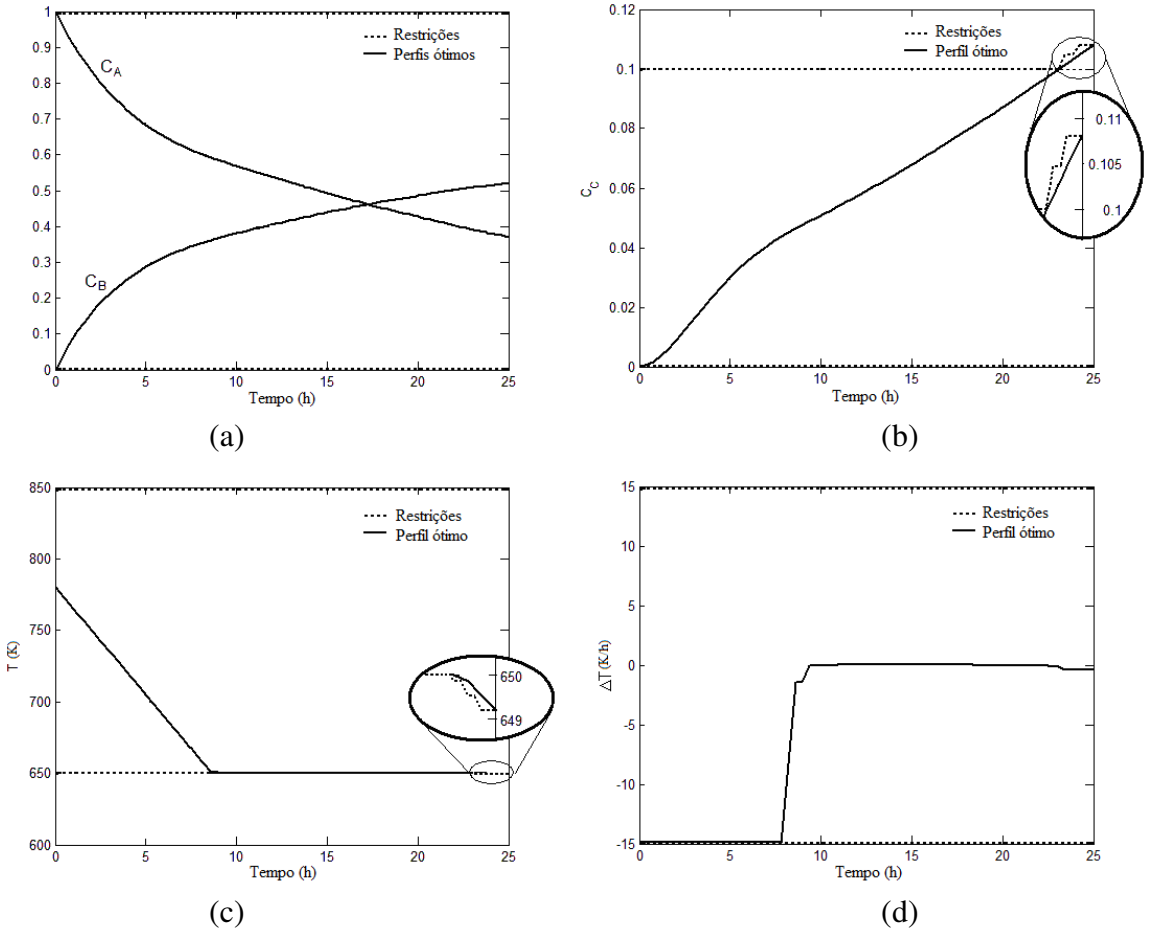
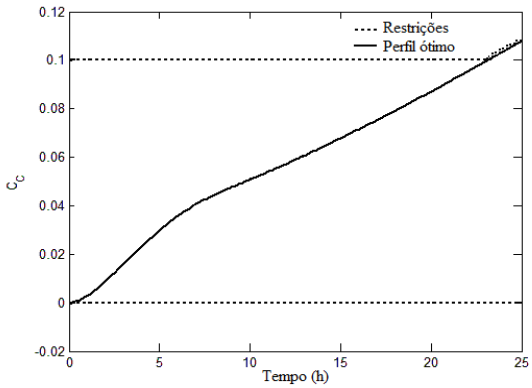
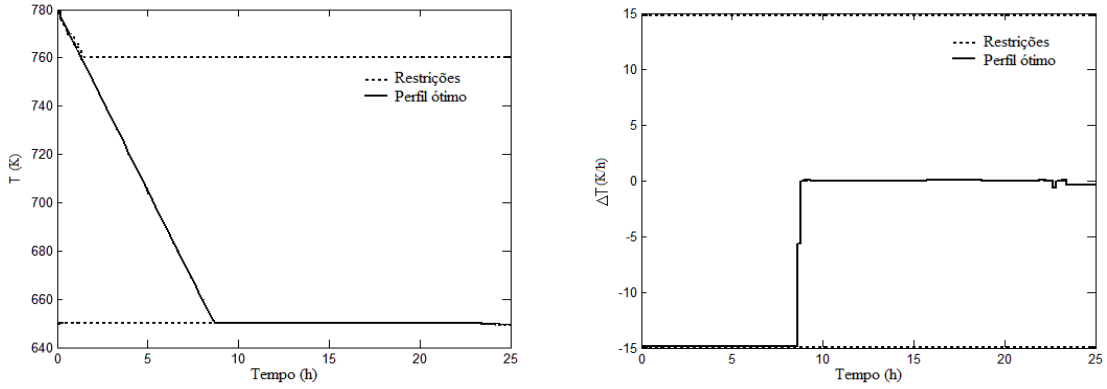


Figura 7.8 - Caso 1 - Solução do problema relaxado. (a) Perfis dos estados C_A e C_B, (b) perfil do estado C_C, (c) perfil de T e (d) perfil de variação de T.

Considere agora que tenhamos um problema de viabilidade inicial onde o limite máximo é movido para 760 K. Nesta condição, o problema se torna inviável pela sua condição inicial. Esta situação não é incomum na otimização dinâmica de uma planta real, pois usualmente as condições iniciais não podem ser escolhidas livremente. Neste caso, temos as seguintes restrições modificadas do problema relaxado: $-30.0 \le s_T(t) \le 30.0$ e $650 \le g_T(t) \le 760$. A solução relaxada fornece os perfis conforme é mostrado na Figura 7.9.



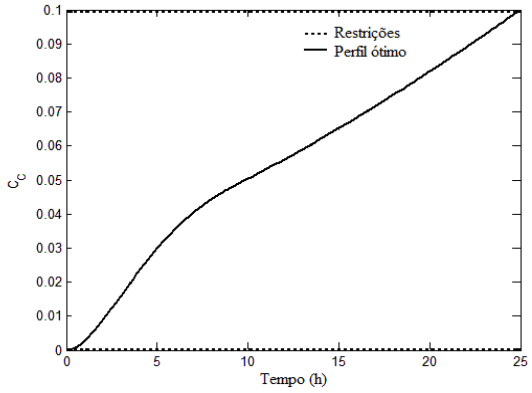
(a)



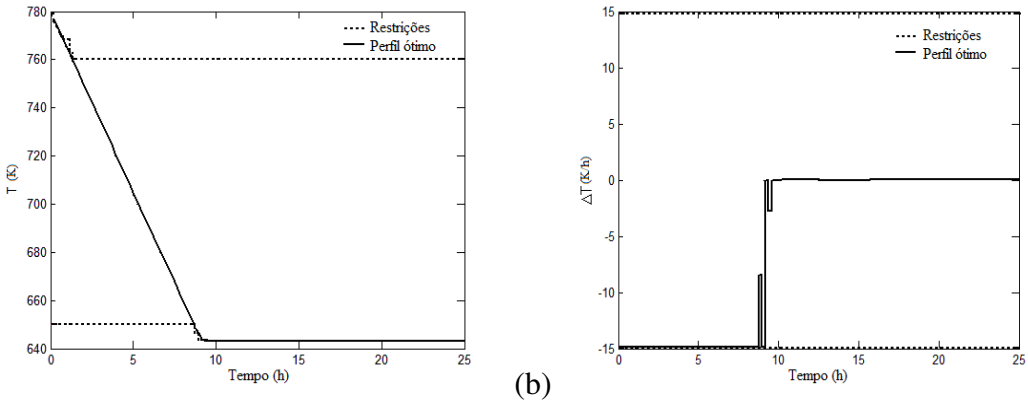
(b) (c)

Figura 7.9 - Caso 1 - Solução do problema relaxado – limite superior de T = 760 K.
(a) Perfil do estado C_C e (b) perfil de T e (c) perfil de variação de T.

Uma situação alternativa a ser explorada seria a não relaxação da restrição de concentração máxima de C. Nesta condição, o perfil de temperatura de reação sofre alterações para compensar esta imposição. Note na Figura 7.10, que houve uma relaxação maior na temperatura mínima de reação em torno de 7 K. Se esse relaxamento for aceitável, é preferível relaxar somente uma restrição. Neste caso, em uma primeira rodada, executa-se o otimizador com relaxamento em todas as restrições. Ao perceber que possa ter outra solução que tenha conveniência de processo, pode-se executar uma segunda rodada onde se endurece uma restrição (no caso a concentração de C).



(a)



(b) (c)

Figura 7.10 - Caso 1 - Solução do problema parcialmente relaxado.
(a) Perfil do estado C_C e (b) perfil de T e (c) de variação de T.

Em um segundo caso, temos um problema de viabilidade intermediária onde o limite mínimo é movido de 650 K para 660 K quando se passaram 10 horas de operação. Nesta condição, o problema se torna inviável no interior do intervalo de otimização. Esta situação não é comum na otimização dinâmica de uma planta real quando a receita é alterada ao longo do processamento em batelada onde as especificações das restrições são alteradas. Neste caso, temos as seguintes restrições modificadas do problema relaxado: $650 \le g_T(t) \le 790$ para $t \in [0, 10]$ e $660 \le g_T(t) \le 790$ para $t \in (10, 25]$. A solução relaxada fornece os perfis ótimos mostrados na Figura 7.11.

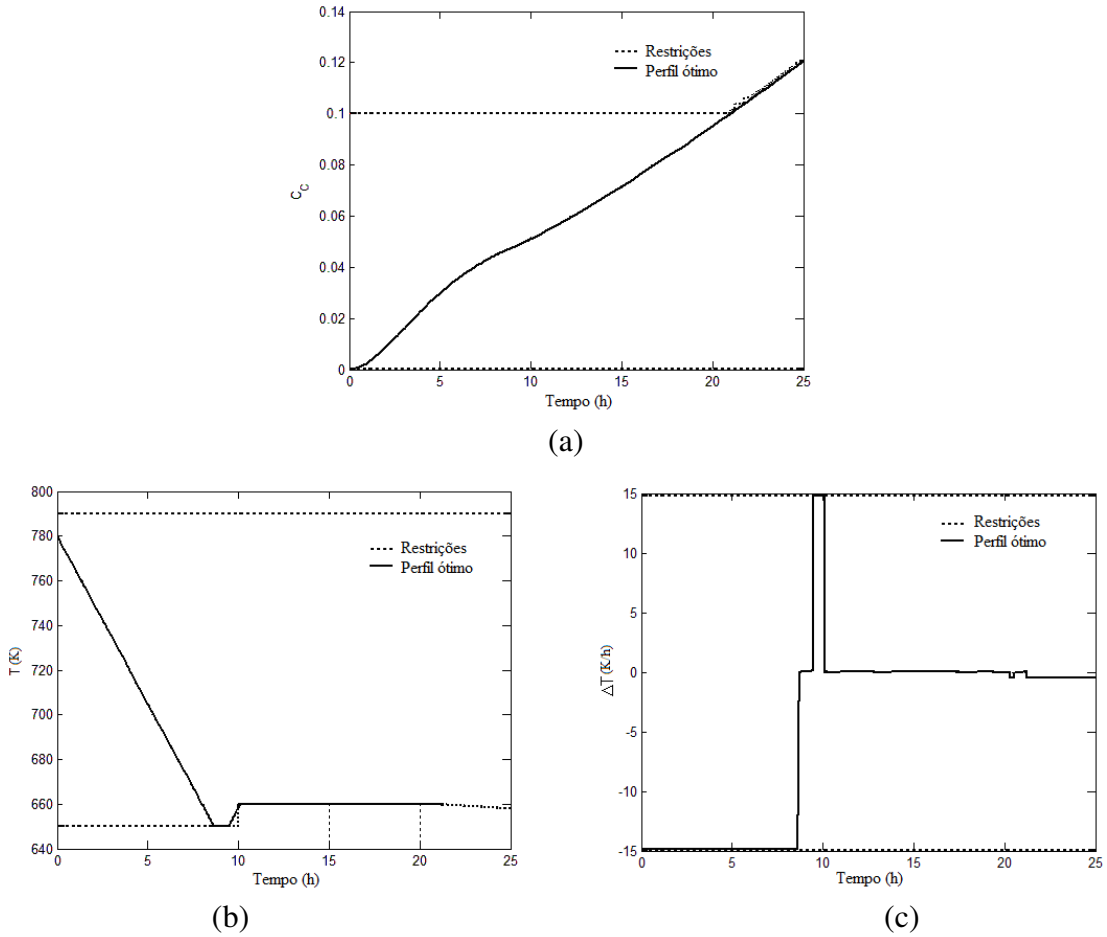


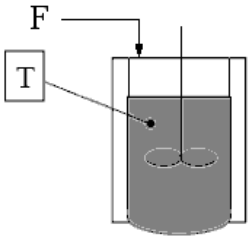
Figura 7.11 - Caso 1 - Solução do problema relaxado – inviabilidade intermediária. (a) Perfil do estado C_C e (b) perfil de T e (c) perfil de variação de T.

Note que o otimizador evitou a inviabilidade na mudança de receita ocorrida em 10 horas, antecipando a transição do processo. Além disso, teve que relaxar a temperatura e a concentração de C no final do horizonte de otimização.

7.3.2 Caso 2 – Otimização dinâmica de um reator semi-batelada não isotérmico

Seja um reator semi-batelada não-isotérmico com reações exotérmicas em série sujeitas às limitações de remoção de calor, conforme ilustra a Figura 7.12. Este problema tem duas variáveis de controle, a vazão de carga F do reagente B e a temperatura do reator (Srinivasan et al., 2003). O objetivo de produção é maximizar a quantidade do componente

C produzido em um horizonte de otimização pré-estabelecido. Há uma restrição de máxima taxa de calor produzido pela reação que deve ser obedecida ao longo do tempo. O volume final máximo no reator é imposto ao processo. As condições iniciais do processo são: $C_A(t_0)$ = 10.0 mol/L, $C_B(t_0)$ = 1.1685 mol/L, $C_C(t_0)$ = 0.0 mol/L, $C_{B,In}$ = 20.0 mol/L, $V(t_0)$ = 1.0 L, $F(t_0)$ = 0.5 L/h e T_0 = 35 °C.



$$A + B \overset{1}{\rightarrow} C \overset{2}{\rightarrow} D$$

Figura 7.12 - Caso 2 - Esquema e reações de um reator semi-batelada.

As equações do modelo são dadas por:

$$\begin{aligned}
&\frac{dc_A}{dt} = -k_1 c_A c_B - \frac{F}{V} c_A \\
&\frac{dc_B}{dt} = -k_1 c_A c_B + \frac{F}{V}\left(c_{B,In} - c_B\right) \\
&\frac{dc_C}{dt} = k_1 c_A c_B - k_2 c_C - \frac{F}{V} c_C \\
&\frac{dV}{dt} = F \\
&Q = \left[\left(-\Delta H_1\right) k_1 c_A c_B + \left(-\Delta H_2\right) k_2 c_C\right] V
\end{aligned} \tag{7.10}$$

e

$$k_1 = k_{01} e^{\left(-\frac{E_1}{R(T+273)}\right)} \qquad k_{01} = 4.0\ L/mol\,h;\ E_1 = 6000\frac{J}{mol};\ R = 8.31\frac{J}{mol\,K}$$

$$k_2 = k_{02} e^{\left(-\frac{E_2}{R(T+273)}\right)} \qquad k_{02} = 800.0\ L/mol\,h;\ E_2 = 20000\frac{J}{mol}$$

onde k_1, k_2, E_1 e E_2 são os parâmetros cinéticos das reações 1 e 2, respectivamente; T é a temperatura de reação em ^oC; c_A, c_B e c_C as concentrações dos componentes A, B e C, respectivamente; $c_{B,In}$ a concentração de alimentação do reagente B; F a vazão de carga do reagente B; V o volume útil do reator; Q é taxa de calor produzido pelas reações; e ΔH_1 = −30000 J/mol e ΔH_2 = −10000 J/mol os calores das reações 1 e 2, respectivamente.

Da mesma forma que no caso anterior, as variáveis manipuladas do reator são transformadas em variáveis de estado diferencias, e passa-se a manipular suas variações, para deixar o problema mais próximo da realidade industrial. Desta forma acrescentamos as seguintes equações:

$$\frac{dF}{dt} = \Delta F \text{ e } \frac{dT}{dt} = \Delta T \text{ sendo } F(t_0) = F_0 \text{ e } T(t_0) = T_0 \tag{7.11}$$

Em aplicações de otimização dinâmica online em tempo real, as condições iniciais do processo são lidas dos sistemas digitais de controle, portanto são inalteráveis e não são graus de liberdade dos sistemas. Desta forma, podemos dizer que F_0 e T_0 são condições dadas pelo processo. Há alguns casos onde se deseja planejar uma operação em batelada, neste caso F_0 e T_0 são variáveis livres. Este caso, não é objeto do presente trabalho.

Igualmente ao caso anterior, iremos dividir este exemplo em três casos: (1) o problema original, sem restrições de concentração; (2) o problema inviável devido às restrições de temperatura de reação e concentração do componente C; e (3) problema com relaxamento das restrições.

7.3.2.1 Problema Original

O problema original possui solução viável, onde se deseja maximizar a quantidade do produto C com um tempo final (t_f) de 0.5 horas, da seguinte forma:

$$\max_{\Delta T(t),\Delta F(t)} \; C_C\left(t_f\right)V\left(t_f\right)$$

s.a.

$$\begin{aligned}
&\frac{dc_A}{dt} = -k_1 c_A c_B - \frac{F}{V}c_A && C_A(0) = 10.0\ {}^{mol}\!/_{L}\\
&\frac{dc_B}{dt} = -k_1 c_A c_B + \frac{F}{V}\left(c_{B,In} - c_B\right) && C_B(0) = 1.1685\ {}^{mol}\!/_{L}\\
&\frac{dc_C}{dt} = k_1 c_A c_B - k_2 c_C - \frac{F}{V}c_C && C_B(0) = 0.0\ {}^{mol}\!/_{L}\\
&\frac{dV}{dt} = F && V(0) = 1.0\ L\\
&\frac{dF}{dt} = \Delta F && F(0) = 0.50\ {}^{L}\!/_{h}\\
&\frac{dT}{dt} = \Delta T && T(0) = 35.0\ {}^{o}C\\
&Q = \left(-\Delta H_1\right)k_1 c_A c_B + \left(-\Delta H_2\right)k_2 c_C V\\
&0.0 \le Q(t) \le 140\ \ J/h\\
&0.0 \le V\left(t_f\right) \le 1.1 L\\
&20^{o}C \le T\left(t\right) \le 50^{o}C\\
&0.0 \le F\left(t\right) \le 1.0 L/h\\
&-100 \le \Delta F\left(t\right) \le 100\ L/h^2\\
&-1000 \le \Delta T\left(t\right) \le 1000\ K/h
\end{aligned} \qquad (7.12)$$

impondo às variáveis de controle ΔF e ΔT perfis constantes por partes. Neste caso, são restritas a vazão de carga e sua variação, a temperatura de reação e sua variação, a carga térmica da camisa de refrigeração e o volume máximo de reação. Não há restrições de concentrações de reagentes e produtos. A solução viável atingiu o limite de carga térmica do reator, porém não inviabilizou a solução do problema de otimização dinâmica, como

mostra a Figura 7.13. Note também que a concentração máxima do componente *B* na reação foi de 1.65 mol/ L.

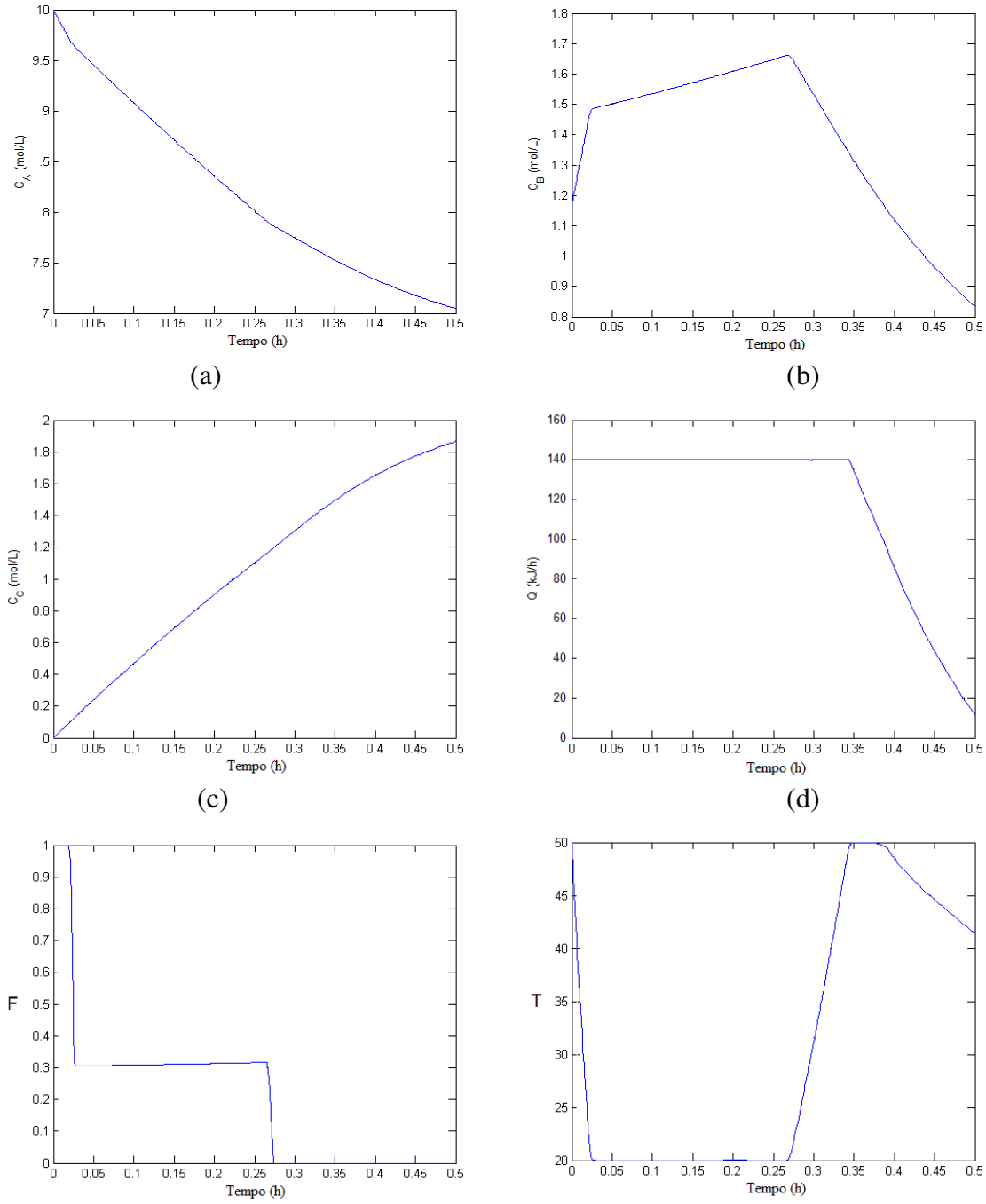
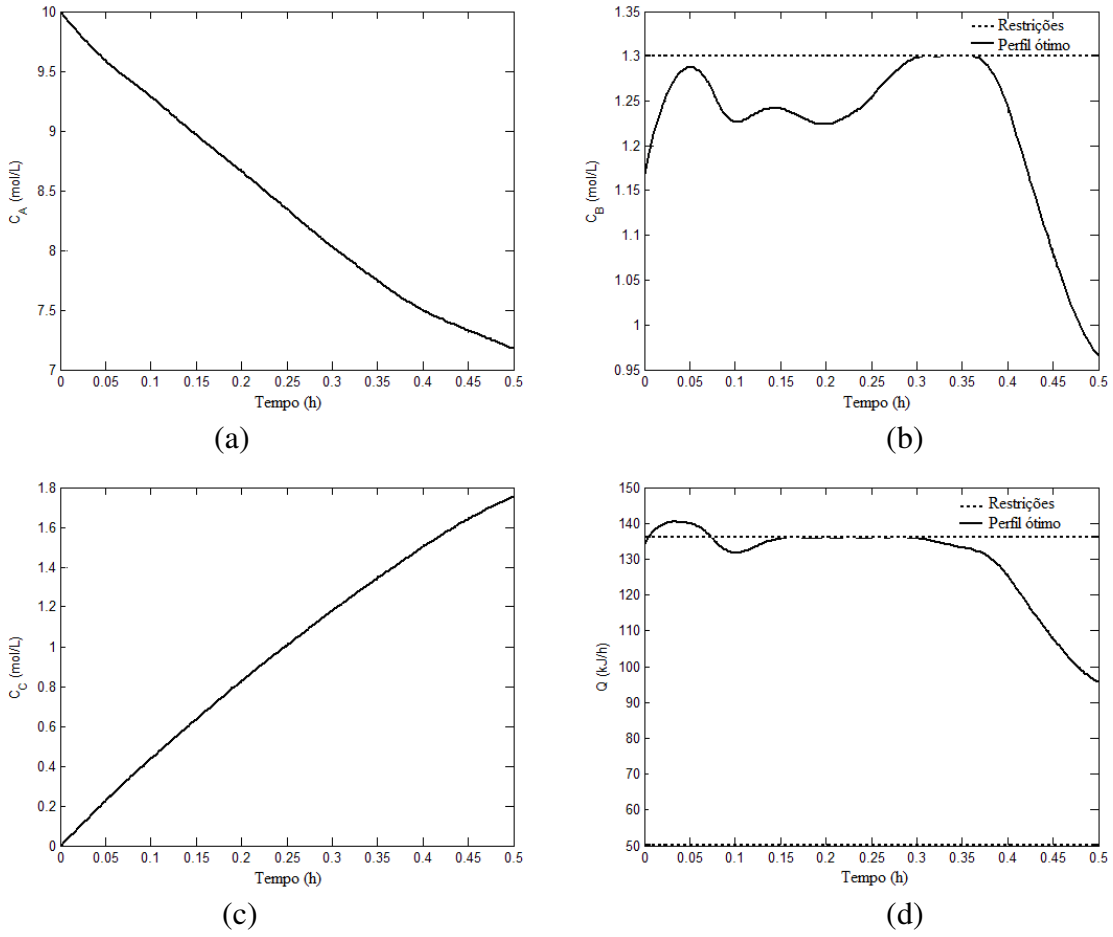


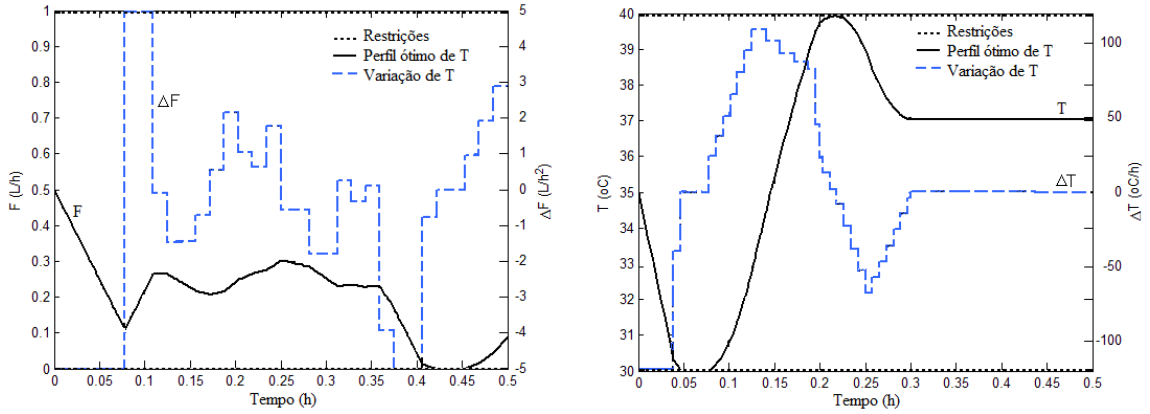
Figura 7.13 - Caso 2 - Solução ótima do problema original.
(a) Perfis dos estados C_A, (b) C_B, (c) C_C, (d) carga térmica Q, (e) perfil da vazão F e (f) perfil de T.

Resolvendo problemas de especificações das restrições

7.3.2.2 Problema Inviável

Considere agora que seja imposto um limite de concentração do co-produto indesejável B, onde a concentração máxima permitida é de 1.3 mol/L e uma carga térmica máxima de 136 kJ/h. Esta estrutura faz com que as restrições de máxima concentração de B e máxima carga térmica no reator Q sejam concorrentes e conflitantes entre si, pois não conseguem ser satisfeitas ao mesmo tempo. Além disso, foi reduzida a faixa de temperatura de reação na operação do sistema. Neste caso, impomos as restrições $0.0 \le C_B(t) \le 1.3$, $50 \le Q(t) \le 136$ e $30.0 \le T(t) \le 40.0$ no problema original o que resulta em problema inviável e evidencia o conflito de especificações, como mostram os resultados na Figura 7.14.





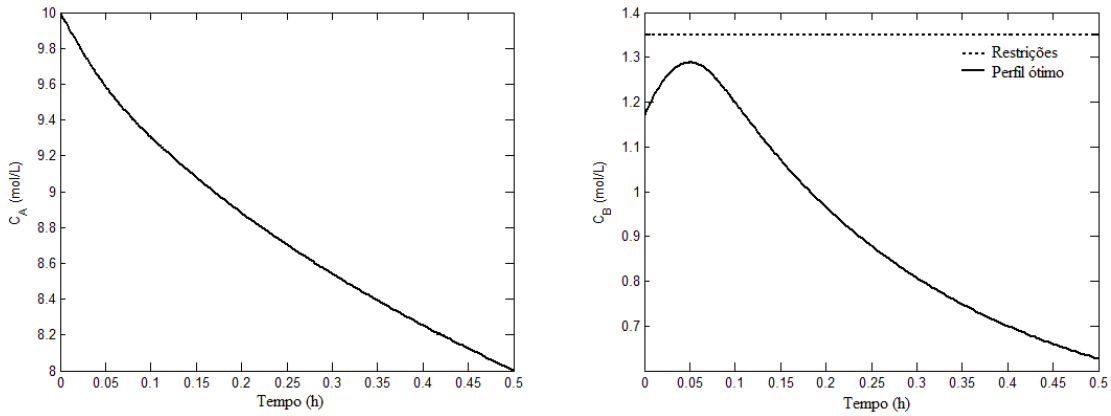
(e) (f)

Figura 7.14 - Caso 2 - Melhor resultado do problema inviável.
(a) Perfis dos estados C_A, (b) C_B, (c) C_C, (d) carga térmica Q, (e) F, ΔF, (f) T e ΔT.

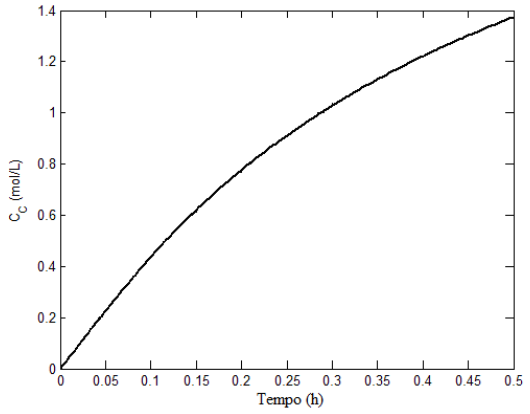
O otimizador procurou controlar a concentração máxima de B, porém a carga térmica máxima do resfriador foi violada, e a temperatura de reação atingiu seu limite inferior no mesmo trecho. Esta configuração não permite encontrar qualquer solução viável, gerando uma falha no otimizador. A solução deste problema seria relaxar as restrições máxima carga térmica, de mínima temperatura de reação e da máxima concentração de B. A solução proposta a seguir é reformular o problema como otimização multiobjetivos onde as restrições citadas acima são relaxadas.

7.3.2.3 Problema Relaxado

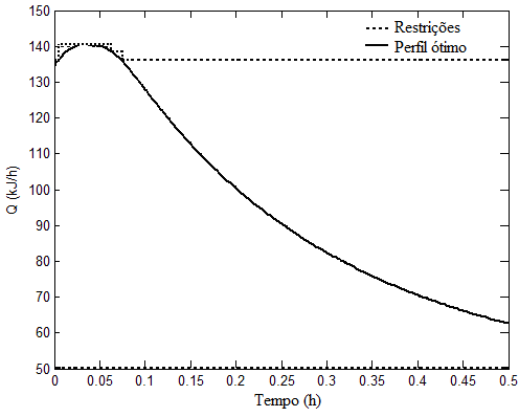
Neste caso se resolve o problema de otimização dinâmica multiobjetivos citado acima, com os seguintes objetivos que concorrem entre si: Maximizar C_B do tempo final, minimizar o relaxamento da restrição C_B, Q e T. O problema multiobjetivos é formulado da mesma forma que no caso anterior. Os resultados são apresentados na Figura 7.15. Note que as restrições de C_B e T não precisaram ser relaxadas. Os valores dos desvios padrões para as restrições relaxadas são $\sigma_{C_B} = 0.01$ e $\sigma_T = 0.1\,^oC$. Isto se deve ao fato que o problema minimiza as relaxações das restrições para viabilizar a solução, fazendo isto de forma algorítmica, sem a necessidade de quaisquer heurísticas.



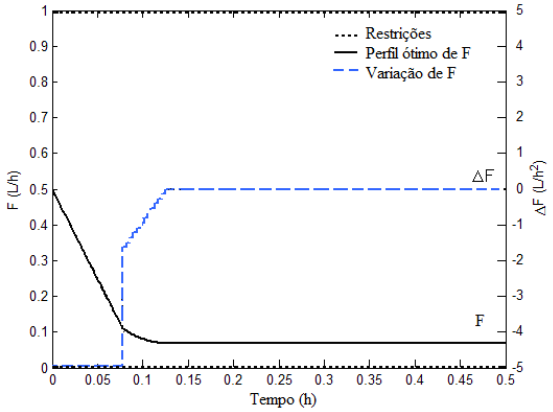
(a) (b)



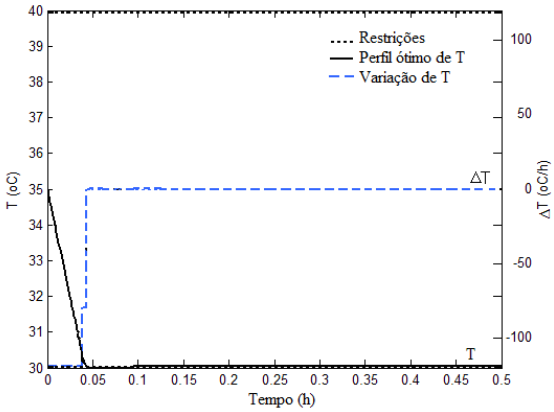
(c)



(d)



(e)



(f)

Figura 7.15 - Caso 2 - Solução do problema relaxado.
(a) Perfis dos estados C_A, (b) C_B, (c) C_C, (d) carga térmica Q, (e) *F*, *ΔF*, (f) *T* e *ΔT*.

Resolvendo conflitos de especificações das restrições

7.3.2.4 Problema Inviável

Considere agora, o mesmo problema onde a concentração máxima do co-produto indesejável B é imposta em 1.2 mol/L, o limite máximo da taxa de calor no reator é movido para 160 kJ/h, e a faixa de operação da temperatura de reação é ampliada ($30.0 \le T(t) \le 60.0$). Isto resulta em um segundo problema inviável onde C_B é muito restritivo, causando conflito e competição entre estas restrições, pois as mesmas não podem ser satisfeitas ao mesmo tempo. A Figura 7.16 mostra este cenário. Note que a restrição de C_B foi violada e os perfis de controle completamente alterados em relação ao caso original, ou seja, houve falha no otimizador para resolver este problema.

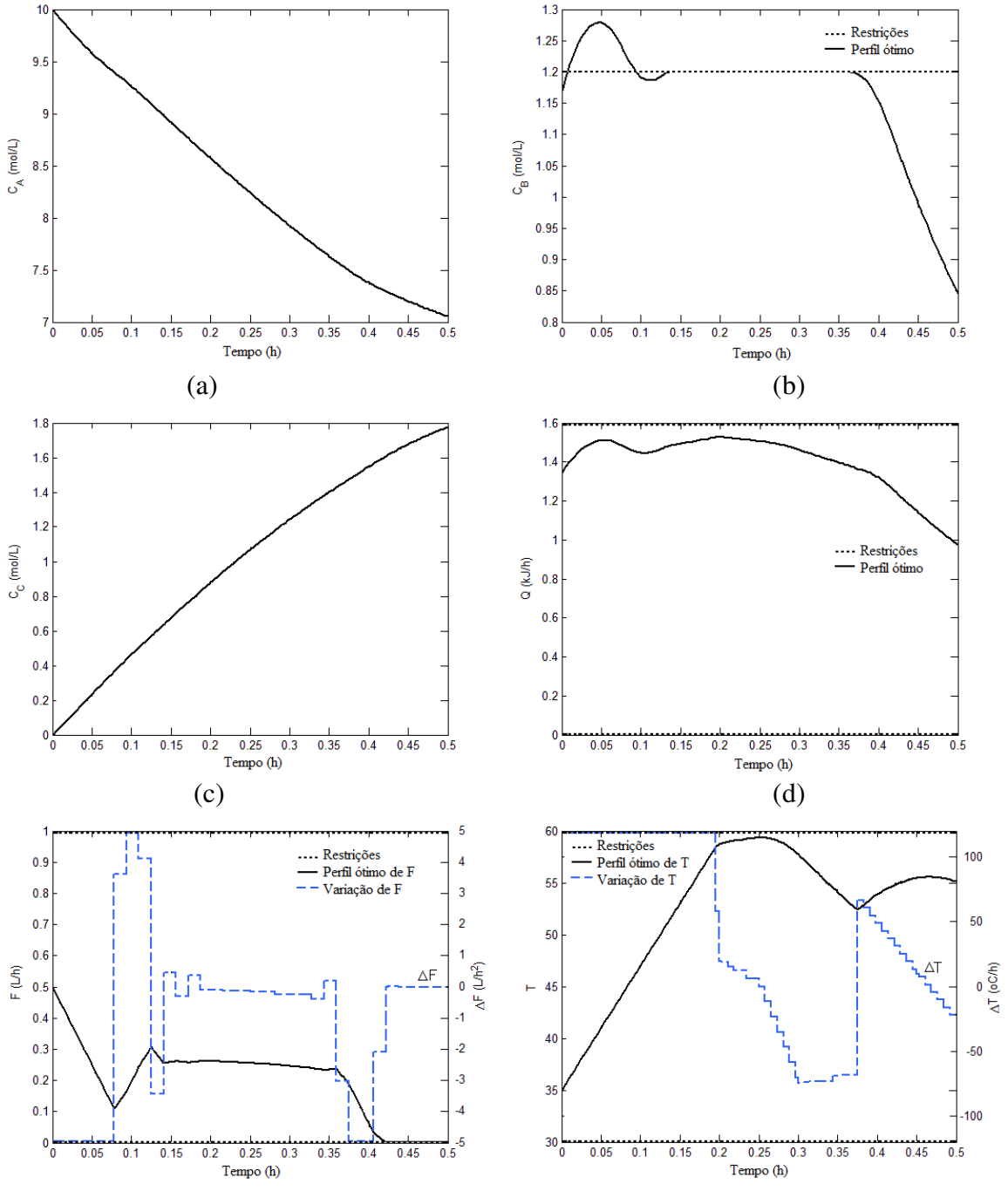


Figura 7.16 - Caso 2 - Melhor resultado do problema inviável – conflito de especificações. (a) Perfis dos estados C_A, (b) C_B, (c) C_C, (d) carga térmica Q, (e) F, ΔF, (f) T e ΔT.

7.3.2.5 Problema Relaxado

Da mesma forma que o caso anterior, propõe-se o mesmo relaxamento e utilizando a mesma estrutura do problema, porém alterando os limites das variáveis, conforme descrito acima. Note que o otimizador só relaxou C_B, mantendo os limites de Q e T nas suas posições originais, como mostra a Figura 7.17. Neste caso, a questão se resume em resolver a inviabilidade inicial problema original. Ao resolver o problema relaxado, tem-se

como consequência a abediência a outras restrições. Isto aconteceu neste caso, devido a estrutura do problema relaxado. Poderá haver outras situações onde seja mais conveniente relaxar mais de uma restrição (como pode ser visto na Figura 7.9, do Caso 1).

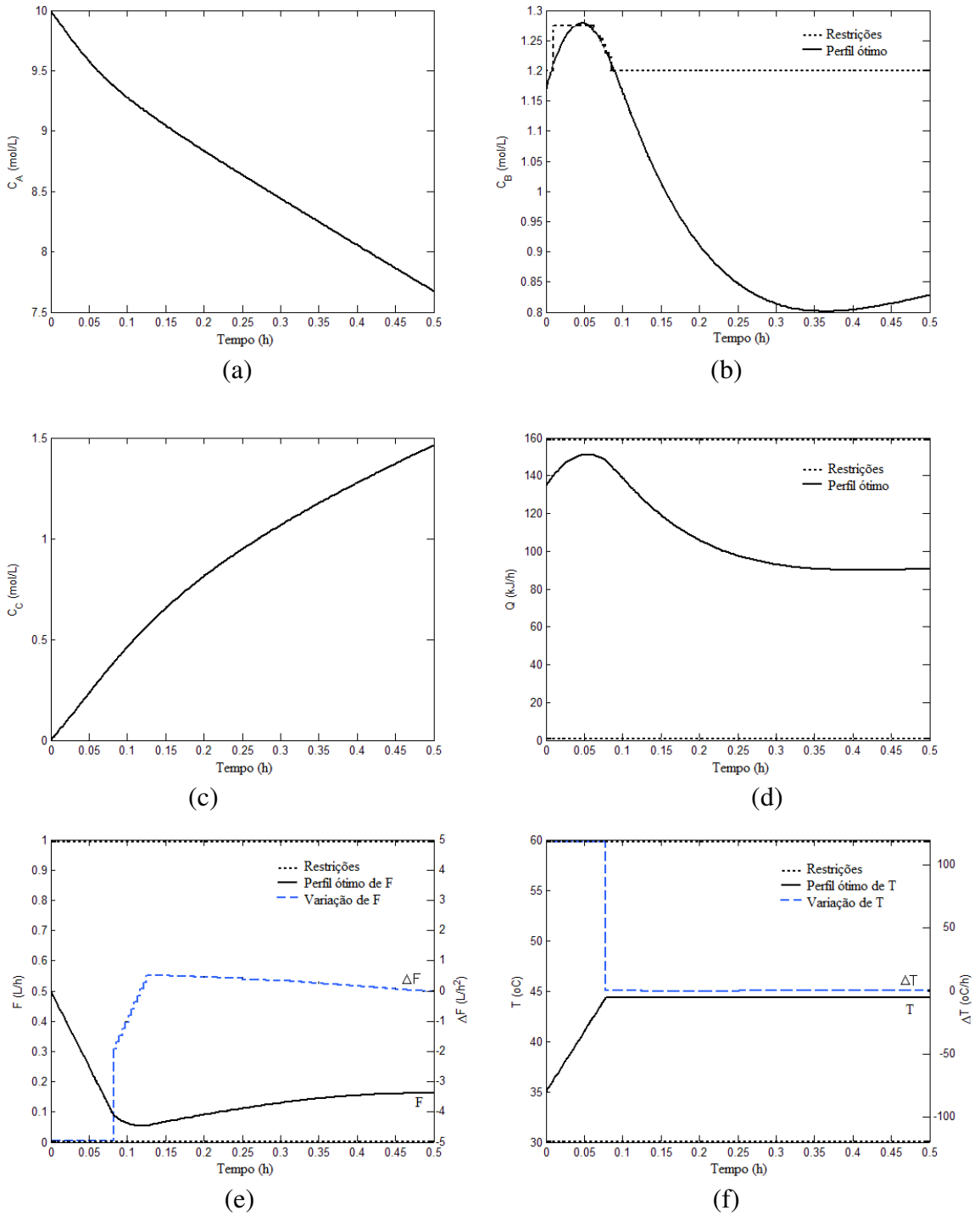
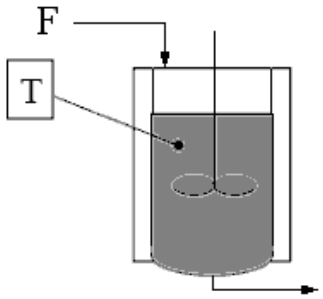


Figura 7.17 - Caso 2 - Solução do problema relaxado – conflito de especificações.
(a) Perfis dos estados C_A, (b) C_B, (c) C_C, (d) carga térmica Q, e (e) F, ΔF, (f) T e ΔT.

7.3.3 Caso 3 – Otimização dinâmica de um reator contínuo

Considere um sistema com múltiplos estados estacionários nas saídas, onde um mesmo valor da variável manipulada resulta em diferentes valores da controlada. Este sistema consiste de um reator *CSTR* não-isotérmico com duas reações exotérmicas irreversíveis em série. Este reator tem uma camisa de resfriamento onde a vazão de água Q_C é manipulada neste problema (Tlacuahuac et al., 2008), como mostra a Figura 7.18. Note que este processo tem um comportamento não-linear complexo, como mostram os diagramas de bifurcação da Figura 7.19. O diagrama principal é a temperatura de reação (x_3) versus vazão de água de resfriamento (Q_C), como mostra a Figura 7.19c.



$$A \xrightarrow{1} B \xrightarrow{2} C$$

Figura 7.18 - Caso 3 - Esquema e reações de um reator contínuo.

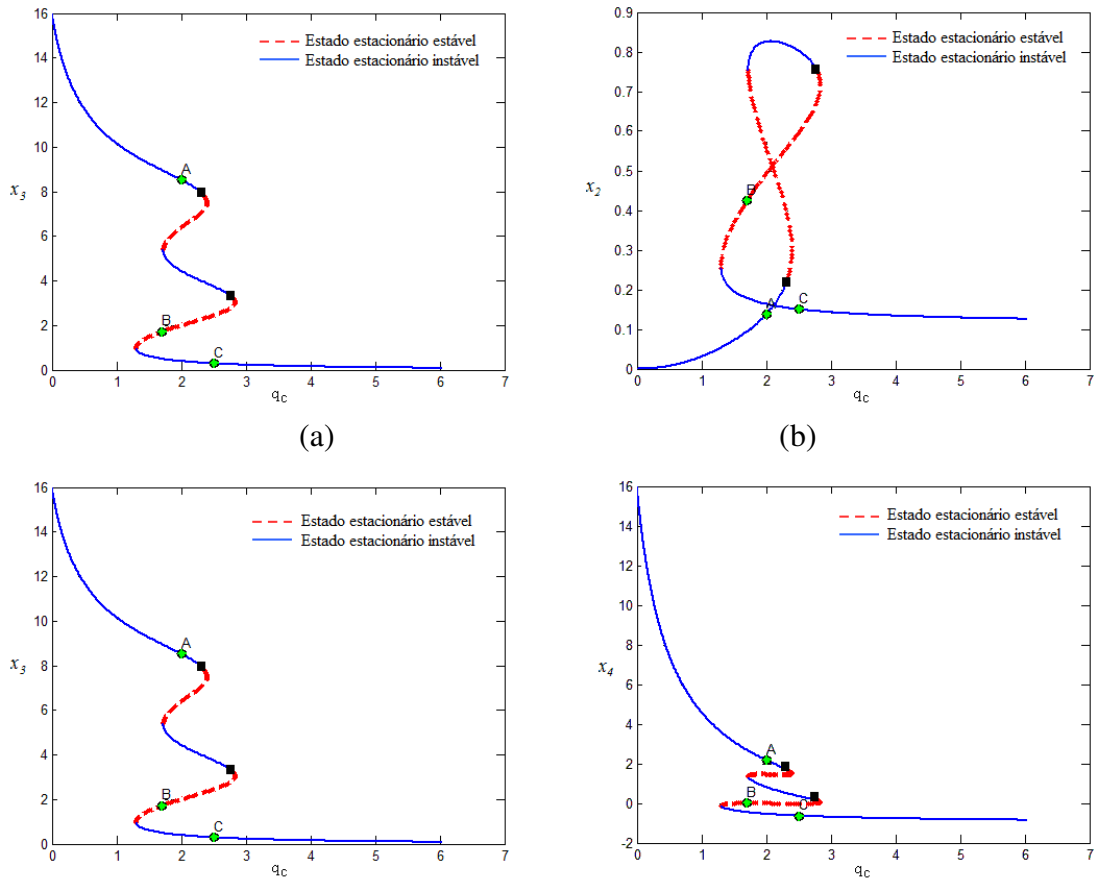


Figura 7.19 - Caso 3 - Diagramas de bifurcação de *CSTR* com dois pontos de bifurcação Hopf (■).

O modelo matemático que descreve o processo é resultado das equações de balanço de massa e de energia e pode ser escrito como:

$$\frac{dC_A}{dt} = \frac{Q}{V}\left(C_{Af} - C_A\right) - k_1(T)C_A$$
$$\frac{dC_B}{dt} = \frac{Q}{V}\left(C_{Bf} - C_B\right) - k_2(T)C_B + k_1(T)C_A$$
$$\frac{dT}{dt} = \frac{Q}{V}\left(T_f - T\right) + k_1(T)C_A\frac{\left(-\Delta H_A\right)}{\rho C_p} + k_2(T)C_B\frac{\left(-\Delta H_B\right)}{\rho C_p} - \frac{UA}{\rho C_p V}\left(T - T_c\right) \qquad (7.13)$$
$$\frac{dT_c}{dt} = \frac{Q_c}{V_c}\left(T_{cf} - T_c\right) + \frac{UA}{\rho_c C_{pc} V_c}\left(T - T_c\right)$$

onde as constantes cinéticas são: $k_1(T) = k_{01}e^{\left(\frac{-E_1}{RT}\right)}$ e $k_2(T) = k_{02}e^{\left(\frac{-E_2}{RT}\right)}$

Reescrevendo o modelo em uma forma adimensional, tem-se:

$$\frac{dx_1}{d\tau} = q\left(x_{1f} - x_1\right) - \phi\eta\left(x_3\right)x_1$$
$$\frac{dx_2}{d\tau} = q\left(x_{2f} - x_2\right) - \phi S\eta_2\left(x_3\right)x_2 + \phi\eta\left(x_3\right)x_1 \qquad (7.14)$$
$$\frac{dx_3}{d\tau} = q\left(x_{3f} - x_3\right) + \delta\left(x_4 - x_3\right) + \beta\phi\left[\eta\left(x_3\right)x_1 + \alpha S\eta_2\left(x_3\right)x_2\right]$$
$$\frac{dx_4}{d\tau} = \delta_1\left[q_c\left(x_{4f} - x_4\right) + \delta\delta_2\left(x_3 - x_4\right)\right]$$

onde x_1 é a concentração adimensional do reagente A; x_2 é a concentração adimensional do reagente B; x_3 é a temperatura adimensional do reator; x_4 é a temperatura adimensional da serpentina de resfriamento. Sendo que os parâmetros do modelo são definidos na Tabela 7.1.

Tabela 7.1 - Variáveis e parâmetros adimensionais do *CSTR*.

$x_1 = \frac{C_A}{C_{Af0}}$	$x_2 = \frac{C_B}{C_{Af0}}$	$\delta_1 = \frac{V}{V_c}$	$\delta_2 = \frac{\rho C_p}{\rho_c C_{pc}}$
$x_3 = \Gamma\left(\frac{T - T_{f0}}{T_{f0}}\right)$	$x_4 = \Gamma\left(\frac{T_c - T_{f0}}{T_{f0}}\right)$	$S = \frac{k_2\left(T_{f0}\right)}{k_1\left(T_{f0}\right)}$	$\phi = \left(\frac{V}{Q_0}\right)k_1\left(T_{f0}\right)$
$\Gamma = \frac{E_1}{RT_{f0}}$	$\tau = \left(\frac{Q_0}{V}\right)t$	$\alpha = \frac{-\Delta H_B}{-\Delta H_A}$	$\beta = \frac{-\Delta H_A C_{Af0}\Gamma}{\rho C_p T_{f0}}$
$q = \frac{Q}{Q_0}$	$q_c = \frac{Q_c}{Q_0}$	$x_{1f} = \frac{C_{Af}}{C_{Af0}}$	$x_{2f} = \frac{C_{Bf}}{C_{Af0}}$
$\psi = \frac{E_2}{E_1}$	$\delta = \frac{UA}{\rho C_p Q_0}$	$x_{3f} = \Gamma\left(\frac{T_f - T_{f0}}{T_{f0}}\right)$	$x_{4f} = \Gamma\left(\frac{T_{cf} - T_{f0}}{T_{f0}}\right)$
$\eta(x_3) = e^{\left[x_3\middle/\left(1+\frac{x_3}{\Gamma}\right)\right]}$	$\eta_2(x_3) = e^{\left[\psi x_3\middle/\left(1+\frac{x_3}{\Gamma}\right)\right]}$		

Os parâmetros utilizados neste caso estão na Tabela 7.2 abaixo.

Tabela 7.2 - Valores das variáveis e parâmetros do modelo (Tlacuahuac et al., 2008)

x_{1f}	x_{2f}	x_{3f}	x_{4f}	q	α	β
1.0	0.0	0.0	-1.0	1.0	1.0	8.0
ϕ	S	ψ	δ	δ_1	δ_2	Γ
0.133	0.01	1.0	1.0	10.0	1.0	1000

O problema de otimização dinâmica deste processo é representado na seguinte forma, onde o objetivo de otimização consiste em determinar as trajetórias de transições ótimas entre estados estacionários de pontos de operação, ou seja, minimizar a integral dos desvios da temperatura de reação em relação ao seu *setpoint* em cada transição proposta (Equação 7.15).

$$\min_{z,u} \int_0^{\theta} \left\| z(t) - \hat{z} \right\|^2 dt$$

s.a.

$$\begin{aligned} &\frac{dz}{dt} = F(z,u,t); \qquad z(0) = z_0 \\ &z_L \le z(t) \le z_U \\ &u_L \le u(t) \le u_U \end{aligned} \tag{7.15}$$

onde z são as variáveis de estado, u são as variáveis de controle e $\hat{z}$ é o *setpoint* da variável de estado.

O estudo deste processo, usualmente consiste dos controles das transições A $\rightarrow$ B, A $\rightarrow$ C e B $\rightarrow$ C (Tlacuahuac et al., 2008). Neste artigo, iremos focalizar a solução do problema de viabilidade da transição A $\rightarrow$ C. A Tabela 7.3 mostra os pontos de operação destas transições.

Tabela 7.3 - Estados estacionários nominais.

	A	B	C
x_1	0.0016	0.5917	0.8495
x_2	0.1387	0.4249	0.1503
x_3	8.5188	1.7306	0.2871
x_4	2.1729	0.0013	-0.6323
Q_c	2.00	1.70	2.50

O problema multiobjetivos referente à otimização e viabilização das soluções deste sistema pode ser escrito da seguinte forma:

$$\min_{\Delta\gamma, x_2, s_{x_2}} \quad \gamma(t_f)$$

s.a.

$$\begin{aligned}
&\frac{dx_1}{dt} = q(x_{1f} - x_1) - \phi\eta(x_3)x_1 \\
&\frac{dx_2}{dt} = q(x_{2f} - x_2) - \phi S\eta_2(x_3)x_2 + \phi\eta(x_3)x_1 \\
&\frac{dx_3}{dt} = q(x_{3f} - x_3) + \delta(x_4 - x_3) + \beta\phi[\eta(x_3)x_1 + \alpha S\eta_2(x_3)x_2] \qquad (7.16) \\
&\frac{dx_4}{dt} = \delta_1(q_c(x_{4f} - x_4) + \delta\delta_2(x_3 - x_4)) \\
&x(0) = \{A, \quad B \quad ou \quad C\} \\
&\frac{d\gamma}{dt} = \Delta\gamma \qquad\qquad \gamma(0) = 0 \\
&\frac{dISE}{dt} = (x_3 - x_3^{SP})^2 \qquad\qquad ISE(0) = 0 \\
&g_{x_2} = x_2 - s_{x_2} \\
&\psi_{ISE}(t_f) = w^{ISE}\gamma(t_f) - ISE(t_f) + ISE_L \\
&\psi_{ISE}(t_f) \geq 0 \\
&\psi_{x_2}(t) = \Delta\gamma(t) - \left(\frac{s_{x_2}(t)}{\sigma_{x_2}}\right)^2 \\
&\psi_{x_2}(t) \geq 0 \\
&0 \leq x_1 \leq 1 \\
&0 \leq x_3 \leq 10 \\
&0 \leq g_{x_2} \leq 0.55 \\
&0 \leq Q_c \leq 7.0 \\
&-0.1 \leq s_{x_2} \leq 0.1
\end{aligned}$$

onde ISE é a integral do erro ao quadrado (o desvio da temperatura do reator do seu *setpoint*, x_3^{SP}). O valor do desvio padrão para a restrição relaxada é $\sigma_{x_2} = 0.01$.

7.3.3.1 Problema Original

No caso da transição A → C, o problema se mostrou viável e o ótimo foi encontrado, como mostra a Figura 7.20. Pode-se observar neste caso que há um *overshoot* acentuado no perfil de C_B (x_2). Note também, pelos diagramas de bifurcação, que o otimizador sugere que a vazão de água de resfriamento vá logo para o seu valor máximo, ou seja, resfrie o reator o mais rápido possível.

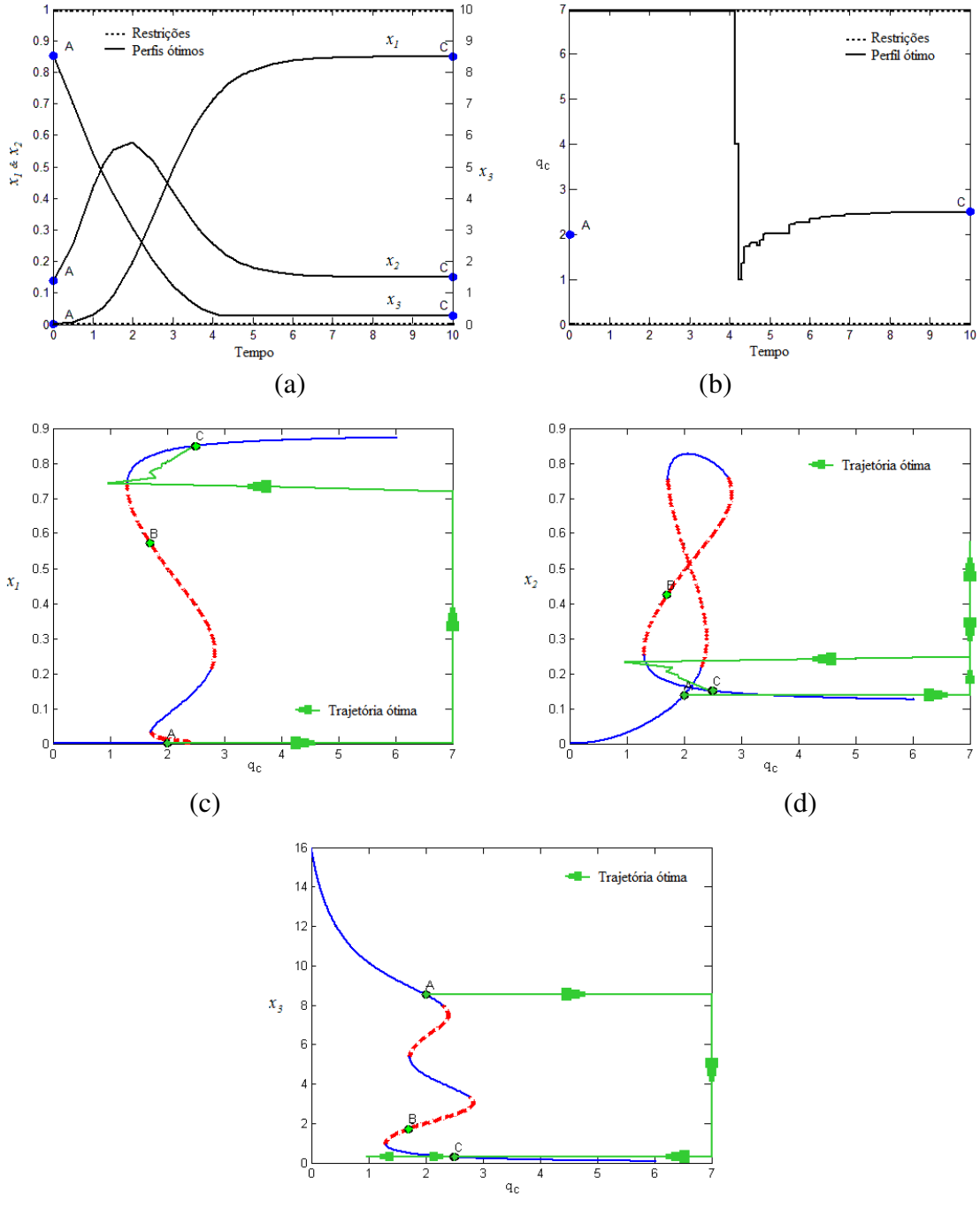


Figura 7.20 - Caso 3 - Solução do problema original.
(a) Perfis dos estados C_A (x_1), C_B (x_2) e T (x_3). (b) Perfil da variável de controle Q_C. (c, d, e) Diagramas de bifurcação das variáveis de estado para a transição A → C.

7.3.3.2 Problema Inviável

Considere agora que seja imposto um limite de concentração de B, onde a concentração máxima permitida é de 0.55 com uma vazão de água de resfriamento máxima de 7.0. Esta condição faz com que as restrições de máxima concentração de B e máxima vazão na

camisa do reator Q_C sejam concorrentes e conflitantes entre si, pois não conseguem ser satisfeitas ao mesmo tempo, como mostra a Figura 7.21.

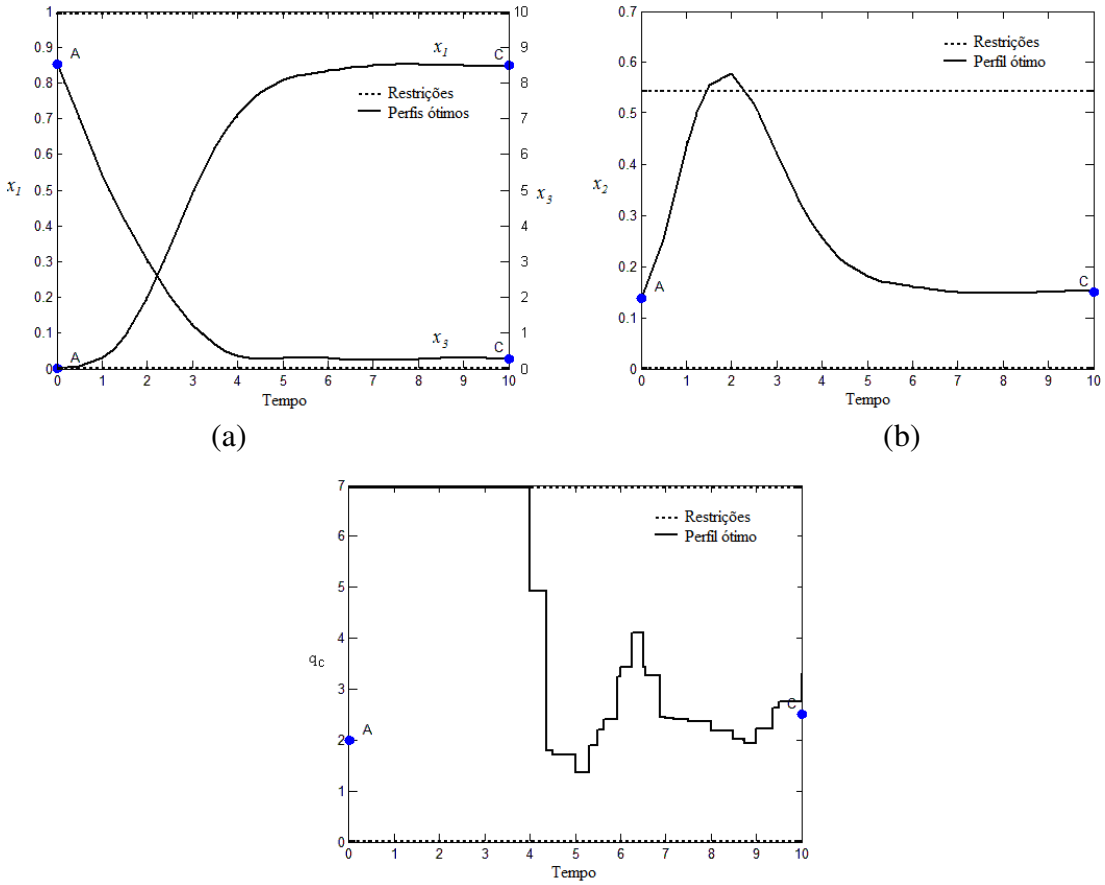
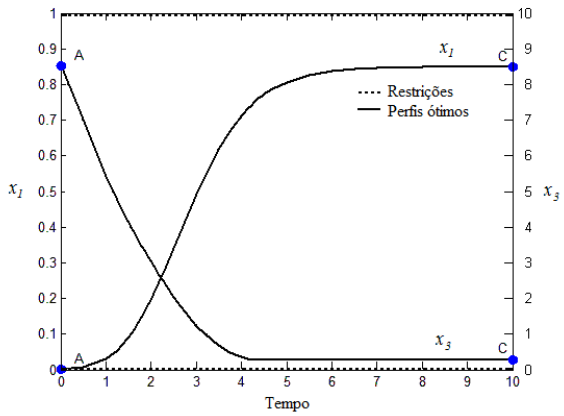


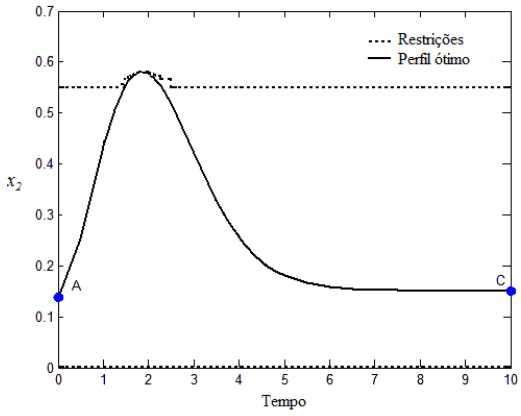
Figura 7.21 - Caso 3 - Solução do problema inviável – conflito de especificações.
(a) Perfis dos estados C_A e T, (b) C_B, (c) perfil de controle Q_C.

7.3.3.3 Problema Relaxado

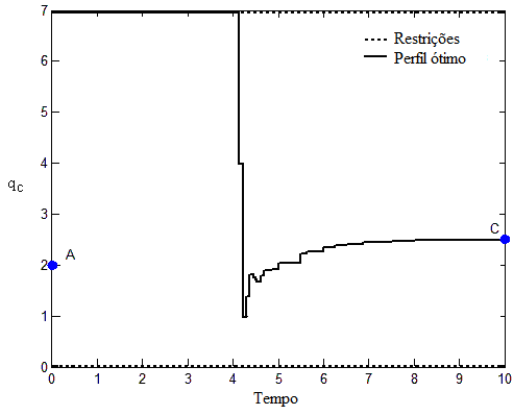
Neste caso se resolve o problema de otimização dinâmica multiobjetivos como apresentado anteriormente, tendo como objetivos que concorrem entre si: Minimizar o *ISE* (integral do erro ao quadrado) do tempo final e minimizar o relaxamento da restrição C_B. Os resultados são apresentados na Figura 7.22. Note que as restrições de Q_C e T não precisaram ser relaxadas. Isto se deve ao fato que o problema minimiza os movimentos das restrições para viabilizar a solução. Note que o relaxamento foi mínimo. C_B. Observe que os perfis dos estados e de controle foram próximos do problema original. Isto mostra que a aplicação do relaxamento recuperou o padrão original do problema viável.


Restrições
Perfis ótimos
x_1
x_3
A
C
Tempo

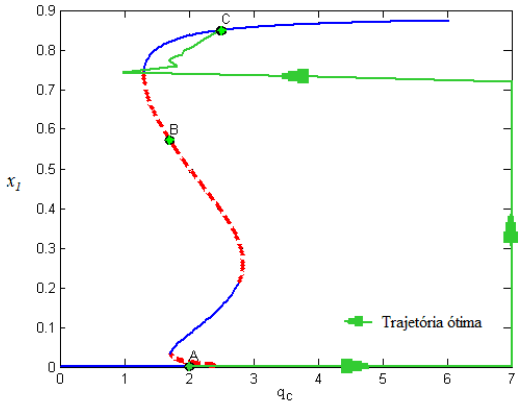
(a)


Restrições
Perfil ótimo
x_2
A
C
Tempo

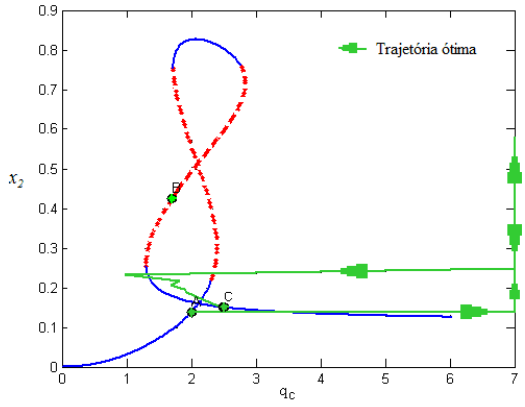
(b)


Restrições
Perfil ótimo
q_C
A
C
Tempo

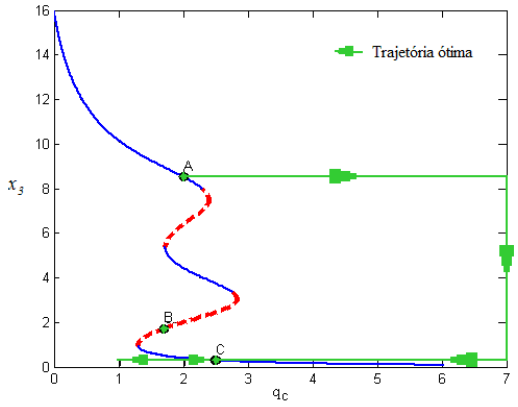
(c)


x_1
A
B
C
Trajetória ótima
q_C

(d)


x_2
A
B
C
Trajetória ótima
q_C

(e)



(f)

Figura 7.22 - Caso 3 - Solução do problema relaxado.
(a) Perfis dos estados C_A e T, (b) C_B, (c) perfil da variável de controle Q_C. (d, e, f) Diagramas de bifurcação das variáveis de estado para a transição A → C.

Os casos apresentados mostram que esta abordagem é eficiente para viabilizar problemas de controle ótimo quando os mesmos são estruturalmente inviáveis. Isto contribui para que as soluções de *DRTO* sejam mais robustas, contribuindo para sua efetividade. Podemos observar também pelos resultados que os relaxamentos propostos para viabilizar os problemas são efetuados com o mínimo relaxamento das restrições. Além disso, sugerimos que seja estabelecida alguma estratégia para relaxar algumas restrições e manter outras restrições inalteradas. Talvez a combinação de técnicas de isolamento de restrições inviáveis com a presente técnica proposta seja um possível caminho para a automatização deste procedimento.

8 Conclusões e Recomendações

Esta tese tem como pontos principais a proposta de um sistema integrado de *DRTO*, associado a uma ferramenta de diagnóstico e sintonia de otimização dinâmica. Juntamente com o sistema de *DRTO* e a ferramenta de diagnóstico, é proposto um novo método de solução de inviabilidades de *DAOP's*. A seguir são apresentadas as principais contribuições desta tese e algumas recomendações para trabalhos futuros.

8.1 Conclusões

A Parte 1 desta tese se concentrou na proposta de construção de um sistema de *DRTO* e de uma ferramenta de monitoração e diagnóstico.

No Capitulo 3 foram apresentadas as diferentes formas de representar o *DAOP*, onde se mostrou a importância de utilizar uma formulação genérica de *DAOP's* de simples e múltiplos estágios. Também foi chamada a atenção para a necessidade de um interpretador eficiente de *DAOP*. Em seguida, foram apresentados os métodos de solução de *DAOP's*, onde se evidenciou a importância de se dispor dos diferentes métodos, já que os métodos possuem aplicabilidades distintas. Além disso, foi chamada à atenção para a importância da utilização da técnica de análise da solução ótima, para refinar a solução do problema e avaliar as condições de otimalidade da solução. Após a apresentação dos métodos de solução, foi feita uma análise dos *softwares* existentes de otimização dinâmica. Constatou-se a inexistência de um sistema genuíno de *DRTO*, e que há somente aplicativos de otimização *offline*. Além disso, evidenciou-se que os pacotes têm funcionalidades importantes para um sistema de *DRTO*, mas não estão completos, ou seja, métodos importantes estão em pacotes diferentes (ex.: não tem métodos seqüenciais e simultâneos no mesmo pacote, não tendo a possibilidade de escolha).

Nesta revisão também se constatou que há iniciativas claras de *DRTO* voltados a *NMPC's*, mas a grande vantagem do *DRTO* é a capacidade de lidar com receitas, e isto não é explorado nas aplicações de *NMPC*. Por outro lado em *BMPC's*, tem-se encontrado diferentes soluções, tais como: otimização batelada-a-batelada, controle auto-otimizante, e algumas soluções customizadas. Pelo fato desta área ter recebido grande atenção recentemente, algumas iniciativas foram tomadas, como a proposta de mecanismo de disparo do otimizador. Porém, ainda não foi encontrada uma solução completa. Acredita-se que com o aprimoramento dos métodos de otimização, aumento de capacidade de processamento e memória das máquinas, e com a maior capacidade e eficiência do processamento paralelo, esta técnica deverá apresentar uma aceleração no seu amadurecimento, e rapidamente se popularizando industrialmente.

Além disso, também foram analisados métodos de refinamento da solução ótima, como a adaptação de malhas e detecção da estrutura da solução. Estas duas áreas têm uma grande importância, consomem muitos tempos computacionais e necessitam de aprimoramento. Talvez possam ser auxiliadas por análises *offline*, para minimizar o esforço computacional na aplicação *online*.

No Capitulo 4, são apresentadas propostas do sistema de *DRTO* e de ferramenta de monitoração e diagnóstico. Estas propostas agregam os melhores métodos dos *softwares* existentes e acrescentam outros (como a monitoração de eventos no gerenciamento de disparo do *DRTO* e nas facilidades de estimação de estados, parâmetros e reconciliação). O sistema de *DRTO* contempla tanto as aplicações *online* quanto as *offline* de uma forma integrada, de modo que uma determinada rodada do otimizador *online* possa ser reproduzida, visualizada ou analisada no modo *offline* sem qualquer esforço adicional de configuração ou alteração de arquivo. Esta facilidade não está presente nem nas aplicações de *RTO* existentes atualmente. A aplicação *offline*, idealizada desta forma, permite que seja realizada a visualização da solução *online* ou depurar um caso problemático, parando o algoritmo a cada iteração e inspecionando os perfis e as informações de diagnóstico geradas pelo otimizador. Esta também é uma funcionalidade inexistente nos *softwares* de otimização dinâmica. Esta aplicação também permite que se realizem estudos de casos, com os cenários importados da otimização *online* e com pequenas modificações, para explorar oportunidades ou resolver problemas. Estas características da aplicação proposta possibilitam mais flexibilidade e robustez ao *DRTO*, fornecendo uma grande capacidade de análise e solução de problemas.

Além dessas facilidades, a proposta de agregar facilidades, tais como: a possibilidade de utilização de diferentes métodos de solução de *DAOP* e de refinamento da solução (adaptação e detecção de estrutura), confere ao sistema maior capacidade de obtenção de solução ótima mais adequada do *DAOP*. Foi observada também neste trabalho a necessidade de se utilizar o processamento paralelo na solução de *DAOP's* para problemas grandes e complexos. Os principais pontos aqui recomendados são a paralelização das etapas de avaliações de funções (resíduos, Jacobianas, Hessianas), solução do *DAOP* (integração) e álgebra linear.

Ao propor uma linguagem de otimização orientada a objetos, permite-se que o engenheiro modele o problema de forma mais intuitiva. Outra característica diferenciadora é o fato do interpretador criar um *DAOP* padrão a partir do modelo escrito na forma que o usuário desejar, como também, a capacidade de reformular o problema incluindo comportamentos como controle de velocidade de atuação (Δu_{max}) e relaxamento de restrições. É importante destacar que são poucos os *softwares* que permitem formular e resolver o *DAOP* multi-estágios, como aqui proposto.

Das funcionalidades exclusivas da otimização *online*, pode-se destacar a proposta de formulação única para estimação de parâmetros, de estados e reconciliação de dados na forma de *DAOP*. Isto permite que este problema seja resolvido com os mesmos algoritmos de solução de otimização dinâmica. Além disso, a abordagem aqui proposta também permite que se faça opção por estimação e reconciliação simultânea ou seqüencialmente, assim como também, que sejam realizadas com freqüências diferentes ou até mesmo sob demanda.

Outra facilidade da aplicação *online* é o mecanismo de disparo do *DRTO*. O mecanismo aqui proposto inclui a monitoração de eventos e da conformidade da implementação da receita, além da avaliação da otimalidade e viabilidade já proposta anteriormente. Isto

fornece um diferencial que permite interromper a execução do otimizador em qualquer iteração quando ocorrer um evento que altere a estrutura do *DAOP* ou quando o operador não segue a última receita proposta, exigindo a re-otimização. Além disso, propõe-se aqui agregar ao critério de otimalidade já proposto, o critério de otimalidade de Hamiltoniano constante, para avaliar a fuga das condições de otimalidade.

Também se destaca aqui, a proposta de análise e implementação de resultados do otimizador. Nesta sistemática, é julgado se os resultados devem ser aplicados na planta ou não. Além disso, a proposta de forma de implementação executa a receita de forma suave e compensando os *biases* de reconciliação e conduzir o processo para as condições ótimas de forma suave, segura e para as posições corretas (já que as leituras têm erros sistemáticos). Nesta proposta procura-se evitar manipulações desnecessárias na planta, minimizando as intervenções nos controles.

A proposta de metodologia e ferramenta de monitoração e de diagnóstico e sintonia conferem mais confiabilidade, robustez e eficiência ao sistema de *DRTO*. Pelo fato da criação de uma sistemática consistente de monitoração e gerenciamento do sistema de *DRTO*, permite-se a percepção rápida dos resultados do otimizador, falhas ou até mesmo desempenhos ruins. E, com isso, remeter de forma pró-ativa para o diagnóstico do problema. A ferramenta de diagnóstico aqui proposto, permite três níveis de investigação de problemas. Esta ferramenta permite que se tenha uma visão geral do *DAOP* e da solução, bem como informações detalhadas. Neste nível de detalhes, entra-se no mérito de cada iteração da solução. No terceiro nível, realiza-se o diagnóstico de cada tarefa do otimizador em qualquer iteração do algoritmo. Esta estrutura permite que seja realizado um diagnóstico consistente e de forma hierarquizada. Pelo fato de ser uma nova ferramenta, ou seja, não há ferramentas similares, o sistema de *DRTO* proposto com esta ferramenta representa um diferencial importante, que confere as características acima citadas.

A Parte 2 desta tese se concentrou na proposta de metodologia de solução de inviabilidades em *DAOP's*.

As metodologias existentes estão direcionadas à solução de inviabilidades em *LP's* ou *NLP's*. A proposta aqui feita objetiva diagnosticar e resolver inviabilidades em *DAOP's* de forma automática. Esta metodologia se mostrou eficiente ao resolver diferentes tipos de inviabilidades. Esta metodologia é baseada na técnica de otimização multiobjetivos de programação por metas (*goal programming*). A ponderação das tolerâncias tem significado físico e portanto, tornando mais fácil a sua sintonia.

Esta metodologia pode ser aplicada tanto na otimização *online* quanto na *offline*. Claro que se deve avaliar previamente a necessidade de aplicar a técnica em determinadas variáveis, pois o procedimento aumenta a dimensão do problema e, conseqüentemente o custo computacional. Uma opção seria resolver inicialmente o *DAOP* original, e caso não obtenha solução bem sucedida, pode-se repetir automaticamente o procedimento, agora utilizando o problema reformulado (relaxado).

8.2 Recomendações

Baseado na experiência adquirida, nas limitações e problemas encontrados e no fato de terem sido desenvolvidos apenas alguns protótipos, sugere-se os seguintes tópicos a serem abordados em trabalhos futuros:

Construir o sistema definitivo, e utilizar em problemas industriais, podendo ser aplicado primeiramente a um problema de mistura em linha ou tanque de diesel ou gasolina da área de movimentação e mistura de uma refinaria. Este problema é de médio tamanho e complexidade. A segunda sugestão seria a solução de *DAOP* da torre desisobutanizadora da *RPBC* - Petrobras S.A., que é um problema bem maior e complexo e que já possui um modelo dinâmico desenvolvido no *EMSO* (Staudt, 2007). Estas implementações deverão consolidar as idéias aqui propostas.

Implementar o processamento paralelo na solução do otimizador dinâmico. Haja visto que a paralelização da execução de algumas funções do otimizador será inevitável, quando o DRTO for aplicado a problemas de grande porte e complexidade.

Construir um sistema definitivo de monitoração e diagnóstico e validar a proposta na solução industrial. Assim como no sistema de *DRTO*, este sistema deverá ser consolidado com aplicações industriais de porte.

Aperfeiçoar os critérios de adaptação de malhas e detecção de estruturas. Inclusive, de forma a definir o número máximo de adaptações e detecções de estruturas que mantenha a viabilidade técnica do seu uso (atende aos critérios de desempenho), bem como decidir quando refinar a solução baseado em resultados passados. O uso da técnica de *wavelets* se mostrou promissora, porém há a necessidade de aprimorar os critérios de tolerância e de parada, além de se explorar o uso de *wavelets* para adaptar malhas utilizando as informações das variáveis de estado.

Melhorar o algoritmo de cálculo da Hessiana e da integração do sistema adjunto no cálculo das sensibilidades. Estas duas tarefas são computacionalmente custosas. Já tem havido iniciativas de aprimoramento destas técnicas, mas ainda precisa ser consolidada industrialmente. Ao se computar exatamente a Hessiana, consome-se o tempo de sua avaliação, mas se ganha em número de iterações do algoritmo de otimização.

No ajuste de parâmetros, aperfeiçoar os critérios de escolha dos parâmetros a serem ajustados. Esta escolha deve ser baseada também nas dinâmicas das mudanças dos valores dos parâmetros e assim definir quando os parâmetros devem ser ajustados, e os dados reconciliados. Em função das variações dos parâmetros, definir quando o ajuste deve ser feito de forma simultânea ou sequencial, sendo que esta definição pode ser feita de forma automática. A escolha do conjunto de parâmetros a serem ajustados pode ser feita com a utilização do algoritmo *SELEST* (Secchi et al., 2006).

9 Referências bibliográficas

ABU-EL-ZEET, Z. H.; BECERRA, V. M.; ROBERTS, P. D. Combined bias and outlier identification in dynamic data reconciliation. *Computers & Chemical Engineering*. v.26, n.6, p.921-935, 2002.

ADAMS, J. New Tool for Monitoring and Sustaining RTO Applications: Aspen RTO Watch. *2003 Aspentech User Group Meetings*. Paris, França. 2003.

AGARWAL, M. Feasibility of on-line reoptimization in batch processes. *Chemical Engineering Communications*, 158. p.19-29, 1997.

ALLGOR, R. J., BARTON P. I. Mixed-integer dynamic optimization I: Problem formulation. *Computers and Chemical Engineering*, 23 (4-5), p.567-584, 1999.

ALMEIDA, E.; SECCHI, A.R. Dynamic Real-Time Optimization of a FCC Converter Unit, *In: ADCHEM 2006 - International Symposium on Advanced Control of Chemical Processes*, Gramado, Brasil, 2006a.

ALMEIDA, E.; SECCHI, A. R. Numerical Aspects for Solving a Dynamic Real-Time Optimization Problem of a FCC Converter Unit, *In: Workshop on Solving Industrial Control and Optimization Problems – SICOP*, Gramado, Brasil, 2006b.

ALMEIDA, E.; SECCHI, A. R. Dynamic optimization of a FCC converter unit: numerical analysis. *Brazilian Journal of Chemical Engineering*. 28(1). p.117-136, 2011.

ARELLANO-GARCIA, H.; BARZ, T.; WENDT, M.; WOZNY, G. Real-Time Feasibility of Nonlinear Predictive Control for Semi-batch Reactors. *European Symposium on Computer Aided Process Engineering- 15*. p.967-972, 2005.

ARIENS, D. *ACADO for Matlab User's Manual*. Optimization in Engineering Center (OPTEC) and Department of Electrical Engineering, Katholieke Universiteit Leuven, Belgium, 2010.

ARORA, N. *Nonlinear programming for data reconciliation and parameter estimation*. Tese de Doutorado. Carnegie Mellon University, 2003.

ASCHER, U. M.; SPITERI, R. J. *Collocation Software for Boundary Value Differential-Algebraic Equations*. Manual - University of British Columbia, Canada. 1995.

ASCHER, U. M.; PETZOLD, L. R. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM - Society for Industrial and Applied Mathematics Philadelphia, PA, USA. p. 247-253. 1998.

ASPENTECH, *Introduction to Aspen Custom Modeler*, 2002.

AZIZ, N.; MUJTABA, I. M. Optimal operation policies in batch reactors. *Chemical Engineering Journal*. 85. p.313–325, 2002.

BACKX, T.; BOSGRA, O.; MARQUARDT, W. Integration of Model Predictive Control and Optimization of Processes. *In: IFAC Symposium on Advanced Control of Chemical Processes*. v. 1., p.249-260, 2000.

BANSAL, V.; SAKIZLIS, V.; ROSS, R.; PERKINS, J. D.; PISTIKOPOULOS, E. N. New algorithms for mixed-integer dynamic optimization. *Computers and Chemical Engineering.* 27, p.647-668, 2003.

BARCLAY, A.; GILL, P. E.; ROSEN, J. B. SQP methods and their application to numerical optimal control. In W.H. Schmidt, K. Heier, L. Bittner, R. Bulirsch (eds.): *Variational Calculus, Optimal Control and Applications, International Series of Numerical Mathematics*, Birkhäuser, Basel. 124, p.207–222, 1998.

BARTON, P. I.; ALLGOR, R. J.; FEEHERY, W. F.; GALÁN, S. Dynamic optimization in a discontinuous world. *Industrial and Engineering Chemistry Research*, 37 (3), p.966–981. 1998.

BECERRA, V. M. *PSOPT Optimal Control Solver User Manual.* University of Reading, Reading, U.K., 2009.

BELLMAN, R., *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1957.

BETTS, J. T.; HUFFMAN, W. P. *Sparse Optimal Control Software-SOCS*, MEA-LR-085 Ed. Boeing Information and Support Services, Seattle, WA. 1997.

BETTS, J. T.; HUFFMAN, W. P. Mesh refinement in direct transcription methods for optimal control. *Optim. Control Appl. Meth.*, 19, p.1-21, 1998.

BETTS, J. T. *Practical methods for optimal control using nonlinear programming*. Society for Industrial and Applied Mathematics Philadelphia, PA, USA. P.81-132, 2001.

BIEGLER, L. T. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers and Chemical Engineering*, 8(3-4), p.243–248, 1984.

BIEGLER, L. T.; NOCEDAL, J.; Schmid, C. A reduced Hessian method for large-scale constrained optimization. *SIAM J. Optimization*. V. 5(2), p.314-347, 1995.

BIEGLER, L. T. Efficient solution of dynamic optimization and NMPC problems. In F. Allgoewer, A. Zheng (Eds.), *Nonlinear model predictive control*. Basel: Birkhaeuser. p.219–245, 2000.

BIEGLER, L. T.; CERVANTES, A. M.; WÄCHTER, A. Advances in simultaneous strategies for dynamic process optimization, *Chemical Engineering Science*. 57(4), p.575-593, 2002.

BIEGLER, L. T.; WÄCHTER, A. SQP SAND Strategies that Link to Existing Modeling Systems. *In: First CSRI Workshop on PDE-based Optimization, Lecture Notes in Computational Sciences and Engineering*, Ghattas, Heinkenschloss, Keyes, Biegler, and van Bloemen Waanders (eds.), Springer Verlag, Berlin, 2003.

BIEGLER, L. T.; GROSSMANN, I. E. Retrospective on optimization. *Computers and Chemical Engineering*. 28, p.1169–1192, 2004.

BIEGLER, L. T.; ZAVALA, V. M. Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization. *Computers and Chemical Engineering*. 33 (3), p.575-582, 2009.

BINDER, T.; VON WATZDORF, R.; MARQUARDT, W. Multiscale perspectives on process simulation and dynamic optimization *AspenWorld 1997*, Boston, MA. 1997.

BINDER, T.; CRUSE, A.; VILLAS, C.; MARQUARDT, W. Dynamic optimization using a wavelet based adaptive control vector parameterization strategy. *Computers and Chemical Engineering*. 24, p.1201–1207, 2000.

BINDER, T.; BLANK, L.; DAHMEN, W.; MARQUARDT, W. Iterative Algorithms for Multiscale State Estimation, Part 2: Numerical Investigations, *Journal of Optimization Theory and Applications*, Vol. III. (3), p.531–553, 2001.

BINDER, T. *Adaptive Multiscale Methods for the Solution of Dynamic Optimization Problems*. Tese de Doutorado. Lehrstuhl für Prozesstechnik, RWTH Aachen University. Aachen, 2002.

BLISS, G. A. *Lectures on the Calculus of Variations*. University of Chicago Press, Chicago. p.187-191. 1946.

BOCK, H. G.; PLITT, K. J. A multiple shooting algorithm for direct solution of optimal control problems. *International Federation of Automatic Control, 9th World Congress*, Budapest, 1984.

BOCK, H.G.; EICH, E.; SCHLODER, J. P. Numerical solution of constrained least squares boundary value problems in differential-algebraic equations. In: K. Strehmel (ed) *Numerical Treatment of Differential Equations*. Teubner, Leipzig 1988.

BONVIN, D. Optimal operation of batch reactors - a personal view. *Journal of Process Control*. 8 (5-6), p.355-368, 1998.

BONVIN, D.; SRINIVASAN, B. Optimal Operation of Batch Processes via the Tracking of Active Constraints. *ISA Transactions*, vol. 42(1), p.123-134, 2003.

BONVIN, D.; BODIZS, L.; SRINIVASAN, B. Optimal Grade Transition for Polyethylene Reactors via NCO Tracking. *Chemical Engineering Research and Design.* 7th World Congress of Chemical Engineering. Vol. 83(6), p.692-697, 2005.

BRENAN, K.E; PETZOLD, L. R. The Numerical Solution of Higher Index Differential/Algebraic Equations by Implicit Methods. *SIAM Journal on Numerical Analysis*, v. 26(4), p.976-996, 1989.

BRENDEL, M.; HARTWICH, A.; OLDENBURG, J.; SCHLEGEL, M.; STOCKMANN, K. *DyOS User's Guide - Version 2.1*. Lehrstuhl für Prozesstechnik, RWTH Aachen University. Aachen, 2008.

BROSILOW, C. B. The Structure and Design of Smith Predictors from the Viewpoint of Inferential Control. *In: Joint Automatic Control Conference*, June 17, Denver. 1979.

BRYSON, A. E.; HO, Y-C. *Applied Optimal Control – Optimization, Estimation and Control*. Hemisphere Publishing Company, Washington, DC, 1975.

BÜSKENS, C. Lösung optimaler Steuerprozesse. Lösung adjungierter Variablen. Automatische Gitterpunktsanpassung, NUDOCCCS. *Technical report, Westfälischen Wilhelms–Universität Münster*, 1996.

CAMACHO, E. F.; BORDONS, C. *Model Predictive Control*. Springer-Verlag, Berlin, p.196-199. 1999.

CERVANTES, A. M.; Biegler, L. T. Large-scale DAE optimization using simultaneous nonlinear programming formulations. *AIChE Journal*, 44, 1038-1050. 1998.

CERVANTES, A. M. *Stable Large-Scale DAE Optimization using Simultaneous Approaches*. Tese de doutorado, Carnegie Mellon University, Eng. Quimica, 2000.

CERVANTES, A..M.; WÄCHTER, A.; TUTUNCU, R.; BIEGLER, L. T. A reduced space interior point strategy for optimization of differential algebraic systems. *Computers and Chemical Engineering*. 24, p.39–51, 2000.

CHACHUAT, B.; SINGER, A. B.; BARTON, P. I. Global Mixed-Integer Dynamic Optimization. *AIChE Journal*, 51(8), p.2235–2253, 2005.

CHACHUAT, B.; SINGER, A. B.; BARTON, P. I. Global Methods for Dynamic Optimization and Mixed-Integer Dynamic Optimization. *Ind. Eng. Chem. Res.* 45, p.8373-8392, 2006.

CHACHUAT, B.; SRINIVASAN, B.; BONVIN, D. Model Parameterization Tailored to Real-time Optimization. *18th European Symposium on Computer Aided Process Engineering – ESCAPE 18*. Bertrand Braunschweig and Xavier Joulia (Editors), 2008a.

CHACHUAT, B.; MARCHETTI, A.; BONVIN, D. Process optimization via constraints adaptation. *Journal of Process Control*. 18, p. 244–257. 2008b.

CHACHUAT, B.; SRINIVASAN, B.; BONVIN, D. Adaptation strategies for real-time optimization. *Computers and Chemical Engineering*. 33, p.1557–1567. 2009.

CHINNECK, J. W. MINOS: Infeasibility analysis using MINOS. *Computers and Operations Research*. 21, p.1–9, 1994.

CHINNECK, J. W. Analyzing Mathematical Programs using MProbe. *Annals of Operations Research*. 104, p.33–48, 2001.

CHINNECK, J. W. The Constraint Consensus Method for Finding Approximately Feasible Points in Nonlinear Programs. *INFORMS Journal on Computing*. 16, 3, p.255-265, 2004.

CHINNECK, J. W. Feasibility and Infeasibility in Optimization: Algorithms and Computational Methods, Vol. 118, *International Series in Operations Research and Management Sciences*, Springer, p.97–129. 2008.

CHUI, C. K. An Introduction to Wavelets, em *Wavelet Analysis and Its Applications - Volume 1*. Academic Press, San Diego, USA, pg 1-20, 1992.

ČIŽNIAR, M.; FIKAR, M.; LATIFI, M. A. MATLAB Dynamic Optimisation Code DYNOPT, User's guide, *Technical Report, KIRP FCHPT STU*, Bratislava, 2006.

CLARKE, D.W.; MOHTADI, C.; TUFFS, P. S. Generalized Predictive Control. Part I: The Basic Algorithm, *Automatica*, v. 23, p.137-148, 1987a.

CLARKE, D. W.; MOHTADI, C.; TUFFS, P. S. Generalized Predictive Control. Part II: Extensions and Interpretations, *Automatica*, v. 23, p.149–160, 1987b.

COELLO, C. C. A.; PULIDO, G. T.; MONTES, E. M. Current and Future Research Trends in Evolutionary Multiobjective Optimization. *Information Processing with Evolutionary Algorithms*. Springer London, p.213-231. 2005.

COSTA, E. F.; VIEIRA, R. C.; SECCHI, A. R.; BISCAIA, E. C. Dynamic simulation of high-index models of batch distillation processes. *Lat. Am. Appl. Res.*, v.33(2), p.155-160, 2003.

CUTHRELL, J. E. *On the Optimization of Differential-Algebraic Systems of Equations in Chemical Enginnering*. Tese de Doutorado, Carnegie Mellon University, Pittsburgh. 1986.

CUTHRELL, J. E.; BIEGLER, L. T. On the optimization of differential-algebraic process systems. *AIChE Journal*, 33:p.1257–1270, 1987.

CUTHRELL, J. E.; BIEGLER, L. T. Simultaneous optimization and solution methods for batch reactor control profiles. *Computers and Chemical Engineering*, 13(1–2), p.49-62. 1989.

DAUBECHIES, I. *Ten Lectures on Wavelets*. SIAM, Philadelphia, Pennsylvania, PA, 1992.

CUTLER, C. R.; RAMAKER, B. L. Dynamic Matrix Control - A Computer Control Algorithm, *In: Proceedings of AIChE 86th National Meeting*, p. 01-23. 1979.

DE BOOR, C. *A Practical Guide to Splines*. In Applied Mathematical Sciences, Vol. 27, Springer-Verlag, New York, 2001.

DEB, K.; PRATAP, A.; MOITRA, S. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *Parallel Problem Solving from Nature - PPSN VI* p.849–858. 2000.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*. 6(2), p. 182-197. 2002.

DEUFLHARD, P.; HAIRER, E.; ZUGCK, J. One-step and extrapolation methods for differential-algebraic systems. *Numerische Mathematik*, (51). p.501–516. 1987.

DEUFLHARD, P.; NOWAK, U.; WULKOW, M. Recent developments in chemical computing, *Computers & Chemical Engineering*. 14 (11). p.1249–1258. 1990.

DIEHL, M.; D. B. LEINEWEBER, D. B.; Schäfer, A. *MUSCOD-II Users' Manual*. IWR-Preprint 2001-25. Universität Heidelberg, 2001.

DIEHL, M.; FINDEISEN, R.; ALLGÖWER, F.; BOCK, H. G.; SCHLÖDER, J. P. Nominal stability of the real-time iteration scheme for nonlinear model predictive control. *IEE Proc.-Control Theory Appl.*, 152(3), p.296-308, 2005.

DOLAN, E. D.; MORÉ, J. J. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2), p.201–213, 2002.

DRUD, A. CONOPT - A large-scale GRG-code. *INFORMS Journal on Computing* 6 (2), p.207-216. 1994.

DUFF, I. S.; REID, J. K. An Implementation of Tarjan's Algorithm for the Block Triangularization of a Matrix. *ACM Trans. Math. Software*. p.137-147. 1978.

DURAISKI R.G.; TRIERWEILER, J. O.; SECCHI, A. R. Nonlinear Model Predictive Control, Using Successive Linearization Approach, *Proceedings of the 3rd Mercosur Congress on Process Systems Engineering (ENPROMER 2001)*, Santa Fé, Argentina, vol. I, p.301-306, 2001.

DURAISKI, R. G.; TRIERWEILER, J. O.; SECCHI, A. R. A New Algorithm for Multi-objective Problems Based on the Goal Attainment Method. *EngOpt2008 - DTS Optimization*. 2008.

EDELBAUM, T. N., Theory of Maxima and Minima, in: *Optimization Techniques with Applications to Aerospace Systems*, edited by G. Leitmann, Academic Press, New York, p.1-32, 1962.

FEEHERY, W. F.; BARTON, P. I. Dynamic optimization with state variable path constraints, *Computers and Chemical Engineering*, 22 (9), p.1241–1256, 1998.

FEEHERY, W. F.; BARTON, P. I. Dynamic optimization with equality path constraints, *Ind. Eng. Chem. Res.* 38, p.2350–2363, 1999.

FIACCO, A. V. *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*. Academic Press, New York, 1983.

FISHER, M. E.; TEO, K. L.; GOH, C. J. *MISER3 - Optimal Control Software. Theory and User Manual - Version 3*. Department of Mathematics The University of Western Australia. Nedlands, WA 6907, Australia, 2004

FLETCHER, R.; LEYFFER, S. *User Manual for filterSQP*. University of Dundee, Scotland, UK. 1998.

FLETCHER, R.; LEYFFER, S.; TOINT, P. *A Brief History of Filter Methods*. Technical Report: ANL/MCS-P1372-0906. Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy. 2006.

FLORES-TLACUAHUAC A., BIEGLER L. T. Simultaneous mixed-integer dynamic optimization for integrated design and control. *Computers and Chemical Engineering*, 31 (5-6), p.588-600, 2007.

FORBES, J. F.; MARLIN, T. E. Design Cost: A Systematic Approach to Technology Selection for Model-Based Real-Time Optimization Systems, *Computers and Chemical Engineering*, 20(6-7), pp.717-734, 1996.

FRANÇOIS, G.; SRINIVASAN, B.; BONVIN, D. Run-to-Run Optimization of Batch Emulsion Polymerization. Em: *IFAC World Congress*, *15th Triennial World Congress*, Barcelona, Espanha. p.1258-1263, 2002.

FRANÇOIS, G.; SRINIVASAN, B.; BONVIN, D. Use of measurements for enforcing the necessary conditions of optimality in the presence of constraints and uncertainty. Journal of Process Control 15, p.701-712, 2005.

GAMS. *GAMS Examiner Manual*. GAMS Development Corp. Manual disponível no site: www.gams.com. 2011.

GARCIA, C. E.; MORARI, M. Internal Model Control –1. A Unifying Review and Some New Results. *Ind. Eng. Chem. Process Des. & Dev*. 21, p.308–323. 1982.

GATH, P. F. *CAMTOS - A Software Suite Combining Direct and Indirect Trajectory Optimization Methods*. Tese de Doutorado, Universität Stuttgart, Germany, 2002.

GESTHUISEN, R.; KLATT, K.-U.; ENGELL, S. Optimization-based state estimation – a comparative study for the batch polycondensation of PET. In: *ECC, 2001*. Porto. p.1062-1067. 2001.

GILL, P. E; JAY, L. O.; LEONARD, M. W.; PETZOLD, L. R.; SHARMA, V. An SQP method for the optimal control of large-scale dynamical systems. *Journal of Computational and Applied Mathematics*, v.120, p.197-213. 2000.

GILL, P. E.; MURRAY, W.; SAUNDERS, M. A. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Review*, 47(1), p. 99–131. 2005.

GOULD, N. I. M. Some reflections on the current state of active-set and interior point methods for constrained optimization', *SIAG/OPT Views-and-News* 14(1), p.2-7. 2003.

GOVATSMARK, M. S.; SKOGESTAD, S. Selection of controlled variables and robust setpoints. *Ind. Eng. Chem. Res*, 44 (7), p.2207-2217. 2005.

GREENBERG, H.J. ANALYZE: A computer-assisted analysis system for linear programming models. *Operations Research Letters*, 6(5), p.249-255. 1987.

GREENBERG, H. J. How to Analyze the Results of Linear Programs-Part 3: Infeasibility Diagnosis. *Interfaces*, 23 (6), p.120-139. 1993.

GREENBERG, H. J. The ANALYZE rulebase for supporting LP analysis. *Annals of Operations Research*. 65, p.91-126. 1996.

GROSDIDIER, P.; FORISY, B.; Hammann, M. The IDCOM-M controller, in T.J. McAvoy, Y. Arkun and E. Zafiriou (eds), *Proceedings of the 1988 IFAC Workshop on Model Based Process Control, Oxford*, p.31–36. 1988.

HELBIG, A.; ABEL, O.; MARQUARDT, W. Structural concepts for optimization based control of transient processes, em Nonlinear Model Predictive Control, Vol. 26, Allgöwer, F. and Zeng, A., Eds., *Progress in Systems and Control Theory*, Birkhäuser Verlag, Basel, 2000.

HERTZBERG, T.; ASBJORNSEN, O. A. Parameter estimation in nonlinear differential equations. *Computer Applications in the Analysis of Data and Plants*, Science Press, Princeton. 1977.

HINSBERGER, H. *Ein direktes Mehrschiessverfahren zur Lösung von Optimalsteuerungsproblemen – DIRMUS – Benutzeranleitung*. - TU Clausthal, 1996.

HILTMANN, P.; CHUDEJ, K.; BREITNER, M. H. *Eine modifizierte Mehrzielmethode zur Lösung von Mehrpunkt-Randwertproblemen*. Technical Report 14, Sonderforschungsbereich 255 der Deutschen Forschungsgemeinschaft: Transatmosphärische Flugsysteme, Lehrstuhl für Höhere und Numerische Mathematik, Technische Universität München, 1993.

HONEYWELL, Inc. *RMPCT Concepts Reference*, Manual de Produto da Honeywell. 1995.

HORN, J.; NAFPLIOTIS, N.; GOLDBERG, D. E. A niched pareto genetic algorithm for multiobjective optimization, in Proc. *1st IEEE Conf. Evolutionary Computation, IEEE World Congr. Computational Computation*, Piscataway, NJ, June 27–29, vol.1, p.82-87. 1994.

INCA Brochura - *INCA MPC4BATCH: Online Control and Optimization Solutions for the Batch Industry*. http://www.ipcos.com/en/batch_solutions. 2011.

ISIDORI, A. *Nonlinear Control Systems*. Springer-Verlag, 2 ed. 1989.

ISIDORI, A. H_∞ Control via Measurement Feedback for Affine Nonlinear Systems, *International of Robust and Nonlinear Control*, v. 4, p.553-574. 1994.

ISIDORI, A.; Kang, W. H_∞ Control via Measurement Feedback for General Nonlinear Systems, *IEEE Transactions on Automatic Control*, v. 40(3), p.466-472. 1995.

JACOBSON, D. H.; LELE, M. M. A Transformation Technique for Optimal Control Problems with a State Variable Inequality Constraint. *IEEE Trans. Autom. Control 1969, AC-14*, p.457-464. 1969.

KADAM, J. V.; SCHLEGEL, M.; MARQUARDT, W.; TOUSAIN, R. L.; V. HESSEM, D. H.; V.D. BERG, J.; BOSGRA, O. H. A Two–Level Strategy of Integrated Dynamic Optimization and Control of Industrial Processes – a Case Study. In: *ESCAPE–12 - European Symposium on Computer Aided Process Engineering – 12*, The Hague, The Netherlands, Elsevier, p.511–516. 2002.

KADAM, J. V.; MARQUARDT, W.; SCHLEGEL, M.; BOSGRA, O. H.; BACKX, T.; BROUWER, P.-J.; DÜNNEBIER, G.; VAN HESSEM, D.; TIAGOUNOV, A.; DE WOLF, S. Towards integrated dynamic real-time optimization and control of industrial processes. In: Proc. *FOCAPO 2003* (I. E. Grossmann and C. M. McDonald, Eds.). p.593-596. 2003.

KADAM, J. V.; MARQUARDT, W. Sensitivity-based solution updates in closed-loop dynamic optimization. In *Proceedings of the DYCOPS 7 conference*, July 2004.

KADAM, J. V.; MARQUARDT, W.; SRINIVASAN, B.; BONVIN, D. Dynamic Real-Time Optimization of Transitions in Industrial Polymerization Processes using Solution Models. *AIChE Annual Meeting*, Austin, TX. 2004.

KADAM, J. V.; MARQUARDT, W.; SRINIVASAN, B.; BONVIN, D. Optimal grade transition in industrial polymerization processes via NCO tracking. *AIChE Journal*, Vol. 53, No. 3. p.627-639. 2007.

KALMAN, R. E. Contributions to the Theory of Optimal Control, *Boletin de la Sociedad Matematica Mexicana*, Vol. 5, p.102–119. 1960.

KAMIEN, M. I.; SCHWARTZ, N. L. *Dynamic Optimization: The Calculus of Variations and Optimal Control in Economics and Management*. Elsevier ,North Holland. p. 227-229. 1991.

KAMESWARAN, S.; BIEGLER, L. T. Simultaneous Dynamic Optimization Strategies: Recent Advances and Challenges. *Computers and Chemical Engineering*, 30, p.1560-1575 2006.

KIRK, D. E. *Optimal Control Theory: An Introduction*. Dover Publications Inc, Mineola, New York. p. 29-95. 2004.

KONAK, A.; COIT, D. W.; SMITH, A. E. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering and System Safety*. 91, p.992–1007, 2006.

LANDLUST, J.; MESBAH, A.; WILDENBERG, J.; KALBASENKA, A. N.; KRAMER, H. J. M.; Ludlage, J. H. A. An industrial model predictive control architecture for batch crystallization, *In Proceedings of the 17th International Symposium on Industrial Crystallization* (The Netherlands), p.35-42, 2008.

LANG, Y.-D.; BIEGLER, L. T. A software environment for simultaneous dynamic optimization. *Computers and Chemical Engineering*. v.31, p.931-942, 2007.

LEE, J. H.; MORARI, M.; GARCIA, C. E. State-space interpretation of model predictive control, *Automatica*, v. 30(4), p.707-717, 1994.

LEE, J. H.; Xiao, J. Use of Two-Stage Optimization in Model Predictive Control of Stable and Integrating Systems, *Computers and Chemical Engineering*, 24, p.1591-1596, 2000.

LEE, J. H.; Cooley, B. Stable Min-Max Control of State-Space Systems with Bounded Input Matrix, *Automatica*, 36, p.463-473, 2000.

LEINEWEBER, D. B. *The theory of MUSCOD in a nutshell*. Dissertação de Mestrado, University of Heidelberg, 1996.

LEINEWEBER, D. B.; Bock, H. G.; Schlöder, J. P.; Gallitzendöfer, J. V.; Schäfer, A.; Jansohn, P. *A boundary value problem approach to the optimization of chemical processes described by DAE models*. IWR-Preprint 97-14, University of Heidelberg, 1997.

LI, R.; HENSON, M. A.; KURTZ, M. J. Selection of Model Parameters for Off-Line Parameter Estimation. *IEEE Transactions on Control Systems Technology*, 12: p.402-412. 2004.

LI, S.; LIM, K. Y.; FISHER, D. G. A state space formulation for model predictive control, *AIChE Journal*, v. 35(2), p.241-249. 1989.

LI, S.; PETZOLD, L. *Design of New DASPK for Sensitivity Analysis*, Technical Report, Dept. of Computer Science, Technical Report UCSB, 1999.

LI, S.; PETZOLD, L. Software and algorithms for sensitivity analysis of large-scale differential algebraic systems. *Journal of Computational and Applied Mathematics*. 125 p.131-145, 2000.

LI, S. PETZOLD, L.; ZHU, W. Sensitivity analysis of differential–algebraic equations: A comparison of methods on a special problem. *Applied Numerical Mathematics*. 32, p.161-174, 2000.

LI, S.; PETZOLD, L. *Description of DASPKADJOINT: An Adjoint Sensitivity Solver for Differential-Algebraic Equations*, UCSB Technical Report. 2002.

Li, W.C.; BIEGLER, L. T. Process control strategies for constrained nonlinear systems. *Ind. Eng. Chem. Res.*, v. 27(8), p.1421-1433, 1988.

LOCKE, M. H.; Westerberg, A. W.; EDAHL, R. H. Improved successive quadratic programming optimization algorithm for engineering design problems. *AIChE Journal*, v. 29, p.871-874, 1983.

LOEBLEIN, C.; PERKINS, I. D.; SRINIVASAN, B.; BONVIN, D. Performance Analysis of On-line Batch Optimization Systems. *Computers & Chemical Engineering*. v.21, Suppl., p. S867-S872, 1997.

LOGIST, .F; VAN ERDEGHEM, P. M. M.; VAN IMPE, J. F. Efficient deterministic multiple objective optimal control of (bio)chemical processes. *Chemical Engineering Science*, 64 (11), p. 2527-2538. 2009.

LOGSDON, J. S.; BIEGLER, L. T. Accurate solution of differential-algebraic optimization problems. *I&EC Research*, 28(11):p.1628–1639, 1989.

LONGHI, L. G. S. *Solução do Problema de Controle H_∞ Não-Linear*. Tese de Doutorado - COPPE/UFRJ, Engenharia Química, 2001.

MALY, T.; PETZOLD, L. R. Numerical methods and software for sensitivity analysis of differential-algebraic systems. *Applied Numerical Mathematics*, 20. p.57-79. 1996.

MARLER, R. T.; ARORA, J. S. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*. 26, p.369–395. 2004.

MATHWORKS, Inc. MATLAB Getting Started Guide. 2011.

MAYNE, D. Q.; MICHALSKA, H. Receding horizon control of nonlinear systems, *IEEE Transactions on Automatic Control*. v. 35, p.814–824. 1990.

MCCARL, B. A. *GAMSCHK User Documentation, Version 1.1*. Department of Agricultural Economics. Texas A&M University. 1998.

MEHRA, R. K.; ROUHANI, R.; ETERNO, J.; RICHALET, J.; RAULT, A. Model Algorithmic Control: Review and Recent Developments, *In: Eng. Foundation Conference on Chemical Process Control II*, Sea Island, GA, D.E. Seborg and T.F. Edgar (eds.) p.287. 1982.

MESBAH, A.; LANDLUST, J.; VERSTEEG, C.; HUESMAN, A. E. M.; KRAMER, H. J. M.; LUDLAGE, J. H. A.; VAN DEN HOF, P. M. J. Model-based Optimal Control of Industrial Batch Crystallizers. *20th European Symposium on Computer Aided Process Engineering – ESCAPE20*. S. Pierucci and G. Buzzi Ferraris (Editors), 2010.

MILETIC, I. P.; MARLIN, T. E. Results analysis for real-time optimization (RTO): Deciding when to change the plant operation. *Computers & Chemical Engineering*. Vol. 20, Suppl., p. S1077-SI082. 1996.

MILETIC, I. P.; MARLIN, T. E. Results Diagnosis for Real-Time Process Operations Optimization. *Computers & Chemical Engineering*. Vol. 22, Suppl., p. S475-S482, 1998.

MISITI, M.; MISITI, Y.; OPPENHEIM, G.; POGGI, J-M. *Wavelets and their applications*. ISTE Ltd, London, UK, 2007.

MODELICA ASSOCIATION. *Modelica - A Unified Object-Oriented Language for Physical Systems Modeling Language Specification Version 3.2*; 2010.

MORSHEDI, A. M.; CUTLER, C. R.; SKROVANEK, T. A. Optimal solution of Dynamic Matrix Control with Linear Programming Techniques (LMDC), *In: Proceedings of the American Control Conference*. 1985.

MUJTABA, I. M. *Batch Distillation: Design and Operation*. Series on Chemical Engineering. Vol. 3, Imperial College Press. Covent Garden, London, p.116-152. 2004.

MURTHY, B. S. N.; GANGIAH, K.; HUSAIN, A. Performance of various methods in computing optimal policies. *The Chemical Engineering Journal*. 19, p.201-208. 1980.

NEMHAUSER, G. L.; RINNOOY KAN, A. H. G.; TODD, M. J. Handbooks in Operations Management Science Vol. 1 Optimization (Elsevier Science Publishers B.V., Amsterdam, p. 672. 1989.

NOCEDAL, J.; WRIGHT, S. *Numerical Optimization*. Springer, 2nd edition New York, NY, USA. pp. 435-441 e 529-562. 2006.

OBERLE, H. J.; GRIMM, W. *BNDSCO - A program for the numerical solution of optimal control problems*. Technical report DLR IB 515-89-22, Institute for Flight Systems Dynamics, DLR, Oberpfaffenhofen, Germany, 1989.

PARK, T.; BARTON, P. I. Analysis and Control of Combined Discrete/Continuous Systems: Progress and Challenges in the Chemical Process Industries, *AIChE Symposium Series*, vol. 93, no. 316, p.102-114, 1997.

PLITT, K. J. *Ein superlinear konvergentes Mehrziel verfahren zur direkten Berechnung beschränkter optimaler Steuerungen*. Diplomarbeit, Universität Bonn, 1988.

POLLARD, G. P.; SARGENT, R. W. H. Off Line Computation of Optimum Controls for a Plate Distillation Column. *Automatica*, v. 6(6), p.59-76, 1970.

PONTRYAGIN, L. S.; BOLTYANSKII, V. G.; GAMKRELIDZE, R. V.;MISHCHENKO, E. F. *The Mathematical Theory of Optimal Processes*. Interscience Publishers, John Wiley & Sons Inc, New York, 1962.

PRETT, D. M.; Gillete, R. D. Optimization and Constrained Multivariable Control of a Catalytic Cracking Unit, *In: Proceedings ot the Joint Automatic Control Conference*. 1980.

PSE - Process Systems Enterprise, *gPROMS Introductory User Guide*, 2004.

PYTLAK, R.; VINTER, R. B. *A Feasible Directions Type Algorithm for Optimal Control Problems with State and Control Constraints: Convergence Analysis*. Technical report, Imperial College of Science, Technology and Medicine, London, England. 1996a.

PYTLAK, R.; Vinter, R. B. *A Feasible Directions Type Algorithm for Optimal Control Problems with State and Control Constraints: Implementation*. Technical report, Imperial College of Science, Technology and Medicine, London, England (1996b)

QIN, S. J.; BADGWELL, T. A. A survey of industrial model predictive control technology. *Control Engineering Practice*. 11, p.733-764. 2003.

RAO, C. V.; RAWLINGS, J. B.; LEE, J. H. Constrained linear state estimation – a moving horizon approach. *Automatica*, 37(10), p.1619-1628, 2001.

RAO, C. V.; RAWLINGS, J. B. Constrained process monitoring: Moving-horizon approach. *AIChE Journal*. v.48, n.1, p.97-109, 2002.

RAO, C. V.; RAWLINGS, J. B.; MAYNE, D. Q. Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations. *IEEE Transactions on Automatic Control*. v.48, n.2, p.246-258, 2003.

RAO, A. V.; BENSON. D.; HUNTINGTON, G. T.; FRANCOLIN, C.; DARBY, C. L.; PATTERSON, M.; SANDERS, I. User's *Manual for GPOPS Version 3.0: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using Pseudospectral Methods*. University of Florida, Gainesville, FL, 2010.

RAY, W. H. *Advanced Process Control*, McGraw-Hill, New York. 1981.

RENFRO, J. G.; MORSHEDI, A. M.; ASBJORNSEN, O. A. Simultaneous optimization and solution of systems described by differential algebraic equations, *Computer and Chemical Engineering*, 11 (5), p.503-517. 1987.

RICHALET, J.; RAULT, A.; TESTUD, J. L. Model Predictive Heuristic Control: Applications to Industrial Processes, *Automatica* , v. 14 p.413-428. 1978.

ROBINSON, E. R. The optimisation of batch distillation operations. *Chemical Engineering Science*, Vol. 24, p.1661-1668. 1969.

ROBINSON, E. R. The optimal control of an industrial batch distillation column. *Chemical Engineering Science*, Vol. 25, p.921-928. 1970.

ROELOFS, M.; BISSCHOP, J. *AIMMS - The User's Guide*. Paragon Decision Technology, 2009.

ROTAVA, O. *Implementation of Linear and Nonlinear Optimal Control Techniques in a Carbon Dioxide Absorption Desorption Plant*. Tese de Doutorado, Imperial College Science, London, 1997.

RUPPEN, D.; BENTHACK, C.; BONVIN, D. Optimization of batch reactor operation under parametric uncertainty - computational aspects. *Journal of Process Control*, 5(4), p.235–240, 1995.

RUTQUIST, P. E.; EDVALL, M. M. *PROPT - Matlab Optimal Control Software*. TOMLAB Optimization, 2010.

SALAU, N. P. G. *Abordagem Sistemática para Construção e Sintonia de Estimadores de Estados Não-Lineares*. Tese de Doutorado do *DEQUI/UFRGS*, 2009.

SALAU, N. P. G.; TRIERWEILER, J. O.; SECCHI, A. R. CEKF&S: Uma alternativa eficiente para a estimação de estados. *XVIII Congresso Brasileiro de Automática*, Bonito-MS, p.2655-2660. 2010.

SANTOS, L.O.; AFONSO, P.; CASTRO, J.; DE OLIVEIRA, N. M. C.; BIEGLER, L. T. On-line implementation of nonlinear MPC: Na experimental case study. *In: ADCHEM 2000 – International Symposium on Advanced Control of Chemical Processes*, V.2, pp. 731-736, Pisa, 2000.

SARGENT, R. W. H; SULLIVAN, G. R. The Development of an Efficient Optimal Control Package. *In: Proc. of the 8th IFIP Conf. on Optimisation Techniques Part 2*. 1977.

SCHAFFER, J. D. *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms*. Tese de doutorado, Vanderbilt University/Electrical Engineering. Nashville, Tennessee. 1984.

SCHLEGEL, M.; MARQUARDT, W. Direct sequential dynamic optimization with automatic switching structure detection. In: *Proc. of DYCOPS 7*, Cambridge, USA (S.L. Shah and J.F. MacGregor, Eds.). Omnipress. 2004.

SCHLEGEL, M.; MARQUARDT, W.; EHRIG, R.; NOWAK, U. Sensitivity analysis of linearly-implicit differential–algebraic systems by one-step extrapolation. *Applied Numerical Mathematics*. 48, p.83-102. 2004.

SCHLEGEL, M.; STOCKMANN, K.; BINDER, T.; MARQUARDT, W. Dynamic optimization using adaptive control vector parameterization. *Computers and Chemical Engineering*. 29. p.1731-1751. 2005.

SCHLEGEL, M. *Adaptive discretization methods for the efficient solution of dynamic optimization problems.* Fortschritt-Berichte VDI, Reihe 3, Nr. 829, VDI-Verlag, Düsseldorf, Tese de Doutorado do AVT/RWTH. 2005.

SCHLEGEL, M.; MARQUARDT, W. Adaptive Switching Structure Detection for the Solution of Dynamic Optimization Problems. *Ind. Eng. Chem. Res.*, 45, p.8083-8094. 2006a.

SCHLEGEL, M.; MARQUARDT, W. Detection and exploitation of the control switching structure in the solution of dynamic optimization problems. *Journal of Process Control*. 16, p.275–290. 2006b.

SCHULZ, V. H. *Reduced SQP Methods for Large Scale Optimal Control Problems in DAE with Application to Path Planning Problems for Satellite Mounted Robots*. Tese de Doutorado. Universitat Heidelberg. 1996.

SCHWARTZ, A. L. *Theory and Implementation of Numerical Methods Based on Runge-Kutta Integration for Solving Optimal Control Problems*. Tese de Doutorado, Department of Electrical Engineering and Computer Science, University of California-Berkeley. 1996.

SCHWARTZ, A.; POLAK, E.; CHEN, Y. *RIOTS_95: A Matlab Toolbox for Solving Optimal Control Problems*.Version 1.0. University of California, Berkeley, CA, 1997.

SECCHI, A. R.; CARDOZO, N. S. M.; ALMEIDA NETO, E.; FINKLER, T. F. An algorithm for automatic selection and estimation of model parameters. In: *International Symposium on Advanced Control of Chemical Processes - ADCHEM, 2006*. Gramado. 2006.

SERBAN, R. *COOPT - Control and Optimization of Dynamic Systems: Users' Guide*, Technical Report UCSB-ME-99-1, University of California, Santa Barbara, 2002.

SHEIKHZADEH, M.; TRIFKOVIC, M.; ROHANI, S. Real-time optimal control of an anti-solvent isothermal semi-batch crystallization process. *Chemical Engineering Science*. 63. p.829 – 839. 2008.

SKOGESTAD, S. Self-optimizing control: the missing link between steady-state optimization and control. *Computers and Chemical Engineering*. 24, p.569-575. 2000.

SIMSCI, *ROMeo Real Time System User's Guide*. Invensys Inc. - SimSci-Esscor. 2002.

SOARES, R. P.; SECCHI, A. R. EMSO: A new Environment for Modelling, Simulation and Optimisation, *13th European Symposium on Computer Aided Process Engineering (ESCAPE-13).* Lappeenranta, Finland, 2003.

SOARES, R. P. Desenvolvimento de um simulador genérico de processos dinâmicos. Dissertação de mestrado - *DEQUI/UFRGS*, Porto Alegre, Brasil. 2003.

SOARES, R. P. EMSO Manual, 2007

SOARES, R. P. Depuração para Simuladores de Processos Baseados em Equações. Tese de doutorado - *DEQUI/UFRGS*, Porto Alegre, Brasil. 2007.

SOARES, R. P.; SECCHI, A. R. Debugging Static and Dynamic Rigorous Models for Equation-oriented CAPE Tools. *IFAC2007 - 8th International IFAC Symposium on Dynamics and Control of Process Systems*, Cancún, Mexico. Preprints Vol.2, June 6-8, 2007.

SOARES, R. P.; SECCHI, A. R. Structural Analysis for Static and Dynamic Models. Preprint submitted to *Mathematical and Computer Modelling*. 2010

SRINIVAS, N.; DEB, K. Multi-objective Optimization Using Nondominated Sorting in Genetic Algorithms. *J. Evolutionary Computation*, 2(3), pp. 221-248. 1994.

SRINIVASAN, B.; MYSZKOROWSKI, P.; BONVIN, D. A Multi-Criteria Approach to Dynamic Optimization. In: *American control conference*. Seattle, WA. p.1767-1771. 1995.

SRINIVASAN, B.; PRIMUS, C. J.; BONVIN, D.; RICKER, N. L. Run-to-run optimization via control of generalized constraints. *Control Engineering Practice*. 9, p.911–919. 2001.

SRINIVASAN, B.; BONVIN, D.; VISSER, E.; PALANKI, S. Dynamic optimization of batch processes II. Role of measurements in handling uncertainty. *Computers and Chemical Engineering*. 27, p.27-44. 2002.

SRINIVASAN, B.; PALANKI, S.; BONVIN, D. Dynamic optimization of batch processes I. Characterization of the nominal solution. *Computers and Chemical Engineering*. 27, p.1-26. 2003.

SRINIVASAN, B.; BONVIN, D. Dynamic Optimization under Uncertainty via NCO Tracking: A Solution Model Approach. In: *BatchPro Symposium*, vol. Plenary talk, p.17-35, 2004.

SRINIVASAN, B.; BONVIN, D. Real-Time Optimization of Batch Processes by Tracking the Necessary Conditions of Optimality. *Ind. Eng. Chem. Res.* 46, p.492-504, 2007.

STAUDT, P. B. *Modelagem e Simulação Dinâmica de Colunas de Destilação*. Dissertação de mestrado - DEQUI/UFRGS, Porto Alegre, Brasil. 2007.

STRYK, O.V.; BULIRSCH, R. Direct and Indirect Methods for Trajectory Optimization. *Annals of Operations Research*, 37, pp. 357-373, 1992.

STRYK, O.V. *User's Guide for DIRCOL Version2.1 – A Direct Collocation Method for the Numerical Solution of Optimal Control Problems*. Technische Universität Darmstadt. 1999.

SUSSMANN, H. J.; WILLEMS, J. C. 300 Years of Optimal Control: From the Brachystochrone to the Maximum Principle. *IEEE Control Systems*.17(3). p32-44. 1997.

TAMIZ, M.; MARDLE, S. J.; JONES, D. F. Detecting IIS in infeasible linear programmes using techniques from goal programming, *Computers and Operations Research*, 23. p.113-119. 1996.

TANARTKIT, P.; BIEGLER, L. T. Stable decomposition for dynamic optimization. *Ind. Eng. Chem. Res.*, 34. p.1253-1266, 1995.

TANARTKIT, P.; BIEGLER, L. T. A nested, simultaneous approach for dynamic optimization problems - I. *Computers and Chemical Engineering*. 20(6/7), p.735-741. 1996.

TANARTKIT, P.; BIEGLER, L.T. A nested simultaneous approach for dynamic optimization problems II: the outer problem, *Computers and Chemical Engineering*. 21(12). p.1365-1388. 1997.

TANARTKIT, P. *Stable computational procedures for dynamic optimization in process engineering*. Tese de doutorado, Carnegie Mellon University, Eng. Quimica, 1996.

TAYNTOR, C. B. *Six Sigma Software Development*. 2nd Edition, Auerbach Publications - Taylor & Francis Group, Boca Raton, New York. 2007.

TERWIESCH, P.; AGARWAL, M.; RIPPIN, D. W. T. Batch Unit Optimization with Imperfect Modelling: A Survey. Journal of Process Control, 4(4), p.238-258. 1994.

TILLER, M. M. Introduction to Physical Modeling with Modelica. Kluwer Academic Publishers. Boston, MA. 2001.

TLACUAHUAC, A. F-; MORENO, S. T.; BIEGLER, L. T. Global Optimization of Highly Nonlinear Dynamic Systems. *Ind. Eng. Chem. Res.*, 47, p.2643-2655. 2008.

VANDECRAEN, B.; ESPINOSA, J.; PLUYMERS, B.; VINSON, D.R.; LUDLAGE, J.; VAN BREMPT, W. An Industrial Approach for Efficient Modeling and Advanced Control of Chemical Batch Processes. *8th International IFAC Symposium on Dynamics and Control of Process Systems*, Cancún, Mexico. Preprints Vol.2, June 6-8, 2007.

VASANTHARAJAN, S.; BIEGLER, L. T. Large-scale decomposition for successive quadratic programming, *Computers and Chemical Engineering*, 12, p.1087–1101. 1988.

VASANTHARAJAN, S.; BIEGLER, L. T. Simultaneous strategies for optimization of differential-algebraic systems with enforcement of error criteria, *Computers and Chemical Engineering*, 14, p.1083–1100. 1990.

VASSILIADIS, V. S. *DAEOPT – a differential-algebraic optimization solver*. Technical report, Centre for Process Systems Engineering, Imperial College, London, 1992.

VASSILIADIS, V. S. *Computational Solution of Dynamic Optimization Problems with General Differential-Algebraic Constraints*, Tese de Doutorado, University of London, London, U.K., 1993.

VASSILIADIS, V. S.; SARGENT, R. W. H.; PANTELIDES, C. C. Solution of a class of multistage dynamic optimization problems. 1. Problems without path constraints, *Ind. Eng. Chem. Res.*, 33(9), p.2111-2122. 1994a.

VASSILIADIS, V.S.; SARGENT, R. W. H.; PANTELIDES, C. C. Solution of a class of multistage dynamic optimization problems. 2. Problems with path constraints, *Ind. Eng. Chem. Res.*, 33(9), p.2123-2133. 1994b.

VAN DER SCHAFT, A. J. L_2-Gain Analysis of Nonlinear Systems and Nonlinear State Feedback H_∞ Control, *IEEE Transactions on Automatic Control*, v. 37(6), p.770-784, 1992.

VAN LOON, J. Irreducibly inconsistent systems of linear inequalities. *European Journal of Operational Research*, 8, p.283–288. 1981.

VON STRYK, O. *User's guide for DIRCOL - a direct collocation method for the numerical solution of optimal control problems*. Ver. 2.1, Technical University of Munich, Germany. 1999.

WÄCHTER, A. *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. Tese de Doutorado, Carnegie Mellon University, Pittsburgh, PA, USA, 2002.

WÄCHTER, A.; BIEGLER, L. T. *Line Search Filter Methods for Nonlinear Programming: Local Convergence*. IBM Research Report - RC23033 (W0312-090). 2003.

WÄCHTER, A.; BIEGLER, L. T. On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming*, 106(1), p.25-57. 2006.

WAJGE, R. M.; GUPTA, S. K. Multiobjective dynamic optimization of a nonvaporizing nylon 6 batch reactor. *Polymer Engineering & Science*, 34: p.1161–1172. 1994.

WÜRTH, L.; HANNEMANN, R.; MARQUARDT, W. Neighboring-extremal updates for nonlinear model-predictive control and dynamic real-time optimization. *Journal of Process Control* 19. p.1277–1288. 2009.

YANG, J. Infeasibility resolution based on goal programming. *Computers & Operations Research*, 35 (5), p.1483-1493. 2008.

ZANIN, A. C. Implementação Industrial de um Otimizador em Tempo Real. *D.Sc. Tese, EP-USP, São Paulo*, 2001.

ZAVALA, V. M. *Computational Strategies for the Optimal Operation of Large-Scale Chemical Processes*. Tese de doutorado, Carnegie Mellon University, Eng. Quimica, 2008.

ZAVALA, V. M.; LAIRD, C. D.; BIEGLER, L. T. A fast moving horizon estimation algorithm based on nonlinear programming sensitivity. *Journal of Process Control*. 18. p.876–884. 2008.

ZAVALA, V. M.; BIEGLER, L. T. The advanced-step NMPC controller: Optimality, stability and robustness. *Automatica* 45. p.86-93. 2009.

ZYNGIER, D. *Monitoring, Diagnosing and Enhancing the Performance of Linear Closed-Loop Real-Time Optimization Systems*. Tese de doutorado, Chemical Engineering - McMaster University. Hamilton, Ontario, Canadá. 2006.

ZITZLER, E.; THIELE, L. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. *Computer Engineering and Communication Networks Lab (TIK)*. Zurich, Switzerland. TIK-Report No 43. 1998.

ZITZLER, E.; THIELE, L. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*. 3(4). p.257-271. 1999.

Apêndice A - Wavelets

A seguir são apresentados alguns conceitos relacionados à análise multi-escalas e da metodologia de adaptação de malhas de discretização dos controles.

O processo de adaptação de malhas começa com uma representação dos perfis de controle na forma de simples escala (ex.: *B-spline* – de Boor, 1978), sendo que uma função genérica *f(t)* pode ser escrita como um *B-spline* de ordem *m* na forma:

$$f(t) = \sum_{j=1}^{n} c_j \varphi_j^{(m)}(t)$$

onde *c* é o vetor de parâmetros do *spline* e φ é a função base. Para uma um perfil constantes por partes, a ordem da aproximação *m = 1*, pode ser definido como:

$$\varphi_j^{(1)}(t) = \Gamma(t) = \begin{cases} 1 & se \quad t_j \le t < t_{j+1} \\ 0 & caso\ contrário \end{cases}$$

Representando o mesmo perfil na forma de uma função discreta de simples escala, pode-se escrever:

$$f_J = \sum_{k \in I_J} c_{J,k} \varphi_{J,k} = c_{I_J}^T \phi_{I_J} \qquad \text{(A.1)}$$

onde $\varphi_{J,k}$ é a função base correspondente à resolução da escala J e $I_J = \{0; 1; \ldots; 2^J - 1\}$ é o conjunto de índices da escala J, considerando que o k indica o índice de translação em uma escala específica.

Uma vez que a função seja representada na forma de simples escala, a etapa seguinte é a representação da função na forma de multi-escalas. Uma opção interessante é a utilização de transformadas *wavelets*. Esta técnica divide o sinal em componentes de diferentes freqüências locais (Daubechies, 1992). São funções de análise multi-escalas (análise multi-resolução), onde a idéia básica do método é representar uma função como um conjunto de coeficientes que tem informações locais das freqüências da função. Neste método, a função é decomposta em uma seqüência de espaços S aninhados, onde $S_{j0} \subset \ldots \subset S_j \subset \ldots$, sendo *j* uma escala específica, que corresponde ao nível de resolução do sinal e j_0 a escala de discretização mais grosseira. A representação multi-escalas da função *f* pode ser escrita na forma:

$$f_{J+1} = \sum_{k \in I_{j0}} c_{j_0,k} \varphi_{j_0,k} + \sum_{l=j_0}^{J-1} \sum_{k \in I_l} d_{l,k} \psi_{l,k} \qquad \text{(A.2)}$$

onde $c_{j_0,k}$ é o coeficiente da representação em simples escala da malha mais grosseira e $\psi_{l,k}$ e a função base *wavelet*. Este conjunto completo $\{ \varphi_{j_0,k} : k \in I_{j0} \} \cup \{ \psi_{l,k} : \in I_l ; l = j_0, \ldots, 2^J\text{-}1 \}$ é chamada de base Riesz. Esta formulação pode ser vista como uma seqüência de

multi-resolução onde se adiciona detalhes à função mais grosseira f_j com um complemento w_j. Desta forma, pode-se escrever:

$$f_{j+1} = f_j + w_j$$

Ou na forma da decomposição de espaços S aninhados, temos:

$$S_{j+1} = S_j \oplus W_j = S_{j_0} \oplus W_{j_0} \oplus W_{j_0+1} \oplus \ldots \oplus W_j \qquad \text{ou} \qquad S_J = S_{j_0} \oplus_{k=j_0}^{J-1} W_k$$

onde J é o nível de escala desejado. A representação dos detalhes w_j da base *wavelet* $\{\psi_{j,k} : \in I_j\}$ pode ser escrita na forma:

$$w_j = \sum_{k \in I_j} d_{j,k} \psi_{j,k}$$

Uma representação eficiente destes espaços aninhados e do nível de detalhes do sinal (S) pode ser visto na Figura A.1, onde os sinais em simples escala e multi-escalas são representados no lado esquerdo da figura e os coeficientes *wavelets* são representados no lado direito da figura, sendo que a sua magnitude é apresentada com o seu valor numérico ou pela coloração da área do respectivo coeficiente.

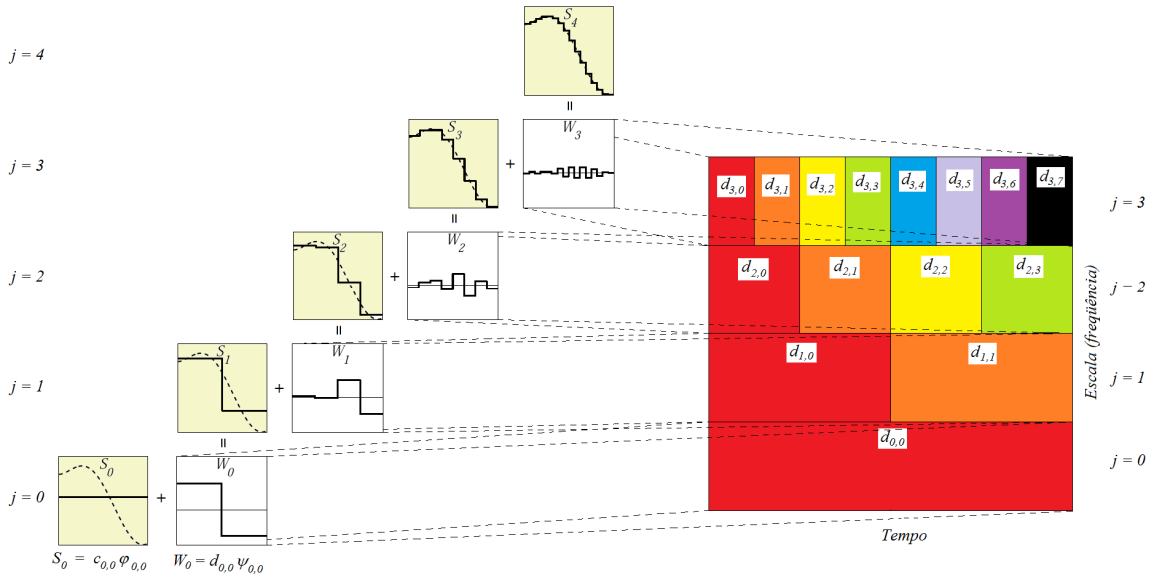


Figura A.1 – Representação do sinal em simples escala (S_j) e *wavelets* (W_j).

Na análise multi-resolução (Chui, 1992), é possível fazer uma transformação rápida de *wavelet* (*FWT – Fast Wavelet Transformation*), onde se converte a representação simples escala em multi-escalas e vice-versa. Para fazer esta transformação, considere que $d_{j_0-1,k} = c_{j_0,k}$ e $\psi_{j_0-1,k} = \varphi_{j_0,k}$. A decomposição de f para o nível J pode ser escrito como:

$$f_J = \sum_{l=j_0-1}^{J-1} \sum_{k \in I_l} d_{l,k} \psi_{l,k} = d_{\Lambda_J}^T \Psi_{\Lambda_J} \qquad \Lambda_J = \{(l,k) \mid j_0 - 1 \le l < J-1; k \in I_J\} \quad \text{(A.3)}$$

Desta forma, igualando a equação A.1 com A.3, pode-se escrever a mudança de base na seguinte forma:

$$f_J = c_{I_J}^T \phi_{I_J} = d_{\Lambda_J}^T \Psi_{\Lambda_J}$$

Pode-se imaginar a utilização de um operador T_J de transformação da base de simples escala em *wavelets* e vice-versa. Desta forma, pode-se dizer que $d_{\Lambda_J} = T_J c_{I_J}$ ou $c_{I_J} = T_J^{-1} d_{\Lambda_J}$ representam a decomposição em *wavelet* pela transformação $T_J : c \rightarrow d$ ou reconstrução do sinal original pela a transformação inversa $T_J^{-1} : d \rightarrow c$ (Binder, 2002).

As funções *wavelets* $\psi_{j,k}$ são aquelas derivadas de um função base (*wavelet* mãe) através de suas operações de escalonamento e deslocamento sobre a função base. A operação de escalonamento equivale à dilatação ou compressão da onda e o deslocamento significa transladar a onda. Quanto mais dilatada a onda, maior é o fator de escala (menor freqüência). De modo geral, pode-se escrever:

$$\psi_{a,b} = \frac{1}{\sqrt{a}} \psi_{0,0}\left(\frac{t-b}{a}\right)$$

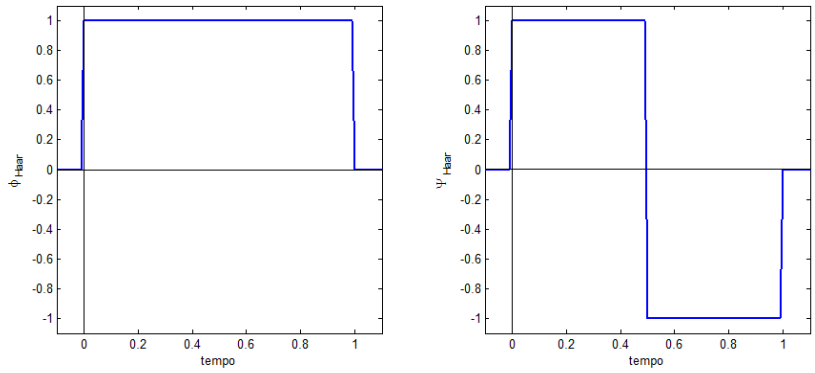
onde a é um fator de escala, b um fator de translação ou deslocamento, $\psi_{0,0}$ uma função base (*wavelet* mãe) (Misiti et al., 1997) e $1/\sqrt{a}$ um fator de normalização da energia do sinal (para ter a mesma energia em todas as escalas).

Existem dois tipos de transformações *wavelet*, a contínua (*CWT – Continuous Wavelet Transformation*) e discreta (*DWT – Discrete Wavelet Transform*). A DWT é mais interessante de ser utilizado, pois pelo fato de produzir menos informações redundantes do que o CWT. Além disso, é conveniente utilizar o DWT, pois os perfis de controle, na solução de problemas de controle ótimo, têm características discretas devido à sua solução pelos métodos diretos. A decomposição do sinal discreto (divisão) ocorre somente com alguns escalonamentos e deslocamentos (Daubechies, 1992). No escalonamento, tem-se uma dilatação binária de $a = 2^j$, e na translação diádica de $b = ka$ (relativo ao grupo de dois – binários). Desta forma, a função *wavelet* pode ser escrever:

$$\psi_{j,k}(t) = 2^{j/2} \psi_{0,0}\left(2^j t - k\right) \qquad k \in \{0,...,2^j\text{-}1\}$$

Uma função de base *wavelet* tem sido usada por Binder (2002) e Schlegel (2004) na adaptação de malhas em problemas de otimização dinâmica, sendo que a base Haar (que corresponde a um perfil constante por partes) tem sido utilizada na adaptação de malhas na solução de problemas de controle ótimo. A aproximação constante por partes é equivalente a *B-spline* $\varphi_j^{(1)}(t)$ no intervalo [0, 1[. Essa função *wavelet* mãe por ser escrita na forma:

$$\phi(t) = \Gamma(t) = \begin{cases} 1 & se \quad 0 \le t < 1 \\ \\ 0 & caso\ contrário \end{cases} \qquad \psi(t) = \Gamma(t) = \begin{cases} 1 & se \quad 0 \le t < 1/2 \\ -1 & se \quad 1/2 \le t < 1 \\ 0 & caso\ contrário \end{cases}$$



Os coeficientes do filtro passa-baixa da base Haar são $h(n) = \left\{ \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right\}$.

Apêndice B - Algoritmo IPOPT

A estrutura do algoritmo *IPOPT* é apresentada a seguir. Esta é uma reprodução do algoritmo apresentado por Wächter e Biegler (2005), de uma forma simplificada.

O *IPOPT* é um método de barreira *primal-dual* que resolve uma seqüência de problemas de barreira (Nocedal e Wright, 2006). A direção de busca pode ser calculada utilizando-se duas abordagens básicas a de espaço completo (*full space*) e a de espaço reduzido (*reduced space*) (Nocedal e Wright, 2006). A convergência global é alcançada usando abordagens de busca em linha (*linesearch*) com diferentes funções de mérito (Lagrangeano aumentado ou função penalidade exata) ou um filtro *linesearch* proposto por Fletcher e Leyffer (1998) (Fletcher et. al., 2006). Este algoritmo resolve eficientemente problemas de grande escala (*número de estados* > 10^6), com grande número de graus de liberdade (*G.L.* > 10^4) e com boa flexibilidade do algoritmo. O algoritmo tem duas etapas básicas, uma etapa de definição da direção de busca e outra de *linesearch*. Na etapa de cálculo da direção de busca, é necessário utilizar a Hessiana do problema de otimização. A Hessiana pode ser cheia ou reduzida e ainda aproximada por técnicas quasi-Newton como *QN-BFGS* e *QN-SR1* (Nocedal e Wright, 2006) ou por diferenças finitas. Também se pode fazer uso do método *PCG* (*Preconditioned Conjugate Gradient*) (Nocedal e Wright, 2006).

De forma muito simplificada, o problema de otimização não-linear pode ser descrito na seguinte forma. Considere o método de barreira *primal-dual* para resolver problemas *NLP* na forma:

$$\begin{aligned} \min_{x\in\Re^n} \quad & f(x) \\ s.a. \quad & c(x) = 0 \\ & x_L \le x \le x_U \end{aligned}$$

Abordagem de Barreira Primal-Dual

Para facilitar o entendimento do algoritmo, o problema *NLP* pode ter sua notação simplificada na forma:

$$\begin{aligned} \min_{x\in\Re^n} \quad & f(x) \\ s.a. \quad & c(x) = 0 \\ & x \ge 0 \end{aligned}$$

Para resolver o problema usando método de barreira, pode-se escrever:

$$\begin{aligned} \min_{x\in\Re^n} \quad & \varphi_\mu(x) = f(x) - \mu\sum_{i=1}^{n} ln\left(x^{(i)}\right) \\ s.a. \quad & c(x) = 0 \end{aligned}$$

Com o parâmetro de barreira $\mu > 0$ e $x^{(i)}$ significa o *i-ésimo* componente do vetor *x*.

As condições de otimalidade podem ser escritas como:

$$\nabla f(x) + \nabla c(x)\lambda - z = 0$$
$$c(x) = 0$$
$$XZe - \mu e = 0$$

Onde $\boldsymbol{\lambda}$ são os multiplicadores de Lagrange das restrições de igualdade, $\boldsymbol{z}$ os multiplicadores das restrições de fronteira, sendo *e* um vetor de dimensão apropriada. As matrizes $\boldsymbol{X}$ e $\boldsymbol{Z}$ são diagonais dos elementos dos vetores $\boldsymbol{x}$ e $\boldsymbol{z}$.

Note que as condições de *KKT* do problema original são obtidas quando $\mu = 0$ e $x, z \geq 0$.

O método calcula uma solução aproximada do problema de barreira para um valor fixo de μ, então reduz o parâmetro de barreira e continua a resolver o problema de barreira a partir da solução prévia.

Usando as partes individuais das condições de otimalidade, define-se o erro de otimalidade do problema de barreira como:

$$E_\mu(x,\lambda,z) = max\left\{\frac{\left\|\nabla f(x_k) + \nabla c(x_k)\lambda - z\right\|_\infty}{s_d}, \left\|c(x_k)\right\|_\infty, \frac{\left\|XZe - \mu e\right\|_\infty}{s_c}\right\}$$

onde $s_d, s_c \geq 1$ são os fatores de escala definidos como:

$$s_d = max\left\{s_{max}, \frac{\|\lambda\|_1 + \|z\|_1}{(n+m)}\right\} \Big/ s_{max} \qquad \text{e} \qquad s_c = max\left\{s_{max}, \frac{\|z\|_1}{n}\right\} \Big/ s_{max}$$

onde s_{max} é o parâmetro de escalonamento máximo.

Uma vez definida a direção de busca, faz-se uma busca em linha para encontrar o avanço adequado das variáveis do problema. Necessita-se escolher um passo $\alpha_k \in (0,1]$ para obter as próximas posições das variáveis:

$$x_{k+1} = x_k + \alpha_k^x d_k^x$$
$$\lambda_{k+1} = \lambda_k + \alpha_k^\lambda d_k^\lambda$$
$$z_{k+1} = z_k + \alpha_k^z d_k^z$$

Para se obter esta nova posição, utiliza-se o método do filtro de busca em linha, que estabelece limites para α de forma que as garante a positividade das restrições implícitas $x_{k+1}, \lambda_{k+1}, z_{k+1} > 0$.

A figura B.1 mostra este esquema de forma simplificada.

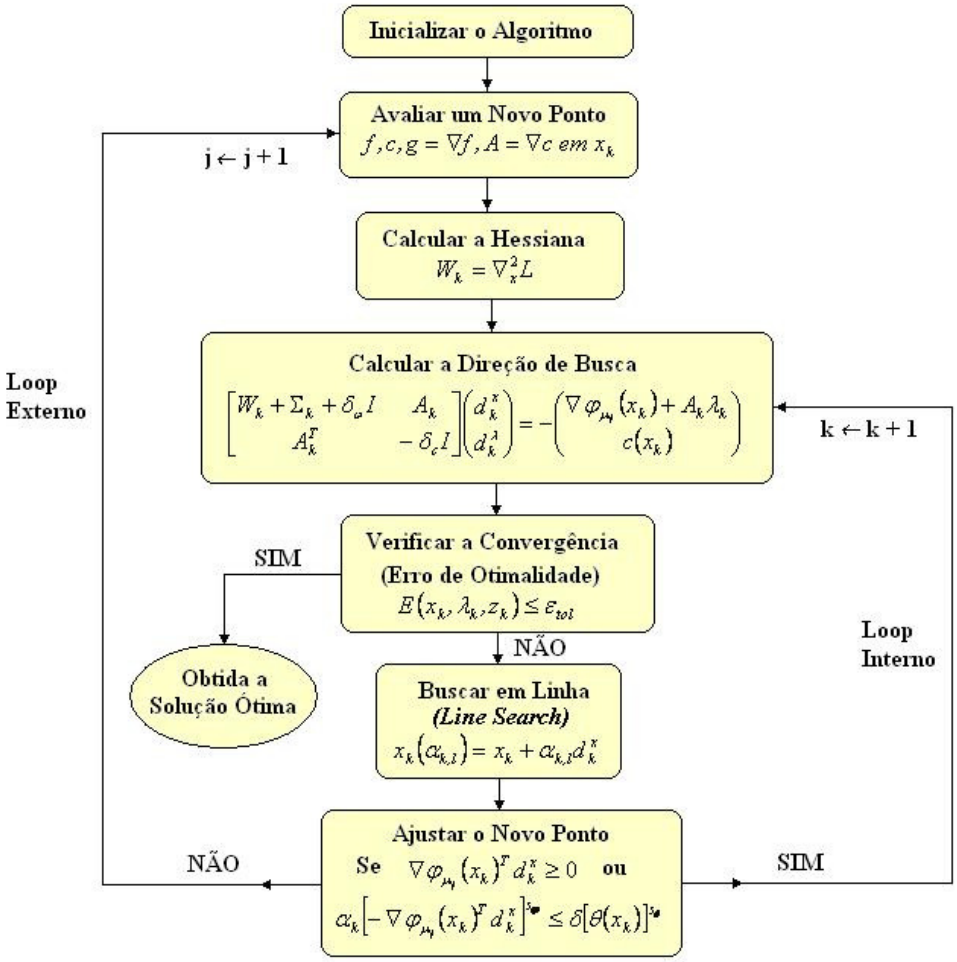


Figura B.1 – Esquema simplificado do algoritmo *IPOPT*

Passos do algoritmo IPOPT

Os passos do algoritmo estão descritos abaixo.

Dados:

- Ponto de partida (x_0, λ_0, z_0) com $x_0, z_0 > 0$;
- Parâmetro inicial de barreira $\mu_0 > 0$;
- Constantes $\varepsilon_{tol} > 0$; $s_{max} \ge 1$; $K_\varepsilon > 0$; $K_\mu \in (0, 1)$; $\theta_\mu \in (1, 2)$; $\tau_{min} \in (0, 1)$; $K_\Sigma > 1$; $\theta_{max} \in (\theta(x_0), \infty]$; $\theta_{max} > 0$; $\gamma_\theta, \gamma_\varphi \in (0, 1)$; $\delta > 0$; $\gamma_\alpha \in (0, 1]$; $s_\theta > 1$; $s_\varphi \ge 1$; $\eta_\varphi \in (0, 1/2)$; $K_{SOC} \in (0, 1)$; $p^{max} \in \{0, 1, 2, \ldots\}$.

1. Inicializar os contadores de iterações $j \leftarrow 0$, $k \leftarrow 0$, e o filtro $F_0 = \left\{(\theta, \varphi) \in \Re^2 : \theta \ge \theta^{max}\right\}$. E obter τ_0 de $\tau_j = max\{\tau_{min}, 1 - \mu_j\}$ onde $\tau_{min} \in (0, 1)$;

2. Verificar a convergência global do problema *NLP* original. Se $E_0(x_k, \lambda_k, z_k) \le \varepsilon_{tol}$ (com erro estimado E_0 definido abaixo), então PARAR [CONVERGIU].

$$E_\mu(x,\lambda,z) = max\left\{\frac{\left\|\nabla f(x_k)+\nabla c(x_k)\lambda - z\right\|_\infty}{s_d}, \left\|c(x_k)\right\|_\infty, \frac{\left\|XZe-\mu e\right\|_\infty}{s_c}\right\}$$

onde $s_d,\ s_c \geq 1$ são os fatores de escala definidos como

$$s_d = max\left\{s_{max}, \frac{\left\|\lambda\right\|_1 + \left\|z\right\|_1}{(n+m)}\right\}\Big/ s_{max} \qquad \text{e} \qquad s_c = max\left\{s_{max}, \frac{\left\|z\right\|_1}{n}\right\}\Big/ s_{max}$$

3. Verificar a convergência do problema de barreira. Se $E_{\mu_j}(x_k,\lambda_k,z_k) \leq K_\varepsilon \mu_j$, então

   3.1. Calcular μ_{j+1} e τ_{j+1} e faça $j \leftarrow j+1$;

   $\mu_{j+1} = max\left\{\frac{\varepsilon_{tol}}{10}, min\left\{K_\mu \mu_j, \mu_j^{\theta_\mu}\right\}\right\}$ e $\tau_{j+1} = max\left\{\tau_{min}, 1-\mu_j\right\}$ onde $\tau_{min} \in (0, 1)$

   3.2. Reinicializar o filtro $F_k = \left\{(\theta,\varphi) \in \Re^2 : \theta \geq \theta^{max}\right\}$;

   3.3. Se $k = 0$ repetir o passo 3, caso contrário continuar com o passo 4.

4. Calcular a direção de busca. Calcular $\left(d_k^x, d_k^\lambda, d_k^z\right)$ de

$$\begin{bmatrix} W_k + \Sigma_k + \delta_\omega I & A_k \\ A_k^T & -\delta_c I \end{bmatrix}\begin{pmatrix} d_k^x \\ d_k^\lambda \end{pmatrix} = -\begin{pmatrix} \nabla\varphi_{\mu_j}(x_k) + A_k\lambda_k \\ c(x_k) \end{pmatrix}$$

para $\delta_\omega,\ \delta_c \geq 0$. A escolha de δ_ω e δ_c a cada iteração é feita através do mecanismo de correção de inércia (Wächter e Biegler, 2005).

5. *Linesearch*

   5.1. Inicializar o *linesearch*. Fazer $\alpha_{k,0} = \alpha_k^{max}$ e $l \leftarrow 0$, com
   $\alpha_k^{max} = max\left\{\alpha \in (0,1] : x_k + \alpha d_k^x \geq (1-\tau_j)x_k\right\}$

   5.2. Calcular o novo ponto de tentativa. Fazer $x_k(\alpha_{k,l}) = x_k + \alpha_{k,l} d_k^x$.

   5.3. Verificar a aceitabilidade do filtro. Se $\left[\theta(x_k(\alpha_{k,l})), \varphi_{\mu_j}(x_k(\alpha_{k,l}))\right] \in F_k$, rejeitar o passo tentativa e passar para o passo 5.5.

   5.4. Verificar se o decréscimo em relação à iteração atual foi suficiente.

   - Caso I: $\theta(x_k) \leq \theta^{min}$, $\nabla\varphi_{\mu_j}(x_k)^T d_k^x < 0$ e $\alpha_{k,l}\left[-\nabla\varphi_{\mu_j}(x_k)^T d_k^x\right]^{s_\varphi} > \delta[\theta(x_k)]^{s_\theta}$: Se $\varphi_{\mu_j}(x_k(\alpha_{k,l})) \leq \varphi_{\mu_j}(x_k) + \eta_\varphi \alpha_{k,l} \nabla\varphi_{\mu_j}(x_k)^T d_k^x$, então aceitar a tentativa, fazer $x_{k+1} = x_k(\alpha_{k,l})$ e passar para o passo 6. Caso contrário, continuar no passo 5.5.
   - Caso II: $\theta(x_k) > \theta^{min}$ ou $\nabla\varphi_{\mu_j}(x_k)^T d_k^x \geq 0$ ou $\alpha_{k,l}\left[-\nabla\varphi_{\mu_j}(x_k)^T d_k^x\right]^{s_\varphi} \leq \delta[\theta(x_k)]^{s_\theta}$: Se $\theta(x_k(\alpha_{k,l})) \leq (1-\gamma_\theta)\theta(x_k)$ ou $\varphi_{\mu_j}(x_k(\alpha_{k,l})) \leq \varphi_{\mu_j}(x_k) + \gamma_\varphi \theta(x_k)$, então aceitar a tentativa, fazer $x_{k+1} = x_k(\alpha_{k,l})$ e passar para o passo 6. Caso contrário, continuar no passo 5.5.

   5.5. Inicializar a correção de segunda ordem (*SOC – Second-Order Correction*). Se $l > 0$ ou $\theta(x_k(\alpha_{k,0})) < \theta(x_k)$, sair da *SOC* e passar ao passo 5.10. Caso contrário,

inicializar o contador do *SOC*, $p \leftarrow 1$, e calcular $c_k^{SOC} = \alpha_{k,0} c(x_k) + c\left(x_k + \alpha_{k,0} d_k^x\right)$. Inicializar $\theta_{old}^{SOC} \leftarrow \theta(x_k)$.

5.6. Calcular a correção de segunda ordem. Calcular $d_k^{x,cor}$, d_k^{λ}, α_k^{SOC} e x_k^{SOC} com

$$\begin{bmatrix} W_k + \Sigma_k + \delta_\omega I & A_k \\ A_k^T & -\delta_c I \end{bmatrix} \begin{pmatrix} d_k^{x,cor} \\ d_k^{\lambda} \end{pmatrix} = -\begin{pmatrix} \nabla \varphi_{\mu_j}(x_k) + A_k \lambda_k \\ c\left(x_k^{SOC}\right) \end{pmatrix}$$

$$\alpha_k^{SOC} = max\left\{\alpha \in (0,1] : x_k + \alpha d_k^{x,cor} \geq (1-\tau_j) x_k\right\}$$

$$x_k^{SOC} = x_k + \alpha_k^{SOC} d_k^{x,cor}$$

5.7. Verificar a aceitação do filtro (em *SOC*). Se $\left[\theta\left(x_k^{SOC}\right), \varphi_{\mu_j}\left(x_k^{SOC}\right)\right] \in F_k$, rejeitar o passo tentativa e passar para o passo 5.10.

5.8. Verificar se o decréscimo em relação a iteração atual foi suficiente (em *SOC*).

- Caso I: $\theta(x_k) \leq \theta^{min}$, $\nabla \varphi_{\mu_j}(x_k)^T d_k^x < 0$ e $\alpha_{k,0}\left[-\nabla \varphi_{\mu_j}(x_k)^T d_k^x\right]^{s_\varphi} > \delta[\theta(x_k)]^{s_\theta}$: Se $\varphi_{\mu_j}\left(x_k^{SOC}\right) \leq \varphi_{\mu_j}(x_k) + \eta_\varphi \alpha_{k,0} \nabla \varphi_{\mu_j}(x_k)^T d_k^x$, então aceitar a tentativa, fazer $x_{k+1} = x_k(\alpha_{k,l})$ e passar para o passo 6. Caso contrário, passar para o passo 5.9.
- Caso II: $\theta(x_k) > \theta^{min}$ ou $\nabla \varphi_{\mu_j}(x_k)^T d_k^x \geq 0$ ou $\alpha_{k,0}\left[-\nabla \varphi_{\mu_j}(x_k)^T d_k^x\right]^{s_\varphi} \leq \delta[\theta(x_k)]^{s_\theta}$: Se $\theta\left(x_k^{SOC}\right) \leq (1-\gamma_\theta)\theta(x_k)$ ou $\varphi_{\mu_j}\left(x_k^{SOC}\right) \leq \varphi_{\mu_j}(x_k) + \gamma_\varphi \theta(x_k)$, então aceitar a tentativa, fazer $x_{k+1} = x_k^{SOC}$ e passar para o passo 6. Caso contrário, passar para o passo 5.9.

5.9. Próxima correção de segunda ordem. Se $p = p^{max}$ ou $\theta\left(x_k^{SOC}\right) > K_{SOC}\theta_k^{SOC}$, abortar *SOC* e passar para o passo 5.10. Caso contrário, acrescentar o contador do *SOC* $p \leftarrow p+1$, e fazer $c_k^{SOC} \leftarrow \alpha_k^{SOC} c_k^{SOC} + c\left(x_k^{SOC}\right)$ e $\theta\left(x_{old}^{SOC}\right) \leftarrow \theta_k^{SOC}$. Voltar para o passo 5.6.

5.10. Escolher o novo passo tentativa. Fazer $\alpha_{k,l+1} = \frac{1}{2}\alpha_{k,l}$ e $l \leftarrow l+1$. Se o passo for muito pequeno, isto é, $\alpha_{k,l} = \alpha_k^{min}$ então fazer $\alpha_{k,l} = \alpha_k^{min}$, passar para a fase de restauração da viabilidade no passo 9. Caso contrário, retornar para o passo 5.2. Sendo:

$$\alpha_k^{min} = \gamma_\alpha \cdot \begin{cases} min\left\{\gamma_\theta, \dfrac{\gamma_\varphi \theta(x_k)}{-\nabla \varphi_{\mu_j}(x_k)^T d_k^x}, \dfrac{\delta[\theta(x_k)]^{s_\theta}}{\left[-\nabla \varphi_{\mu_j}(x_k)^T d_k^x\right]^{s_\varphi}}\right\} & se \quad \nabla \varphi_{\mu_j}(x_k)^T d_k^x < 0 \; e \; \theta(x_k) \leq \theta^{min} \\ min\left\{\gamma_\theta, \dfrac{\gamma_\varphi \theta(x_k)}{-\nabla \varphi_{\mu_j}(x_k)^T d_k^x}\right\} & se \quad \nabla \varphi_{\mu_j}(x_k)^T d_k^x < 0 \; e \; \theta(x_k) > \theta^{min} \\ \gamma_\theta \; caso \; contrario \end{cases}$$

Com o fator de segurança $\gamma_\alpha \in (0, 1]$.

6. Aceitar o ponto tentativa. Fazer $\alpha_k = \alpha_{k,l}$ (ou $\alpha_k = \alpha_k^{SOC}$ se o ponto *SOC* foi aceito em 5.8, e atualizar o multiplicador λ_{k+1} e z_{k+1}.

$$\lambda_{k+1} = \lambda_k + \alpha_k d_k^{\lambda}$$

$$z_{k+1} = z_k + \alpha_k^z d_k^z \qquad \text{sendo} \quad \alpha_k^z = max\left\{\alpha \in (0,1] : z_k + \alpha d_k^z \geq (1-\tau_j) z_k\right\}$$

Aplicar uma correção em z_{k+1} se necessário, na forma

$$z_{k+1}^{(i)} \leftarrow max\left\{ min\left\{ z_{k+1}^{(i)}, \frac{K_\Sigma \mu_j}{x_{k+1}^{(i)}} \right\}, \frac{\mu_j}{K_\Sigma x_{k+1}^{(i)}} \right\}, \qquad i = 1,..., n$$

Para $K_\Sigma \geq 1$ a cada passo.

7. Aumentar o filtro se necessário. Se $\nabla \varphi_{\mu_j}(x_k)^T d_k^x \geq 0$ ou $\alpha_k \left[-\nabla \varphi_{\mu_j}(x_k)^T d_k^x \right]^{s_\varphi} \leq \delta [\theta(x_k)]^{s_\theta}$, aumentar o filtro usando

$$F_{k+1} = F_k \cup \left\{ (\theta, \varphi) \in \Re^2 : \theta \geq (1-\gamma_\theta)\theta(x_k) \quad e \quad \varphi \geq \varphi_{\mu_j}(x_k) - \gamma_\varphi \theta(x_k) \right\}$$

Caso contrário, manter o filtro e fazer $F_{k+1} = F_k$.

8. Continuar com a próxima iteração. Calcular o valor da função objetivo e acrescentar o contador de iterações $k \leftarrow k+1$ e retornar para o passo 2.

9. Fase de restauração da viabilidade. Aumentar o filtro usando

$$F_{k+1} = F_k \cup \left\{ (\theta, \varphi) \in \Re^2 : \theta \geq (1-\gamma_\theta)\theta(x_k) \quad e \quad \varphi \geq \varphi_{\mu_j}(x_k) - \gamma_\varphi \theta(x_k) \right\}$$

E calcular uma nova iteração $x_{k+1} > 0$ reduzindo a medida da inviabilidade $\theta(x)$, tal que x_{k+1} seja aceitável para o filtro aumentado, isto é, $\left(\theta(x_{k+1}), \varphi_{\mu_j}(x_{k+1})\right) \notin F_{k+1}$. Então continuar com a iteração no passo 8.

A ferramenta de diagnóstico e sintonia deve ter a capacidade de avaliar detalhadamente e identificar os fatores de insucesso de um algoritmo de otimização. A monitoração da convergência do algoritmo é feita observando-se a cada iteração os seguintes pontos:

- Erro de otimalidade (item 2 do algoritmo)
  - Viabilidade primal
  - Viabilidade dual
  - Complementaridade
- Parâmetro de barreira (item 3 do algoritmo)
- Violação da restrição (item 9 do algoritmo)
- Direção do passo (item 4 do algoritmo)
- Tamanhos dos passos para as variáveis primais e duais (item 5 do algoritmo)
- Número de tentativas de tamanhos de passos no *linesearch* (item 5 do algoritmo)
- Valor da função objetivo (item 8 do algoritmo)

Apêndice C - Estrutura do DRTO

Em função do descrito no capitulo 4 desta tese, são apresentadas, neste apêndice, as propostas de implementacao da estrutura do *DRTO* composta pela aplicação de otimização *offline* e do sistema de otimização *online*.

Propõe-se aqui a seguinte estrutura (como mostra a Figura C.1) para a aplicação de otimização *offline (EMSO_DYNOPT.EXE)*, baseada no uso da infra-estrutura do software de simulação dinâmica *EMSO*. Esta aplicação consiste de um programa executável que realiza o fluxo de funções das diferentes partes do sistema. São proposta as criações de quatro *DLL's* (*Dynamic-link Library*) que são reusadas nas outras aplicações do sistema de *DRTO*. Estas *DLL's* separam as funções do otimizador em: *EMSO.DLL* – funcionalidades ligadas ao ambiente *EMSO*; *DAOP2NLP.DLL* – funcionalidades dos métodos de solução de *DAOP*; *DAOP_SOLVER.DLL* – funcionalidades relativas às diferentes camadas no processo de solução do *DAOP*; e *SE_SOLVER.DLL* – funcionalidades dos métodos de estimação de estados e calibração do modelo do processo.

Além da aplicação de otimização *offline*, propõe-se o sistema de *DRTO* (conjunto de aplicações *online* e em tempo real) consiste de três programas executáveis que realizam as funções das diferentes partes deste sistema, como mostra a Figura C.2. São propostas as criações de um gerenciador (*EMSO_MGR.EXE*), um estimador (*EMSO_SE.EXE*) e o otimizador (*EMSO_OPT.EXE*). Esta separação é sugerida pelo fato destas três aplicações terem funções bem diferentes e tempos de execuções bem distintos.

O estimador *online* (*EMSO_SE.EXE*) é uma aplicação que é executada de forma cíclica comandada pelo gerenciador do *DRTO* (Vide Figura C.3). E obtém as condições iniciais consistentes e os parâmetros do modelo através da solução de um problema de otimização dinâmica específico. O gerenciador deverá disparar a estimação e deverá estabelecer a tarefa a ser executada. Se a estratégia for de reconciliação e ajuste de parâmetros simultâneos, o gerenciador irá solicitar que o estimador utilize o problema de otimização combinado. Se a estratégia for seqüencial, o gerenciador irá definir a quando o problema de reconciliação será resolvido e o momento e as bases do ajuste de parâmetros. Caso este procedimento falhe, o gerenciado do *DRTO* irá somente atualizar os dados utilizando informações consolidadas da planta.

As informações geradas por esta aplicação serão usadas para resolver o problema de otimização dinâmica. Isto é feito de forma contínua, a esta função executada numa freqüência mais elevada que o otimizador. Este fato garante que as informações do estado da planta estejam mais atualizadas no momento em que o otimizador necessitar da informação.

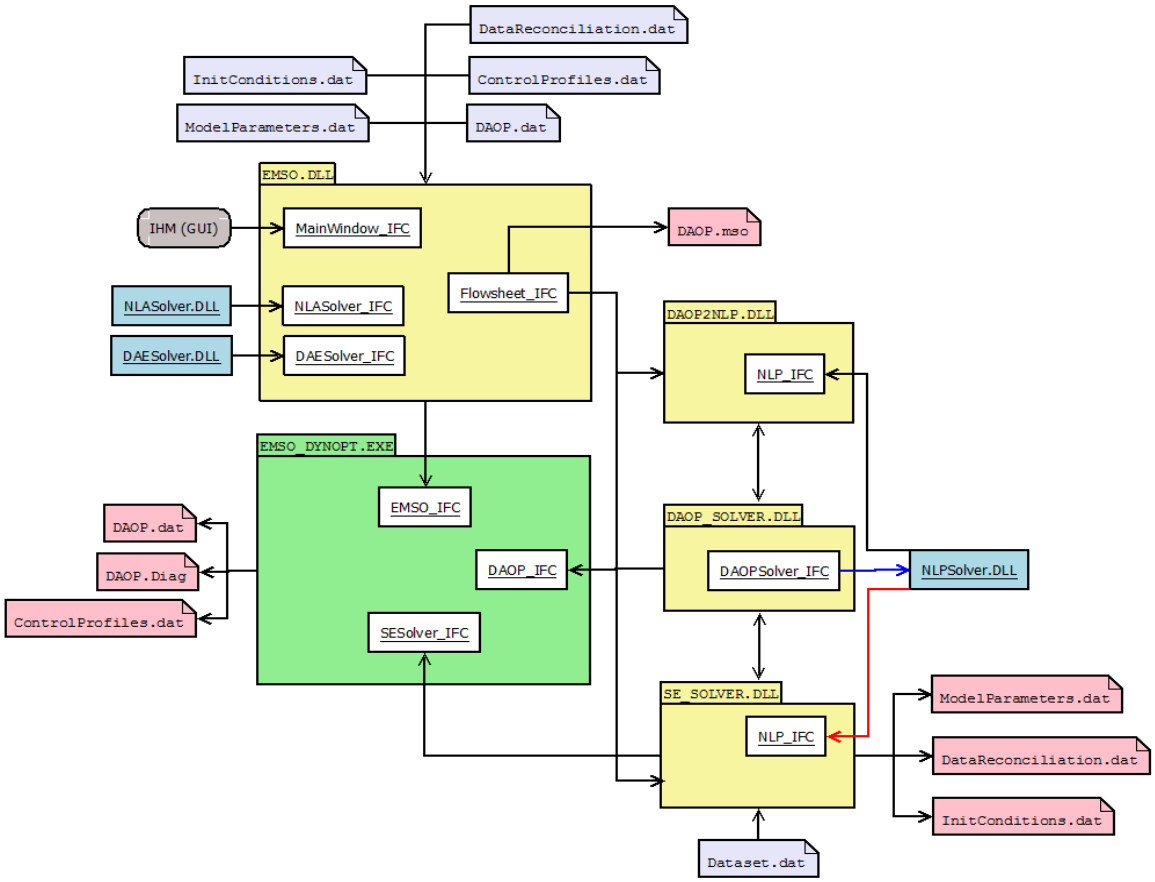


Figura C.1 – Estrutura esquemática do software de otimização *offline*.

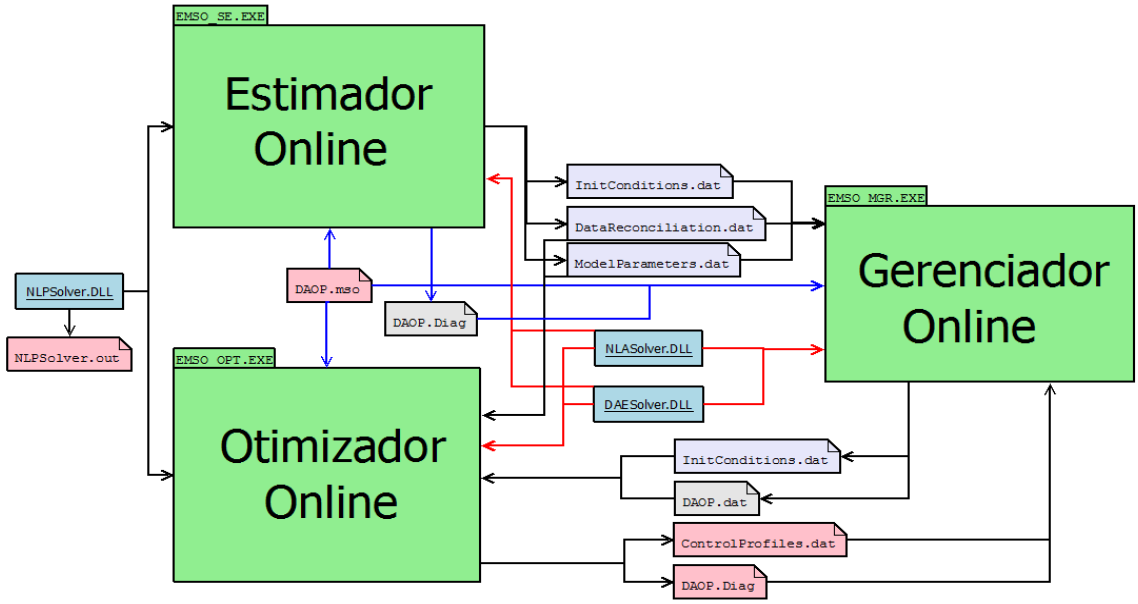


Figura C.2 – Esquema geral dos módulos do sistema de *DRTO*.

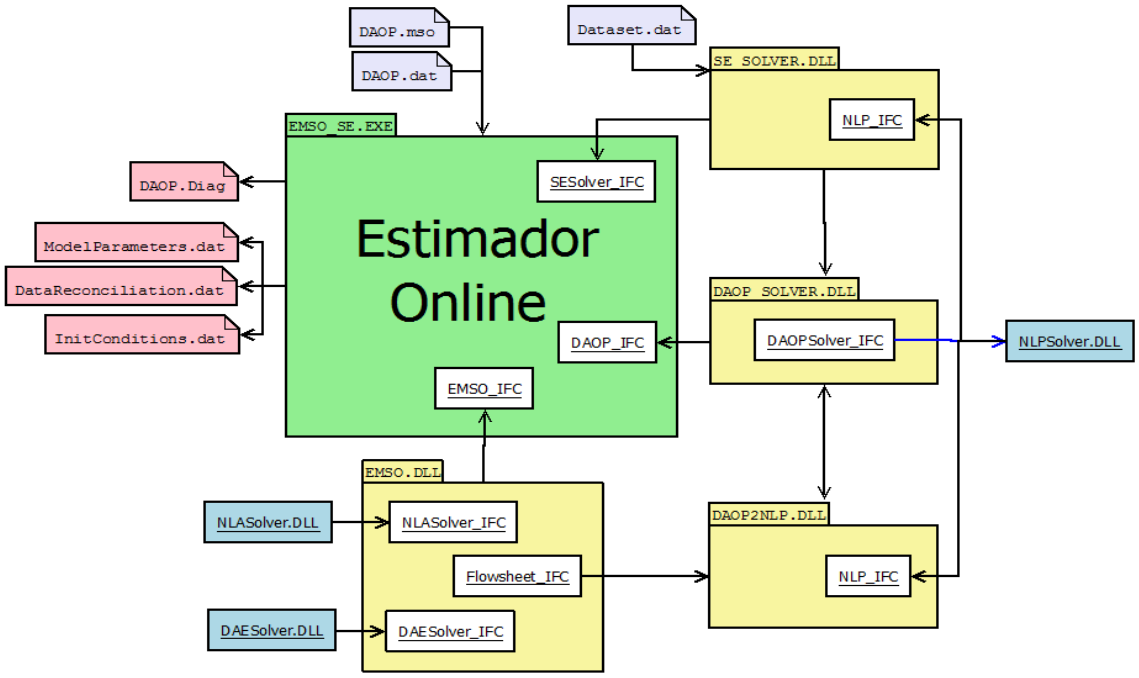


Figura C.3 – Esquema geral do estimador *online* do sistema de *DRTO*.

O estimador envia as condições iniciais (*InitConditions.dat*), os dados de reconciliação (*DataReconciliation.dat*) e de atualização de parâmetros (*ModelParameters.dat*) através de arquivos para o gerenciador *online*. O estimador *online* pode demorar até alguns minutos para resolver o problema de estimação. O gerenciador *online* analisa o estado atual da planta e passa as condições iniciais atualizadas (*InitConditions.dat*) e a estrutura do *DAOP* (*DAOP.dat*) para o otimizador *online* e comanda a sua execução.

A aplicação de otimização dinâmica *online* (*EMSO_OPT.EXE*) é semelhante à ferramenta *offline*. Esta aplicação é executada de forma cíclica e independente da aplicação de estimação de estados, porém de forma sincronizada e coordenada pelo gerenciador do *DRTO*. O otimizador *online* recebe as informações das condições iniciais consistentes atualizadas pelo gerenciador do *DRTO* através do arquivo *InitConditions.dat*. Monta o problema de otimização dinâmica, utilizando os recursos do *EMSO*, onde as definições do problema estão nos arquivos *DAOP.mso* e *DAOP.dat*, e os parâmetros atualizados do modelo estão no arquivo *ModelParameters.dat*. Com o método de otimização definido, o sistema resolve o problema de otimização dinâmica. Uma vez resolvido o *DAOP*, o otimizador envia a solução ótima (*ControlProfiles.dat*) para o gerenciador, assim com as informações de monitoração e diagnóstico (*DAOP.Diag*). Este esquema está mostrado na Figura C.4.

Da mesma forma que o otimizador *offline*, o otimizador *online* utiliza as bibliotecas do *EMSO* (*EMSO.DLL*), dos métodos de discretização do *DAOP* (*DAOP2NLP.DLL*) e das funcionalidades relativas às camadas do processo de solução, adaptação de malhas do *DAOP* e refinamento da estrutura da solução ótima (*DAOP_SOLVER.DLL*).

As funções de inicialização do motor do *EMSO*, a realização da simulação dinâmica, o estabelecimento do problema de otimização dinâmica (*DAOP*) e as avaliações de funções de simulação para o algoritmo de solução de *DAOP* são tarefas que estão ligadas ao ambiente de simulação e otimização do *EMSO*, e são executadas através da *"EMSO.DLL"*.

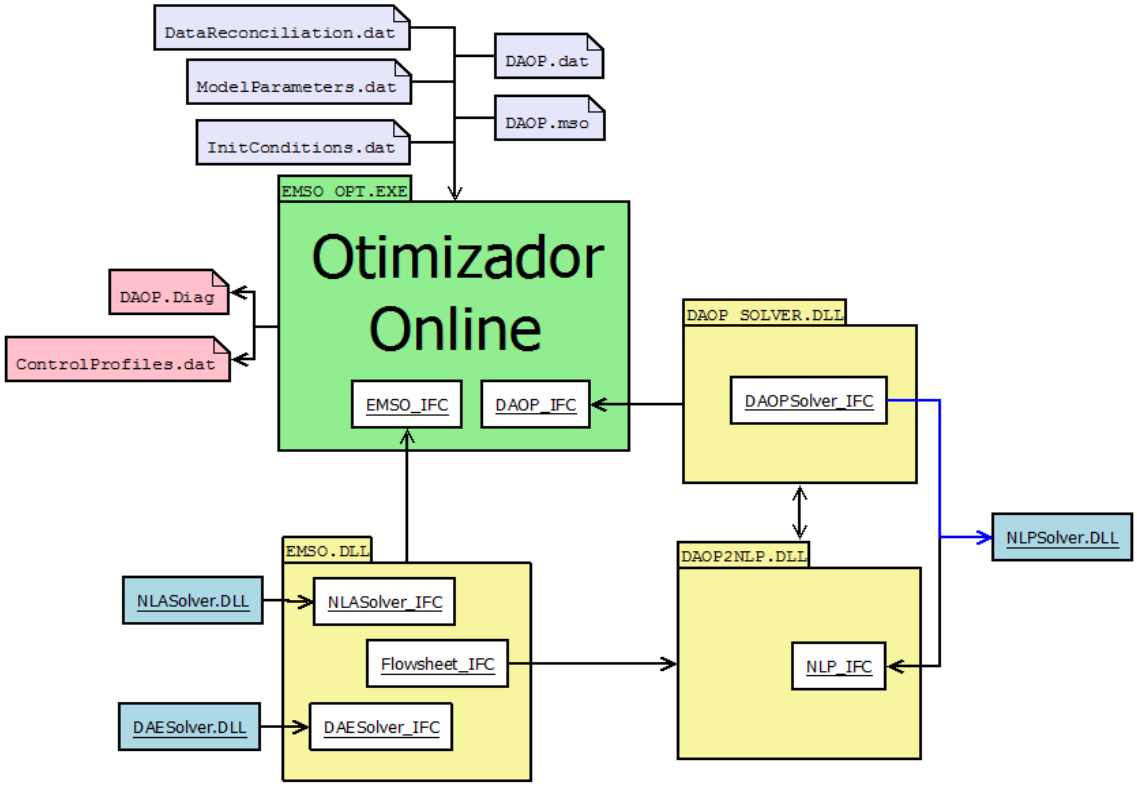


Figura C.4 – Esquema geral do otimizador *online* do sistema de *DRTO*.

A aplicação de gerenciamento do *DRTO* (*EMSO_MGR.EXE*) tem a função de coordenar as tarefas a serem realizadas pelo *DRTO*, como mostra o esquema da Figura C.5. O gerenciador é uma aplicação executada de forma cíclica, com freqüência de execução elevada e conectada à planta (ex.: a cada minuto). O mesmo deve gerenciar a estimação de estados, inclusive a seqüência das tarefas de reconciliação de dados, ajuste de parâmetros do modelo e de estimação das condições iniciais consistentes. Além disso, também deve decidir quando usar dados antigos em caso de falha de alguma das operações da estimação de estados. Neste caso, o gerenciador realiza a integração do modelo do processo até o instante atual (momento do disparo do otimizador). A decisão do momento do disparo do estimador de estados e do otimizador também deve ser realizado pelo gerenciador.

O gerenciador também define se o otimizador deverá interromper a sua execução em andamento e reiniciar nova otimização, ou se deverá concluir a rodada atual. Também, deve gerenciar a implementação das ações de controle e envio de informações para monitoração do sistema. A comunicação com os dispositivos de controle deve ser controlada pelo gerenciador de operações e a coleta, o envio e validação de dados dos instrumentos da planta são efetuados em tempo real e conectados com servidor *OPC* do sistema de controle digital. Uma vez obtidos os resultados do otimizador, o gerenciador fará uma análise dos resultados do otimizador (do arquivo *ControlProfiles.dat*) e

implementar os perfis ótimos de controle no sistema de controle da planta. Esta tarefa é executada através de um procedimento de bateladas que envia a cada instante um ponto ótimo a ser perseguido pelo sistema de controle da planta. Além disso, o gerenciador gera informações para monitoração de resultados e diagnóstico e sintonia. Em outras palavras, o gerenciador de operações é um módulo que coordena e estabelece as comunicações de cada parte do sistema e é responsável por manter a modularidade e flexibilidade do sistema de *DRTO*. O mesmo deve ter uma estrutura de servidor de aplicativos.

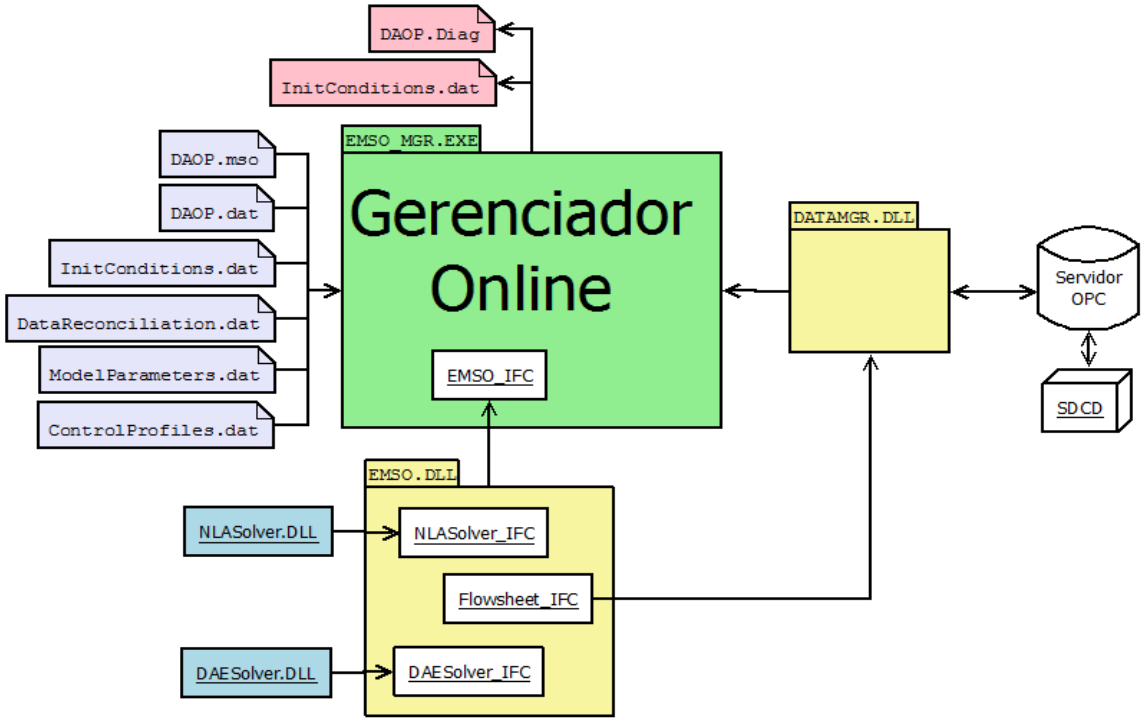


Figura C.5 – Esquema geral do gerenciador *online* do sistema de *DRTO*.

Outro aspecto importante ligado a este módulo é a necessidade de desempenho computacional aceitável, para obter a solução otimizada dentro do tempo disponível para o ciclo de execução. Isto pode remeter até a utilização de técnicas de processamento paralelo nas tarefas de avaliações de funções do modelo de otimização dinâmica, na integração numérica dos estados e sensibilidade e nas operações de álgebra linear do solver. Isto pode trazer o desempenho computacional do sistema de *DRTO* a patamares requeridos pelos usuários.

Apêndice D - Sintaxe da Formulação do DAOP

Para resolver o *DAOP* é necessário, primeiramente, formular o problema de forma estruturada. Para isso, é necessário criar uma sintaxe para construir o *DAOP*. Uma vez definida esta sintaxe de construção do *DAOP* é necessário projetar um interpretador das funções *EML* adicionais. Para definir as características do interpretador da *EML* para o modelo de otimização dinâmica, propõem-se algumas sintaxes adicionais para o sistema. O usuário tem a liberdade de representar o modelo de otimização de diversas formas, o mesmo precisa somente seguir a sintaxe do *EMSO*. A lógica de construção do problema de otimização dinâmica é a seguinte:

```
DAOPOptimization «Nome_da_Seção_Otimização» {as <Nome_do_Fluxogama_de_Processo>}
PARAMETERS    "Declarações de parâmetros de problema de otimização";
       SET           "Valores dos parâmetros";
VARIABLES     "Declarações das variáveis do problema de otimização";
       INITIAL       "Valores iniciais das variáveis diferenciais";

       CONTROL       "Definição de variáveis de controle"
       FINAL_TIME    "Definição de tempo final"
       MINIMIZE/MAXIMIZE   "Definição de função objetivo"
       CONSTRAINTS   "Definição de restrições nas variáveis de estado"
       EQUATIONS     "Eqs Diferenciais e algébricas de e lógicas usadas no DAOP"
       OPTIONS       "Definição dos métodos de solução do problema"
              DAOPSolver(...);
              NLPSolver(...);
              DAESolver(...);
End
```

Definições das variáveis de controle

De modo geral, qualquer variável diferencial, algébrica ou até mesmo parâmetro do modelo do processo pode ser eleita variável de decisão. Porém, deve-se ter o cuidado ao estabelecer uma variável de controle, pois nem todas as variáveis podem se tornar manipuladas. Uma especificação equivocada pode tornar o sistema *DAE* estruturalmente ou até mesmo numericamente singular. Isto faz com que se obtenham múltiplas soluções (caso sub-especificado) ou solução do *DAE* inviável (caso sobre-especificado). Uma análise baseada em grafos (Soares e Secchi, 2010; Soares, 2007) ou usando o algoritmo de Tarjan (Duff e Reid, 1978) adaptado para sistemas dinâmicos podem ser opções para evitar tais equívocos.

A seção **CONTROL** representa o conjunto de variáveis de controle ou parâmetro conforme escrito no modelo. Esta seção tem a seguinte sintaxe:

```
CONTROL       "Definição de variáveis de controle"
       <Variável> (Guess = [...],Lower = <Mínimo>, Upper = <Máximo>,
       MaxDelta = <Variação>, LowerTolerance =<Valor>, UpperTolerance = <Valor>,
       StdDev =<Valor>, Decision =<´Tipo de Decisão´>, Unit = <´Unidade´>);
              ...
       <Parâmetro> (Guess = <Valor>,Lower = <Mínimo>, Upper = <Máximo>,
       Decision = <´Tipo de Decisão´>, Unit = <´Unidade´>);
              ...
```

As variáveis têm os seguintes significados:
FREE - variável incluída no conjunto de variáveis de decisão;
FIXED - variável excluída do conjunto de variáveis de controle;
PW_CTE - controle com perfil constante por partes;
PW_LIN_C - controle com perfil linear por partes contínuo entre elementos;
PW_LIN_D - controle com perfil linear por partes descontínuo entre elementos;

Para as variáveis de controle, temos: <Variável> é o nome da variável, **Guess** o seu valor ou vetor inicial de perfis de controle (se for um valor escalar, então o perfil é constante ao longo de todo o horizonte de otimização), **Lower** o limite mínimo, **Upper** o limite máximo, **MaxDelta** a taxa de variação máxima da variável (opcional), **Decision** o tipo de perfil de controle e **Unit** a unidade de medida (ao omitir, utiliza a unidade definida no modelo do processo). No caso de flexibilização dos controles, os seguintes atrubutos são adicionados: **LowerTolerance** é a tolerância da violação do limite mínimo, **UpperTolerance** é a tolerância da violação do limite máximo e **StdDev** é o desvio padrão da variável de controle para relaxamento.

Exemplo: `u1 (Guess = [10,...,12], Lower = 5, Upper = 15, MaxDelta = 0.2, Decision = {PW_CTE/ PW_LIN_C/ PW_LIN_D/ FIXED}, Unit = 'atm');`

Para os parâmetros, temos: < Parâmetro> é o nome do parâmetro, **Guess** o seu valor inicial, **Lower** o limite mínimo, **Upper** o limite máximo, **Decision** inclusão (FREE) ou exclusão (FIXED) do parâmetro como variável de decisão do *DAOP* e **Unit** a unidade de medida..

Exemplo: `p1 (Guess = 10, Lower = 5, Upper = 15, Decision = {FREE/FIXED}, Unit = 'm');`

A imposição de máximo movimento das variáveis de controle permitido no problema de otimização dinâmica deve sempre estar presente. Ela deve ser estabelecida na forma de taxa máxima de mudança, e caso os perfis estabelecidos sejam de constantes por partes devem ser transformados em máxima amplitude dos degraus considerando os tamanhos dos intervalos discretos de tempos.

A seção **FINAL_TIME** define as condições do problema quando o tempo final é livre, ou seja, se torna uma variável de decisão. Esta seção tem a seguinte sintaxe:

```
FINAL_TIME (Guess = <Valor>, Lower = <Mínimo>, Upper = <Máximo>,
            Decision = <Tipo de Decisão>, Unit = <'Unidade'>);
```

Para o tempo final livre, temos: **Guess** o seu valor inicial, **Lower** o limite mínimo, **Upper** o limite máximo, **Decision** inclusão (FREE) ou exclusão (FIXED) do tempo final como variável de decisão do *DAOP* e **Unit** a unidade de medida.

Exemplo: `FINAL_TIME Guess = 10, Lower = 5, Upper = 15, Decision = {FREE/ FIXED}, Unit = 'h');`

Construção da função objetivo

A seção **MINIMIZE/MAXIMIZE** define a função objetivo do problema. Nesta seção, são escritas as expressões da função objetivo. Se tiver mais de um objetivo, o método de solução do problema multi-objetivos é definido na seção de opções do *DAOP*. Esta seção tem a seguinte sintaxe:

```
MINIMIZE Expressão 1
(Target=<Valor>, Weight=<Valor>, Lower=<Valor>, Upper=<Valor>, Unit=<´Unidade´>);
MAXIMIZE Expressão 2;
...
OPTIONS       "Definição dos métodos de solução do problema"
       DAOPSolver(
              MultiObjectiveMethod = <"Nome_do_Método">,...);
```

As opções de métodos de otimização multi-objetivos têm os seguintes significados:

WS	- método de média ponderada;
GP	- método de programação por metas;
ECONS	- método do ε-restrito
NBI	- método de interseção da fronteira normal;
NNC	- método da restrição normal normalizada;

Na otimização multi-objetivos, tem-se parâmetros adicionais que podem ser utilizados (a depender do problema e do método de otimização utilizado). O **Target** é o valor alvo ou utópico da função objetivo, o **Weight** é o peso relativo da função objetivo, o **Lower** e **Upper** são os seus limites, **Unit** a sua unidade de medida.

Os parâmetros para cada algoritmo também são definidos na seção **OPTIONS**.
Exemplo:

```
MINIMIZE J = PHI + Intg(PSI,t0,tf);
MAXIMIZE x2;

OPTIONS
       DAOPSolver (MultiObjectiveMethod = {WS/ GP/ ECONS, NBI, NCC}],…)
```

Construção das Restrições

As restrições são constituídas de equações e inequações do modelo de otimização dinâmica. Quando uma expressão de desigualdade não estiver associada a uma variável diferencial, algébrica ou de controle, a mesma deverá ser atribuída a uma variável genérica na forma:

[LabelName] := Expression i;

Exemplo: $z = x^2 y$; $y = 1/x$ ao atribuir $y := 1/x$, y será substituído em todas as equações em que estiver presente. Portanto, as duas expressões equivalem à $z = x$.

Esta atribuição permite a redução do número de variáveis e equações no *DAOP*. Ao atribuir uma expressão a uma variável, na interpretação do problema, esta variável é simplesmente substituída pela expressão nas equações dependentes da variável. Isto faz

com que não acrescente linhas no vetor de resíduos e nem na matriz Jacobiana durante a solução do problema ou até mesmo reduza não linearidades pela combinação de equações. As variáveis atribuídas poderão ser acompanhadas ao longo do tempo através de gráficos de tendência ou tabela de dados.

As restrições de desigualdade serão automaticamente escritas no problema de otimização dinâmica através dos domínios definidos na seção de declaração das variáveis.

Quando uma restrição for escrita explicitamente na forma de uma variável diferencial, algébrica ou de controle, a mesma deverá sobrescrever a respectiva restrição de domínio.

A seção CONSTRAINTS define as restrições do problema. Esta seção tem a seguinte sintaxe:

```
CONSTRAINTS  "Definição de restrições nas variáveis de estado"
<Variável> (Lower = [...], Upper = [...], PointTime = [...], Unit = <´Unidade´>,
ConstrType = <´Tipo de Restrição´>, LowerTolerance =[...], UpperTolerance =[...],
StdDev =<Valor>);
             ...
```

Para as restrições, temos: <Variável> é o nome da restrição, **Lower** o limite mínimo (se for um valor escalar, então o limite é constante ao longo de todo o estágio de otimização; se for vetor, então são restrições de pontos interiores e o limite só é imposto nos pontos definidos), **Upper** o limite máximo, **PointTime** é o vetor dos elementos de tempo para as restrições do tipo INTERIOR_POINT do *DAOP* , **ConstrType** é o tipo de restrição do problema e **Unit** a unidade de medida. No caso de flexibilização de restrições, os seguintes atributos são adicionados: **LowerTolerance** é a tolerância da violação do limite mínimo, **UpperTolerance** é a tolerância da violação do limite máximo e **StdDev** é o desvio padrão da restrição para relaxamento. Se for de outro tipo, este atributo é omitido.

Exemplo:

```
z1 (ConstrType = {PATH/ END_POINT}, Lower = 5, Upper = 15);
Z2 (ConstrType=INTERIOR_POINT, Lower=[5,6,4], Upper=[15,14,16], PointTime=[1,3,8];
```

Definição das opções do método de solução

As opções são definidas para utilizar os métodos de solução de *DAOP*, de *NLP*, de adaptação de malhas e detecção de estrutura da solução. Nesta seção define-se o método utilizado e seus parâmetros de sintonia. Esta seção tem a seguinte sintaxe:

```
OPTIONS      "Definição dos métodos de solução do problema"
      DAOPSolver(
            Method        = "Nome_do_Método", [ex.: Method = {SSH/ MSH/ OCEF},]
            NumberOfElements = Valor,  [ex.: NumberOfElements = 10 ,]
NCOL              = Valor,  [ex.: NCOL  = 3,]
ElementsGrouping = "Definição",[ex.: ElementsGrouping = {YES/ NO},]
Groups              = [(Grupo 1),..., (Grupo n)],
      [ex.: Groups = [(1:3),4,(6:10)],]
MeshAdaptation    = "Definição",[ex.: MeshAdaptation    = {YES/ NO},]
AdaptationMethod= "Nome do Metodo",[ex.:AdaptationMethod={Wavelet},]
AdaptationOptions    = (Opção 1 = #",..., "Opção n = #")
[ex.: AdaptationOptions = (MAX_LEVEL = 3,...)]
StructureDetection = "Definição",[ex.:StructureDetection={YES/ NO},]
DetectionOptions     = (Opção 1 = #",..., "Opção n = #")
[ex.: DetectionOptions = (MAX_DETECT = 10,...)]
            Startup       = "Tipo_de_Partida",[ex.: Startup = {COLD/ WARM},]
```

```
            AbsoluteAccuracy = Valor,  [ex.: AbsoluteAccuracy = 1e-8,]
                  ...);
      NLPSolver(
            File = "Nome_do_Solver",   [ex.:File={ipopt_emso/complex/optpp },]
            RelativeAccuracy    = Valor,      [ex.: RelativeAccuracy = 1e-6,]
                  ...);
      DAESolver(
            File   = "Nome_do_Solver", [ex.: File = {dassl/ mebdef/ pside},]
            AbsoluteAccuracy    = Valor,      [ex.: AbsoluteAccuracy = 1e-8,]
                  ...);
      NLASolver(
            File   = "Nome_do_Solver", [ex.: File = {sundials / nlasolver},]
            AbsoluteAccuracy    = Valor,      [ex.: AbsoluteAccuracy = 1e-8,]
                  ...);
```

Definição das equações

A seção EQUATIONS representa as expressões das restrições igualdade e desigualdade adicionais do modelo de otimização dinâmica. Estas expressões são as mesmas adotadas pelo simulador *EMSO*. Além disso, podem-se definir atribuições a algumas variáveis. Neste caso, estas variáveis não aparecem na estrutura do problema de otimização dinâmica, reduzindo sua dimensão. Esta seção tem a seguinte sintaxe:

```
EQUATIONS     "Equações Diferenciais/ algébricas e algumas lógicas usadas no DAOP"

              Expressão da igualdade 1;
              Var_i := Expressão da igualdade 2;
              ...
              Expressão da igualdade n_1 (Label = <´Nome´>);

              Expressão da desigualdade 1 (Label = <´Nome´>);
              ...
              Expressão da desigualdade n_2;
```

O comando de **Label** pode ser utilizado para identificar uma expressão no problema de otimização ou para visualização de seus resultados.

Exemplo:
```
A_3 = A_1 + A_2;
A_3 := A_1 + A_2;
A_3 ≤ A_1 + A_2 (Label = ´D´);
```

Problema de otimização com único estágio ou definições globais

Em problemas de um único estágio, todas as definições são globais. As definições comuns a todos os estágios podem ser colocadas na seção global. A seção global sempre está antes da definição do primeiro estágio do problema *DAOP*.

```
DAOPOptimization «Nome_da_Seção_Otimização» {as <Nome_do_Fluxogama_de_Processo>}

PARAMETERS    "Declarações de parâmetros de problema de otimização";
...
      SET           "Valores dos parâmetros";
...
VARIABLES     "Declarações das variáveis do problema de otimização";
...
      INITIAL       "Valores iniciais das variáveis diferenciais";
...
      MINIMIZE/MAXIMIZE   "Definição de função objetivo"
...
```

```
       CONSTRAINTS   "Definição de restrições nas variáveis de estado"
...
       EQUATIONS     "Equações Diferenciais/algébricas e algumas lógicas do DAOP"
              ...
OPTIONS       "Definição dos métodos de solução do problema"
```

Problema de otimização com múltiplos estágios

Todas as definições, equações e lógicas são escritas dentro da estrutura [STAGE #]. Todas as declarações pertencem ao estágio em referência até encontrar um novo estágio.

```
DAOPOptimization «Nome_da_Seção_Otimização» {as <Nome_do_Fluxogama_de_Processo>}
       ....
STAGE <#k-1>
       PARAMETERS    "Declarações de parâmetros de problema de otimização";
...
              SET           "Valores dos parâmetros";
...
       VARIABLES     "Declarações das variáveis do problema de otimização";
...
              INITIAL       "Valores iniciais das variáveis diferenciais";
...
              MINIMIZE/MAXIMIZE   "Definição de função objetivo"
...
              CONSTRAINTS   "Definição de restrições nas variáveis de estado"
...
              EQUATIONS     "Equações Diferenciais/algébricas e lógicas do DAOP"
              ...
       OPTIONS       "Definição dos métodos de solução do problema"
              JUNCTIONS     "Definição condições de junções entre dois estágios"
STAGE  <#k>
```

A declaração JUNCTION se refere às condições de junções dos estágios, normalmente são condições de continuidade das variáveis de estado diferenciais entre estágios. Esta seção tem a seguinte sintaxe:

```
JUNCTIONS     "Definição condições de junções entre os estágios k e k-1"
       <Variável no estágio k>, <Variável no estágio k-1>;
              ...
```

Para as junções, temos: <Variável no estágio k> é o nome da variável no estágio k, <Variável no estágio k-1> é o nome da variável no estágio anterior k-1. Se forem todas junções de continuidade, basta omitir esta diretiva.

Exemplo: `z1, y2;` Isto equivale dizer que: $z_1^{(k)} = y_2^{(k-1)}$.

Apêndice E - Sintaxe da reformulação do DAOP

Quando problema de otimização não estiver na forma padrão para serem resolvidos pelos softwares de solução de *DAOP*, o mesmo deverá ser reformulado para a sua forma padrão. Neste caso, as expressões das funções objetivo, restrições, variáveis de controle, de tempo final devem ser reformuladas.

A forma padrão da função objetivo deve estar no formato Mayer ($\phi(x(t_f))$). Se a função objetivo estiver, po exemplo, no formato de: `MINIMIZE J = PHI + Intg(PSI,t0,tf);` o interpretador irá reformular o objetivo conforme a Tabela E.1.

Tabela E.1 – Reformulação da função objetivo.

De:	Para:

`MINIMIZE J = PHI + Intg(PSI,t0,tf);`	`MINIMIZE J;`
	EQUATIONS
	`J = PHI + IntPSI;`
	`diff(IntPSI) = PSI;`
	INITIAL
	`IntPSI = 0;`

Da mesma forma, todas as expressões de desigualdade que estiverem na seção EQUATIONS deverão ser transformadas em uma equação algébrica. No exemplo dado anteriormente: `A3 ≤ A1 + A2;` o interpretador irá reformular as restrições de desigualdade conforme a Tabela E.2.

Tabela E.2 – Reformulação das restrições de desigualdade.

De:	Para:

EQUATIONS
`A3 ≤ A1 + A2;`
	EQUATIONS
	`C1 = A1 + A2 – A3;`
	CONSTRAINTS
	`C1 (Lower = 0.0, Upper = Inf (Infinito), ConstrType = PATH);`

Para as variáveis de controle que forem constantes por partes com restrições de movimentos (`MaxDelta`), propõe-se aqui que o problema seja reformulado. Isto porque, na operação industrial, as manipulações dos controles e suas taxa de variações são restritas. Primeiramente, são poucos os casos de otimização onde a posição inicial de uma variável de controle não é pré-definida pela operação. Se a posição inicial for livre e sua variação ilimitada, simplesmente se omite a diretiva `MaxDelta` na seção `CONTROL`. Neste caso, as definições das variáveis de controle não são alteradas, utilizando a variável na sua forma original. Caso contrário, o interpretador adiciona a equação da Tabela E.3 que transforma a variável de controle real em variável de estado e limita a sua atuação.

Tabela E.3 – Reformulação das variáveis de controle.

De:	Para:
CONTROL `R1 (Guess = 0.25,Lower = 0.0, Upper = 10, MaxDelta = 0.1, Decision = PW_CTE);`	CONTROL `DR1 (Guess = 0.0,Lower = -0.1, Upper = 0.1, Decision = PW_CTE);` EQUATIONS

	diff(R_1) = DR_1; INITIAL R_1 = 0.25; CONSTRAINTS C_1 (Lower = 0.0, Upper = 10, ConstrType = PATH);

Caso o tempo final seja livre, declarado na seção FINAL_TIME o problema é reformulado, acrescentando-se um parâmetro de controle e normalizando o tempo nas equações diferenciais (0 a 1). Esta seção tem a sintaxe dada pela Tabela E.4.

Tabela E.4 – Reformulação da definição de tempo final.

De:	Para:
FINAL_TIME (Guess = 10,Lower = 5, Upper = 15, Decision = FREE);	CONTROL t_f (Guess = 10,Lower = 5, Upper = 15, Decision = FREE); EQUATIONS diff(x_i) = t_f * f_i(.); ...
MINIMIZE t_f; FINAL_TIME (Guess = 10,Lower = 5, Upper = 15, Decision = FREE);	MINIMIZE t_f; CONTROL t_f (Guess = 10,Lower = 5, Upper = 15, Decision = FREE); EQUATIONS diff(x_i) = t_f * f_i(.); FINAL_TIME (Guess = 1,Lower = 1, Upper = 1, Decision = FIXED);

Nesta mesma etapa de interpretação do modelo de otimização, é efetuada uma pré-análise heurística das restrições do problema. O interpretador verifica se o valor máximo está menor do que o seu limite ou domínio mínimo ou se seu valor mínimo está maior do que o seu domínio máximo. Se isto acontecer, o otimizador deve alertar sobre o problema e não executar a tarefa solicitada. Isto é aplicável às seções de CONTROL, CONSTRAINTS e FINAL_TIME. Além disso, o interpretador executa a análise de consistência do sistema *DAE*, como é feito atualmente no *EMSO*. Uma vez que o problema *DAOP* está na sua forma padrão e não há inconsistências no modelo de otimização, a aplicação está pronta para resolver o referido problema.

Apêndice F - Utilização de arquivos na solução do DAOP

A cada rodada do otimizador *online*, o mesmo deverá criar automaticamente os arquivos "*InitConditions.dat*", "*ControlProfiles.dat*" e "*DAOP.dat*", e deverá criar sob demanda os arquivos "*ModelParameters.dat*", "*DataReconciliation.dat*" e "*Dataset.dat*". Estes arquivos poderão ser utilizados de diferentes formas pela aplicação *offline*. Por outro lado, o otimizador *offline* poderá exportar dados nos arquivos "*ModelParameters.dat*" e "*DataReconciliation.dat*" para serem utilizados na aplicação *online* e outras tarefas do otimizador *offline*.

O arquivo **InitConditions.dat** contém as informações das condições iniciais consistentes do problema de otimização dinâmica. O Otimizador *online* irá criar este arquivo a cada ciclo de execução. O objetivo da criação deste arquivo é permitir que o sistema sempre tenha as condições iniciais atualizadas para serem utilizadas por qualquer módulo do *DRTO*, inclusive para repetir uma determinada rodada do otimizador *online* no otimizador *offline*. Estas condições iniciais podem ser usadas para verificar a rodada *online* tal como aconteceu na planta em qualquer momento, ou como base para um estudo de caso ou até mesmo para diagnóstico. O processo de geração deste arquivo será explicado na seção de estimação de estados e gerenciamento de tarefas, neste capítulo. Este arquivo contém as informações das condições iniciais das variáveis de estado diferenciais ($x(t_0)$) e suas derivadas em relação ao tempo ($\dot{x}(t_0)$), das variáveis algébricas ($y(t_0)$) e de controle ($u(t_0)$).

O arquivo `InitConditions.dat` tem o seguinte layout proposto:

```
Case Name: <DAOPFileName>.<DAOPOptimizatioName>
Date: YYYY-MM-DD hh:mm:SS
INITIAL CONDITIONS
[DIFFERENTIAL]
<VarName>, <x0>, <dx0>
…
[ALGEBRAIC]
<VarName>, <y0>
...
[CONTROL]
<VarName>, <u0>
...
```

O arquivo **ControlProfiles.dat** contém as informações das trajetórias de controle do problema de otimização dinâmica. Estas trajetórias podem ser referentes à receita ótima da última rodada do otimizador a ser implementada e as trajetórias iniciais a serem utilizadas pelo otimizador *online* na obtenção da próxima solução do problema (na partida a quente). Outro objetivo da criação deste arquivo é permitir que se repita ou verifique determinada rodada do otimizador *online* no otimizador *offline*. Além disso, estes perfis poderão ser utilizados juntamente com as definições do *DAOP*, e verificar os resultados do otimizador *online*.

O arquivo `ControlProfiles.dat` tem o seguinte layout proposto:

```
Case Name: <DAOPFileName>.<DAOPOptimizatioName>
Date: YYYY-MM-DD hh:mm:SS
CONTROL PROFILES
[STAGE 1]
TimePoints = [t0, t1, t2, ...,tne-1];
FinalTime  = tf;
<ControlName>, [u(t0), u(t1), u(t2), ..., u(tne-1)];
…
[STAGE 2]
...
...
[STAGE ns]
...
```

O arquivo **DAOP.dat** contém as informações do problema de otimização dinâmica. O otimizador *online* cria este arquivo a cada rodada do problema com as definições do *DAOP*. Este arquivo contém as definições do *DAOP* que são diferentes daquelas contidas no arquivo de modelos, ou seja, as informações do *DAOP.mso* serão sobrescritas por aquelas que estão no *DAOP.dat*. Vale lembrar que o operador pode ligar ou desligar variáveis de controle e restrições, bem como mover seus limites. Estas informações que são alteradas durante a operação em tempo real do sistema, serão escritas neste arquivo. O objetivo da criação deste arquivo é permitir que se repita determinada rodada do otimizador *online* no otimizador *offline*.

O arquivo `DAOP.dat` tem o seguinte layout:

```
Case Name: <DAOPFileName>.<DAOPOptimizatioName>
Date: YYYY-MM-DD hh:mm:SS
DAOP CHANGING DEFINITIONS
# Devem ser colocados apenas os atributos alterados ou acrescentados
[STAGE 1]
  FINAL_TIME (Lower = <Mínimo>, Upper = <Máximo>,...);
 [CONTROL]
<Variável> (Lower = <Mínimo>, Upper = <Máximo>,...);
...
[CONSTRAINTS]
<Variável> (Lower = <Mínimo>, Upper = <Máximo>,...);
...
[STAGE 2]
...
...
[STAGE ns]
...
```

O arquivo **ModelParameters.dat** contém as informações do ajuste de parâmetros para ser usado na otimização *offline* e na análise de resultados do otimizador *online*. Nesta análise dos resultados são necessárias informações do ajuste de parâmetros para avaliar as incertezas no modelo e conseqüentemente nos resultados do otimizador. A matriz de covariância dos parâmetros é a informação mais importante no processo de decisão da implementação dos resultados do otimizador. Outro objetivo da criação deste arquivo é permitir que se repita determinada rodada do otimizador *online* no otimizador *offline*.

O arquivo `ModelParameters.dat` tem o seguinte layout proposto:

```
Case Name: <FlowSheetFileName>.<FlowSheetName>
Date: YYYY-MM-DD hh:mm:SS
PARAMETERS ESTIMATION
[ADJUST_INFORMATION]
NumberOfParameters      = np;
NumberOfMeasures        = nx_exp + ny_exp + nu_exp;
NumberOfExperiments     = nexp;
AdjustConfidence        = α; (ex.: α = 0.95)
ParsConfidenceInterval = η;
[PARAMETERS]
<NomeParâmetro>, <Valor>, <Mínimo>, <Máximo>, 'UnidadeEng';
...
[COVARIANCES]
<NomeParâmetro>, [σ_1,1];
<NomeParâmetro>, [σ_2,1, σ_2,2];
...
<NomeParâmetro>, [σ_np,1, σ_np,2, ..., σ_np,np];
[MEASURES_VARIANCES]
<NomeVarMedida>, <VarianciaExperimental>;
...
```

O arquivo **DataReconciliation.dat** contém as informações da reconciliação de dados de medição da otimização *online*. Estas informações são usadas para corrigir, da melhor maneira possível, os valores das variáveis medidas. Estas informações consistem basicamente dos valores dos *biases* (*BIAS = Valor estimado – Valor Experimental*) obtidos do processo de reconciliação de dados do otimizador *online*. Esta é a melhor informação para corrigir as variáveis medidas sem realizar a etapa de reconciliação. Claro que o usuário poderá realizar a reconciliação de dados no otimizador *offline* com dados experimentais, mas há situações onde o mesmo não precisa fazer esta operação. Este mesmo *BIAS* deve ser usado para compensar o erro de medição na implementação das ações de controle que são medidas. Lembre que sempre há um erro sistemático nas medidas e este erro também deve ser considerado no momento de implementar as ações de controle. Neste arquivo também, acrescenta-se a informação da variância da medida para avaliar a precisão do instrumento de medição. Isto também deve ser feito se o usuário desejar comparar os resultados do otimizador *offline* com informações de medições da planta.

O arquivo `DataReconciliation.dat` tem o seguinte layout proposto:

```
Case Name: <FlowSheetFileName>.<FlowSheetName>
Date: YYYY-MM-DD hh:mm:SS
DATA RECONCILIATION
[BIAS]
<NomeVarMedida>, <BIAS>, <VarianciaExperimental>;
...
```

Para que o otimizador importe os exporte cada um destes arquivos, é necessário estabelecer a instrução na área de opções de cada seção do modelo de otimização.

A sintaxe na área de `OPTIONS` é dada nas Tabelas F.1 e F.2.

Tabela F.1 – Instruções das importações de dados de arquivos.

Instrução	**Para Importar**
`ImportInitConditions`(File = "<FileName>");	condições iniciais consistentes
`ImportControlProfiles`(File = "<FileName>");	Perfis de controle
`ImportDAOPInfo`(File = "<FileName>");	informações do *DAOP*
`ImportModelParameters`(File = "<FileName>");	informações do ajuste de parâmetros
`ImportDataReconciliation`(File = "<FileName>");	informações da reconciliação de dados

Tabela F.2 – Instruções das exportações de dados de arquivos.

Instrução	**Para Exportar**
`ExportInitConditions`(File = "<FileName>");	Condições iniciais consistentes
`ExportControlProfiles`(File = "<FileName>");	Perfis de controle
`ExportDAOPInfo`(File = "<FileName>");	informações do *DAOP*
`ExportModelParameters`(File = "<FileName>");	informações do ajuste de parâmetros
`ExportDataReconciliation`(File = "<FileName>");	informações da reconciliação de dados

Apêndice G - Sintaxe adicionais na formulação do DAOP

Na etapa de construção do modelo de otimização dinâmica (topico 4.1.2.1), têm-se definições referentes à calibração do modelo e reconciliação de dados das medidas e de estudo de casos. Estas definições se referem à formulação da estimação de parâmetros e reconciliação de dados para *DRTO* assim como diretivas adicionais para a realização de estudos de casos. Uma vez definidas as sintaxes destas modelagens, é necessário estabelecer as regras do interpretador da linguagem do sistema. A seguir, são apresentadas estas regras de modelagem destes casos.

Sintaxe de calibração do modelo e reconciliação de dados

Quanto à estratégia, têm-se as seguintes possibilidades:
`SequentialEstimation` - Estratégia de estimação de parâmetros e reconciliação seqüencial;
`SimultEstimation` - Estratégia de estimação de parâmetros e reconciliação simultâneos;

Esta seção tem a seguinte sintaxe:

```
OPTIONS       "Definição dos métodos de solução do problema"
       EsimationSolution(
              Method        = "Nome_do_Método", [ex.: Method = {CEKF/ MHE },]
              NumberOfStages = Valor,     [ex.: NumberOfElements = 10 ,]
...);
       EstimationStrategy = { SequentialEstimation / SimultEstimation };
```

Sintaxe de estudo de casos

A sintaxe da seção `DAOPCaseStudy` tem vários itens em comum com a seção `DAOPOptimization`. Caso altere a função objetivo, é necessário apenas escreve a seção MINIMIZE/MAXIMIZE com a nova expressão do estudo de casos. Caso adicione ou remova equações, é necessário apenas alterar as equações na seção EQUATIONS. Se for alterar o conteúdo da equação é só necessário reescrever a equação na nova expressão. Se a equação for nova, de escrever a equação nova na seção EQUATIONS do estudo de casos. Caso deseje remover a equação, deve sinalizar `REMOVE` no lado esquerdo da equação na mesma seção (ex.: `REMOVE diff(C); para remover a equação diff(C) = A + B;`). Caso deseja adicionar, reescrever ou remover (ex.: `REMOVE A;`) as variáveis de controle, é necessário apenas alterar as definições na seção CONTROL do estudo de casos. Caso deseja adicionar ou remover restrições (ex.: `REMOVE A;`), é necessário apenas alterar as restrições na seção CONSTRAINTS do estudo de casos. Também, se desejar alterar as definições do tempo final, deve-se reescrever a seção FINAL_TIME do estudo de casos. Se desejar alterar opções do otimizador ou utilizar informações de arquivos, devem-se alterar as diretivas na seção OPTIONS do estudo de casos ou acrescentar a instrução `ImportInitConditions`, `ImportControlProfiles`, `ImportDAOPInfo`, `ImportModelParameters` e `ImportDataReconciliation` para importar os respectivos dados de arquivo.

A sintaxe deste caso é simples, e consiste apenas de definir o tipo de caso na seção `OPTIONS` e definir os arquivos a serem importados do caso real, quando for o caso.

```
DAOPOptimization «Nome_da_Seção_Otimização» {as <Nome_do_Fluxogama_de_Processo>}

       <Todas as definições do DAOP>

OPTIONS # Incluir a opção
DAOPCaseStudy = ANALIZE_RESULTS;
```

A sintaxe deste caso é simples, e consiste apenas de definir o tipo de caso na seção `OPTIONS` e definir os arquivos a serem importados do caso real, quando for o caso.

```
DAOPOptimization «Nome_da_Seção_Otimização» {as <Nome_do_Fluxogama_de_Processo>}

       <Todas as definições do DAOP>

OPTIONS # Incluir a opção
DAOPCaseStudy = FEASIBILITY;
```

A sintaxe deste caso é simples, e consiste apenas colocar a diretiva `REMOVE` nas restrições não essenciais e definir os arquivos a serem importados do caso real, quando for o caso.

```
DAOPOptimization «Nome_da_Seção_Otimização» {as <Nome_do_Fluxogama_de_Processo>}

       <Todas as definições do DAOP>

CONSTRAINTS  "Definição das restrições a serem removidas"
REMOVE C1, C2, ...;

OPTIONS # Incluir a opção
       DAOPCaseStudy = OPTIMALITY;
```

Para flexibilizar o problema, as diretivas das restrições devem ser escritas na seguinte sintaxe:

```
DAOPOptimization «Nome_da_Seção_Otimização» {as <Nome_do_Fluxogama_de_Processo>}

       <Todas as definições do DAOP>

OPTIONS # Incluir a opção
DAOPCaseStudy = RELAXATION;
```

Apêndice H - Visualização dos resultados do otimizador

Pode-se selecionar e visualizar as variáveis na forma gráfica ou em tabelas. Desta forma, o usuário pode analisar os perfis de controle e seus limites, das variáveis de estado com suas restrições, a função objetivo e outras variáveis complementares (Figura H.1). Se o usuário desejar visualizar os perfis das variáveis em uma determinada iteração, o mesmo deve comandar a pausa no *menu* de tarefas (Figura 4.8). E visualizar os perfis intermediários da mesma forma comentada anteriormente. Note que esta capacidade de executar novamente e visualizar um problema vivido na otimização *online* ou *offline* iteração-a-iteração (proposto aqui) é um diferencial importante no diagnóstico de problema em relação aos sistemas existentes.

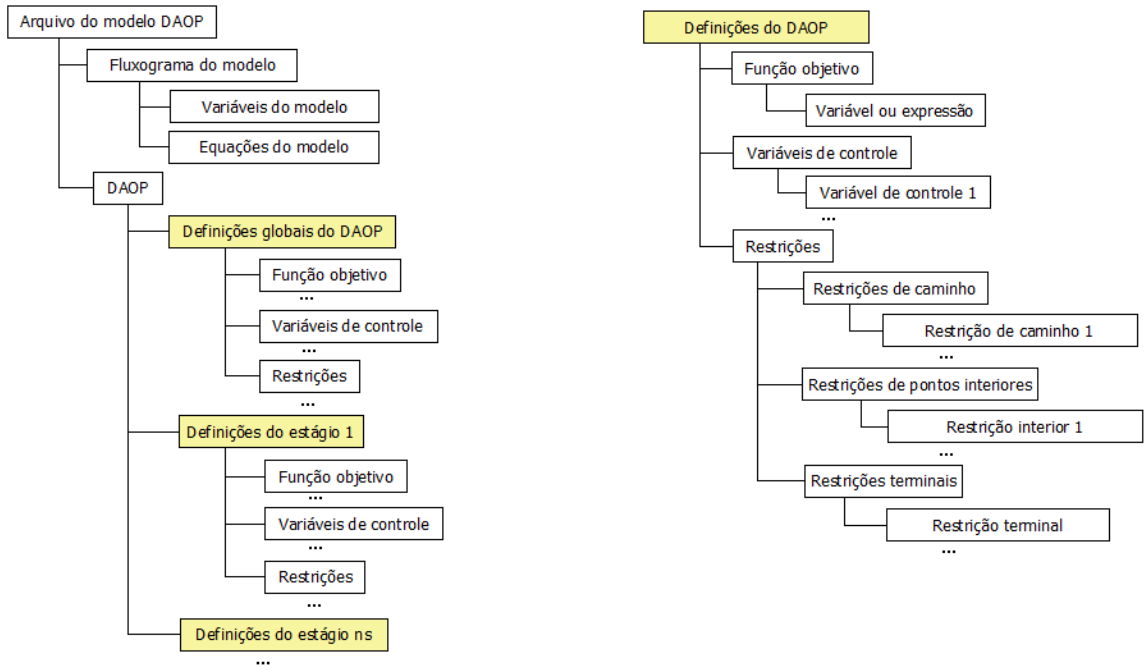


Figura H.1 – Árvore de visualização dos resultados do *DAOP*.

Apêndice I - Estrutura da discretização do DAOP

O processo de discretização é dividido em blocos a níveis do problema de *NLP*, do estágio e do elemento finito. Por uma questão didática, iremos descrever aqui o processo de discretização de trás para frente.

No problema de NLP

A montagem dos vetores e matrizes de avaliações das funções consiste basicamente das agregações dos diferentes estágios do *DAOP*. Estas agregações envolvem os resíduos das restrições do *DAOP* (Res), do vetor de variáveis de decisão para o *NLP* (Ctrl), das Jacobianas das restrições com as variáveis de decisão (Jac). Além disso, tem-se a avaliação da função objetivo, o gradiente e a Hessiana da função Lagrangeana do *DAOP*. As estruturas destes vetores e matrizes são as mesmas para todos os métodos. O que se alteram são as suas dimensões e variáveis envolvidas, a depender do método de solução do *DAOP* utilizado (*single-shooting*, *multi-shooting* e colocação em elementos finitos).

O vetor de variáveis de decisão do *NLP* consiste de sub-vetores de controles em cada estágio do *DAOP* (u_1, u_2, ..., u_{ns}). Estes vetores podem ser diferentes, e dependem das variáveis de controle envolvidas em cada estágio e do método de solução do problema de otimização. Uma variável de controle pode estar presente em um determinado estágio, mas não estar presente em outro. Para o algoritmo de *NLP*, cada elemento de controle é diferente na função ou no tempo. Além disso, ao final de cada estágio, há a variável tempo final (t_f), quando livre. Se a mesma não for livre, este elemento desaparece do vetor. Entre um estágio e outro, há novas variáveis de decisão associadas às variáveis de estado diferenciais do modelo do processo (x_2, ..., x_{ns}). Estas variáveis de decisão são necessárias para garantir as continuidades destas variáveis nas junções dos estágios (vide Figura I.1).

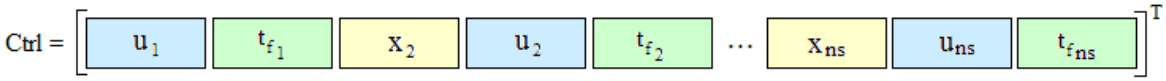


Figura I.1 – Montagem do vetor de variáveis de decisão do problema multi-estágios.

O vetor de resíduos do *NLP* consiste de sub-vetores das restrições impostas em cada estágio do *DAOP* (F_1, F_2, ..., F_{ns}). Estes vetores podem ser diferentes, e dependem das restrições envolvidas em cada estágio e do método de solução do problema de otimização. Além disso, na união dos estágios do *DAOP* há novas restrições de junções que são impostas ao problema. Estas junções dizem respeito às condições de continuidades das variáveis de estado diferenciais do modelo do processo. Estas condições são: $J_k = x_{ne}^{(k-1)} - x_0^{(k)} = 0$. Ou seja, os valores de x no último elemento do estágio $k-1$ tem que ser igual à condição inicial do mesmo x no estágio k. Sendo que a variável de decisão é $x_0^{(k)}$. Além disso, ao final do problema de otimização há uma restrição adicional de tempo final, onde a soma dos tempos finais em cada estágio tem que ser igual ao tempo final do *DAOP*. Lembre que os tempos finais de cada estágio podem ser livres e o tempo final total

ser fixo. Esta restrição de $t_f = \sum_{k=1}^{ns} t_f^{(k)}$ garante esta consistência. A montagem deste vetor de restrições é mostrada na Figura I.2.

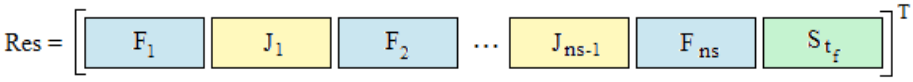


Figura I.2 – Montagem do vetor de resíduos das restrições do probl. multi-estágios.

A matriz Jacobiana das restrições com as variáveis de decisão do *NLP* consiste de sub-matrizes Jacobianas das restrições do *DAOP* impostas com as variáveis de controle e tempo final em cada estágio do *DAOP* (Jac_{u_k} e $Jac_{t_f^{(k)}}$). Estes vetores também podem ser diferentes, e dependem das restrições envolvidas em cada estágio e do método de solução do problema de otimização. Além disso, na união dos estágios do *DAOP* há Jacobianas relativas às novas restrições de junções que são impostas ao problema. Estas Jacobianas são as derivadas das condições de junção e das restrições em relação às condições iniciais das variáveis de estado diferenciais em cada estágio. A montagem deste vetor de restrições é mostrada na Figura I.3.

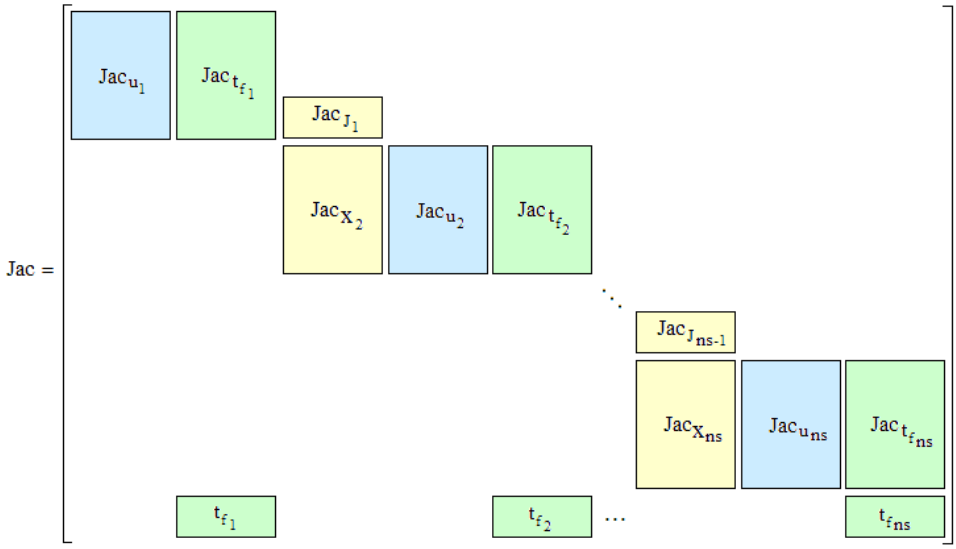


Figura I.3 – Montagem da matriz Jacobiana das restrições do probl. multi-estágios.

No estágio

Seguindo o mesmo raciocínio anterior, a montagem dos vetores e matrizes de avaliações das funções no estágio consiste basicamente das agregações dos mesmos resíduos (Res), dos controles (Ctrl), das Jacobianas (Jac). As estruturas destes vetores e matrizes são quase as mesmas para todos os métodos. O que diferencia é a existência das condições de continuidades entre os elementos finitos. O método *single-shooting* não tem condições de continuidade entre os elementos, pois o processo de solução é seqüencial, isto é, o modelo do processo e sua sensibilidade são integrados até o final de um elemento finito, e passados ao problema discreto. Depois, continua o processo de integração até o final do próximo elemento. Já na utilização dos métodos *multi-shooting* e colocação em elementos finitos, é necessário introduzir os termos das condições de continuidade dos estados. Da mesma

forma que na montagem do problema inteiro, as dimensões dos vetores e matrizes dependem das variáveis envolvidas e do método de solução utilizado. A Figura I.4 mostra a montagem deste vetor em cada caso.

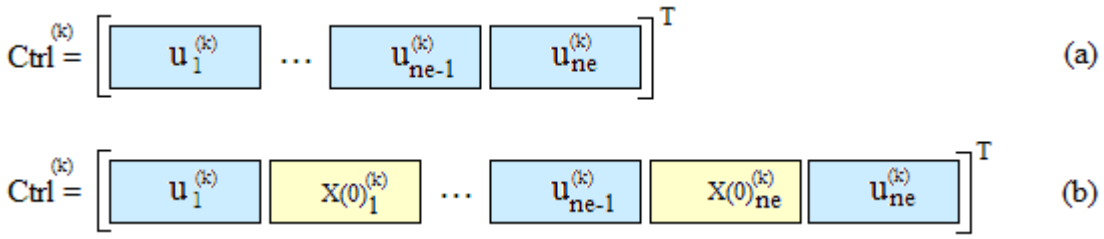


Figura I.4 – Montagem do vetor de variáveis de decisão no *NLP* no estágio *k* do *DAOP*. Caso (a), para o método *single-shooting* e caso (b), para os métodos *multi-shooting* e colocação em elementos finitos.

O vetor de resíduos do *NLP* em um estágio do *DAOP* consiste de sub-vetores das restrições impostas em cada elemento do estágio. Da mesma forma descrita anteriormente, estes vetores podem ser diferentes, e dependem das restrições envolvidas em cada estágio e do método de solução do problema de otimização. Além disso, é necessário introduzir as condições de continuidades das variáveis de estado do modelo do processo. Estas condições são: $C_i = x(t_i)_{i-1}^{(k)} - x_{0,i}^{(k)} = 0$. Ou seja, os valores de x em t_i do elemento $i - 1$ do estágio k tem que ser igual à condição inicial do mesmo x no elemento i do estágio k. Sendo que a variável de decisão é $x_{0,i}^{(k)}$. A montagem deste vetor de restrições é mostrada na Figura I.5.

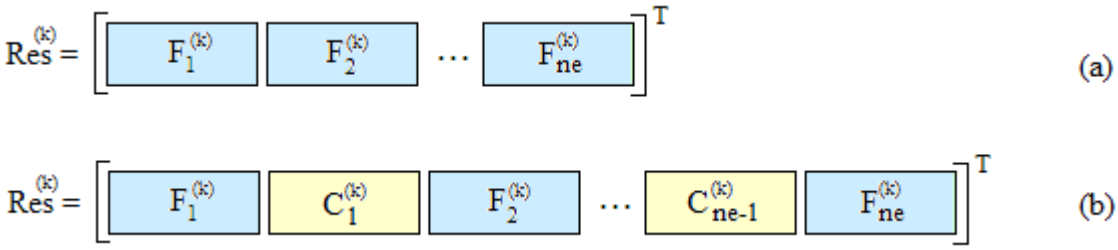


Figura I.5 – Montagem do vetor de resíduos do *NLP* no estágio *k* do *DAOP*. Caso (a), para o método *single-shooting* e caso (b), para os métodos *multi-shooting* e colocação em elementos finitos.

De forma semelhante, a matriz Jacobiana das restrições com as variáveis de decisão do *NLP* num determinado estágio do *DAOP*, consiste de sub-matrizes Jacobianas das restrições impostas com as variáveis de controle e tempo final em cada elemento de um determinado estágio do *DAOP*. Estes vetores também podem ser diferentes, e dependem das restrições envolvidas em cada estágio e do método de solução do problema de otimização. Além disso, na união dos elementos de um estágio do *DAOP* há Jacobianas relativas às novas restrições de continuidade que são impostas ao problema. Estas Jacobianas são as derivadas das condições de continuidade e das restrições em relação às condições iniciais das variáveis de estado em cada estágio. A montagem deste vetor de restrições é mostrada na Figura I.6.

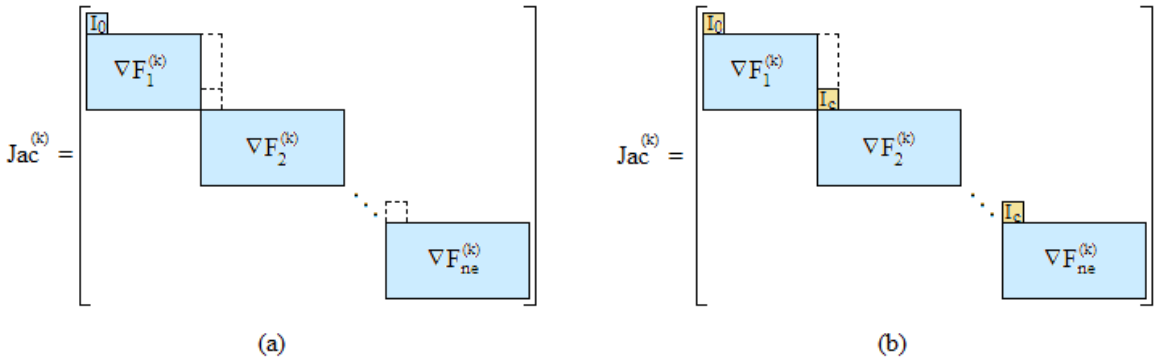


Figura I.6 – Montagem da matriz Jacobiana das restrições no estágio *k* do *DAOP*. Caso (a), para o método *single-shooting* e caso (b), para os métodos *multi-shooting* e colocação em elementos finitos.

No elemento finito

Finalmente, chegando à discretização no elemento finito, a montagem dos vetores e matrizes de avaliações das funções no elemento finito consiste basicamente das avaliações de funções no final do elemento e em pontos interiores. As estruturas dos vetores e matrizes específicos para cada método de solução e o conteúdo de cada elemento dependem das posições dos mesmos e dos tipos de variáveis de controle e de restrições envolvidas naquele ponto. Aqui serão apresentadas as construções dos controles, resíduos e Jacobianas para cada método aqui considerado.

No uso do método *single-shooting*, a integração do modelo é efetuada por algoritmos de *DAE* (ex.: *DASSLC*) de forma seqüencial para cada elemento finito, onde a condição final do elemento anterior passa a ser a condição inicial no novo elemento (vide Figura I.7).

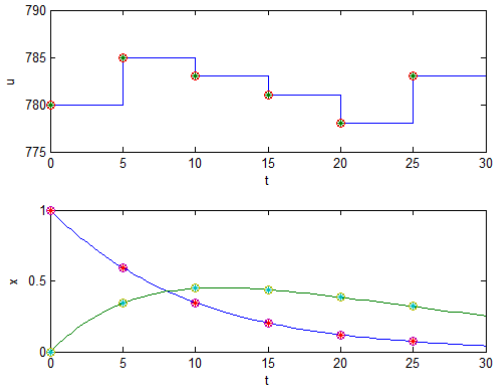


Figura I.7 – Esquema de discretização no método *single-shooting*.

O vetor das variáveis de decisão do *NLP* é composto simplesmente das variáveis de controle do *DAOP* original no final de um elemento i. Pode-se escrever o vetor da seguinte maneira:

$$u_i^{(k)} = \begin{bmatrix} u_{i,1} & \cdots & u_{i,nu} \end{bmatrix}^T; \qquad u_i^{(k)} \in \Re^{nu^{(k)}}$$

Os resíduos das restrições, para os elemento finitos de 1 até *ne* – 1 do estágio *k*, podem ser escritos como (Equação I.1):

$$F_{i,k} = \begin{bmatrix} F_{i,k}^{P} \\ F_{i,k}^{IP\notin P} \end{bmatrix}; \qquad \forall i = 1,\ldots,ne-1 \tag{I.1}$$

onde $F_{i,k}^{P} = \begin{bmatrix} F_{1,i,k}^{P} & \cdots & F_{npc_{i,k},i,k}^{P} \end{bmatrix}^{T}$ e $F_{i,k}^{IP\notin P} = \begin{bmatrix} F_{1,i,k}^{IP\notin P} & \cdots & F_{nipc_{i,k},i,k}^{IP\notin P} \end{bmatrix}^{T}$. Sendo $F_{i,k}^{P}$ os resíduos das restrições de caminho e $F_{i,k}^{IP\notin P}$ as restrições de pontos interiores (que não são restrições de caminho).

Para o elemento finito *ne* do estágio *k*, os resíduos devem incluir as restrições terminais, e podem ser escritos na forma (Equação I.2):

$$F_{ne,k} = \begin{bmatrix} F_{ne,k}^{P} \\ F_{ne,k}^{T\notin P} \end{bmatrix}; \qquad i = ne \tag{I.2}$$

onde $F_{ne,k}^{P} = \begin{bmatrix} F_{1,ne,k}^{P} & \cdots & F_{npc_{ne,k},ne,k}^{P} \end{bmatrix}^{T}$ e $F_{ne,k}^{T\notin P} = \begin{bmatrix} F_{k}^{T\notin P} & \cdots & F_{ntc_{k},k}^{T\notin P} \end{bmatrix}^{T}$. Sendo $F_{i,ne}^{P}$ os resíduos das restrições de caminho no elemento final do estágio e $F_{ne,k}^{T\notin P}$ as restrições terminais (que não são restrições de caminho).

A Jacobiana das restrições, para os elementos finitos de 1 até *ne* – 1 do estágio *k*, pode ser escrita como (Equação I.3):

$$Jac_{i,k}^{u} = \begin{bmatrix} dF_{i,k}^{P} \\ dF_{i,k}^{IP\notin P} \end{bmatrix}; \qquad \forall i = 1,\ldots,ne-1 \tag{I.3}$$

onde

$$dF_{i,k}^{P} = \begin{bmatrix} \left(\frac{\partial F_1^{P}}{\partial u_1}\right)_{i,k} & \cdots & \left(\frac{\partial F_1^{P}}{\partial u_{nu}}\right)_{i,k} \\ \vdots & \ddots & \vdots \\ \left(\frac{\partial F_{npc}^{P}}{\partial u_1}\right)_{i,k} & \cdots & \left(\frac{\partial F_{npc}^{P}}{\partial u_{nu}}\right)_{i,k} \end{bmatrix}; \quad dF_{i,k}^{IP\notin P} = \begin{bmatrix} \left(\frac{\partial F_1^{IP\notin P}}{\partial u_1}\right)_{i,k} & \cdots & \left(\frac{\partial F_1^{IP\notin P}}{\partial u_{nu}}\right)_{i,k} \\ \vdots & \ddots & \vdots \\ \left(\frac{\partial F_{nipc}^{IP\notin P}}{\partial u_1}\right)_{i,k} & \cdots & \left(\frac{\partial F_{nipc}^{IP\notin P}}{\partial u_{nu}}\right)_{i,k} \end{bmatrix}$$

Para o elemento finito *ne* do estágio *k*, a Jacobiana fica (Equação I.4):

$$Jac_{ne,k}^{u} = \begin{bmatrix} dF_{ne,k}^{P} \\ dF_{k}^{T\notin P} \end{bmatrix} \tag{I.4}$$

onde

$$dF_{ne,k}^{P} = \begin{bmatrix} \left(\frac{\partial F_1^P}{\partial u_1}\right)_{ne,k} & \cdots & \left(\frac{\partial F_1^P}{\partial u_{nu}}\right)_{ne,k} \\ \vdots & \ddots & \vdots \\ \left(\frac{\partial F_{npc}^P}{\partial u_1}\right)_{ne,k} & \cdots & \left(\frac{\partial F_{npc}^P}{\partial u_{nu}}\right)_{ne,k} \end{bmatrix}; \quad dF_k^{T\notin P} = \begin{bmatrix} \left(\frac{\partial F_1^{T\notin P}}{\partial u_1}\right)_{i,k} & \cdots & \left(\frac{\partial F_1^{T\notin P}}{\partial u_{nu}}\right)_{i,k} \\ \vdots & \ddots & \vdots \\ \left(\frac{\partial F_{ntc}^{T\notin P}}{\partial u_1}\right)_{i,k} & \cdots & \left(\frac{\partial F_{ntc}^{T\notin P}}{\partial u_{nu}}\right)_{i,k} \end{bmatrix}$$

No uso do método *multi-shooting*, a integração do modelo também é efetuada por algoritmos de *DAE* em paralelo para cada elemento finito, isto é, a condição inicial do elemento não está relacionada com a condição final do elemento anterior, até que se atinja a convergência do sistema (vide Figura I.8).

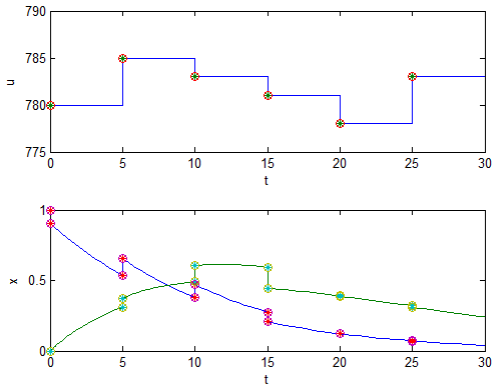


Figura I.8 – Esquema de discretização no método *multi-shooting*.

O vetor das variáveis de decisão do *NLP* é composto das variáveis de controle do *DAOP* original no final de um elemento *i* e das variáveis de estado no início do mesmo elemento. Pode-se escrever o vetor da seguinte maneira:

$$u_i^{(k)} = \begin{bmatrix} u_{i,1} & \cdots & u_{i,nu} \end{bmatrix}^T; \qquad u_i^{(k)} \in \Re^{nu^{(k)}}; \quad i = 1$$

e $$u_i^{(k)} = \begin{bmatrix} u_{i,1} & \cdots & u_{i,nu} & x_{i,1}^0 & \cdots & x_{i,nx}^0 \end{bmatrix}^T; \qquad u_i^{(k)} \in \Re^{(nu+nx)^{(k)}}; \qquad \forall i = 2,\ldots,ne$$

Os resíduos das restrições, para os elementos finitos do estágio *k*, podem ser escritos como (Equação I.5):

$$F_{1,k} = \begin{bmatrix} F_{1,k}^P \\ F_{1,k}^{T\notin P} \end{bmatrix}^T; \qquad i = 1 \quad \text{e} \quad F_{i,k} = \begin{bmatrix} F_{i,k}^C \\ F_{i,k}^P \\ F_{i,k}^{IP\notin P} \end{bmatrix}^T; \qquad \forall i = 2,\ldots,ne \qquad \text{(I.5)}$$

onde :

$$F_{i,k}^P = \begin{bmatrix} F_{1,i,k}^P & \cdots & F_{npc_{i,k},i,k}^P \end{bmatrix}^T, F_{i,k}^{IP\notin P} = \begin{bmatrix} F_{1,i,k}^{IP\notin P} & \cdots & F_{nipc_{i,k},i,k}^{IP\notin P} \end{bmatrix}^T \text{ e } F_{i,k}^C = \begin{bmatrix} F_{1,i,k}^C & \cdots & F_{nx_{i,k},i,k}^C \end{bmatrix}^T.$$

Sendo $F_{i,k}^P$ os resíduos das restrições de caminho, $F_{i,k}^{IP\notin P}$ as restrições de pontos interiores (que não são restrições de caminho) e $F_{i,k}^C$ representam as condições de continuidade das variáveis de estado.

A Jacobiana das restrições, para os elementos finitos do estágio *k*, pode ser escrita como (Equação I.6):

$$Jac^{u}_{1,k} = \begin{bmatrix} dF^{P}_{1,k} \\ dF^{T\notin P}_{1,k} \end{bmatrix}; \qquad i = 1 \quad \text{e} \quad Jac^{u}_{i,k} = \begin{bmatrix} 0 & dF^{C}_{i,k} \\ dF^{P}_{i,k} & 0 \\ dF^{IP\notin P}_{i,k} & 0 \end{bmatrix}; \quad \forall i = 2,\ldots,ne \qquad \text{(I.6)}$$

onde

$$dF^{P}_{i,k} = \begin{bmatrix} \left(\frac{\partial F^{P}_{1}}{\partial u_1}\right)_{i,k} & \cdots & \left(\frac{\partial F^{P}_{1}}{\partial u_{nu}}\right)_{i,k} \\ \vdots & \ddots & \vdots \\ \left(\frac{\partial F^{P}_{npc}}{\partial u_1}\right)_{i,k} & \cdots & \left(\frac{\partial F^{P}_{npc}}{\partial u_{nu}}\right)_{i,k} \end{bmatrix}; \quad dF^{IP\notin P}_{i,k} = \begin{bmatrix} \left(\frac{\partial F^{IP\notin P}_{1}}{\partial u_1}\right)_{i,k} & \cdots & \left(\frac{\partial F^{IP\notin P}_{1}}{\partial u_{nu}}\right)_{i,k} \\ \vdots & \ddots & \vdots \\ \left(\frac{\partial F^{IP\notin P}_{nipc}}{\partial u_1}\right)_{i,k} & \cdots & \left(\frac{\partial F^{IP\notin P}_{nipc}}{\partial u_{nu}}\right)_{i,k} \end{bmatrix}$$

$$dF^{C}_{i,k} = \begin{bmatrix} \left(\frac{\partial F^{C}_{1}}{\partial x_1}\right)_{i,k} & \cdots & \left(\frac{\partial F^{C}_{1}}{\partial x_{nx}}\right)_{i,k} \\ \vdots & \ddots & \vdots \\ \left(\frac{\partial F^{C}_{nx}}{\partial x_1}\right)_{i,k} & \cdots & \left(\frac{\partial F^{C}_{nx}}{\partial x_{nx}}\right)_{i,k} \end{bmatrix}$$

No uso do método de colocação em elementos finitos, a integração do modelo é efetuada por quadratura utilizando colocação ortogonal em cada elemento finito no horizonte de otimização (vide Figura I.9).

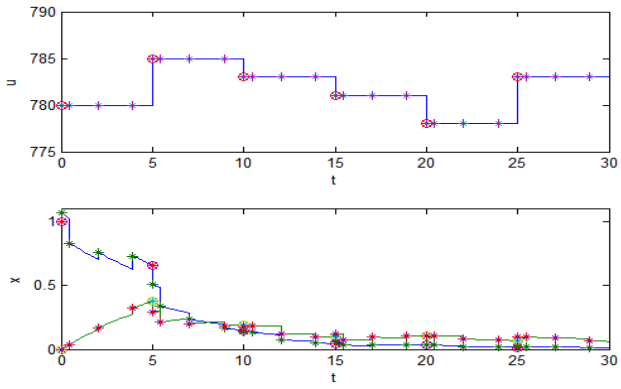


Figura I.9 – Esquema de discretização com colocação ortogonal em elementos finitos.

As posições dos pontos de colocação são definidas pela obtenção das raízes do polinômio de Jacobi, A Figura I.10 mostra as raízes para os números de pontos de colocação (*ncol*) de 1 a 6.

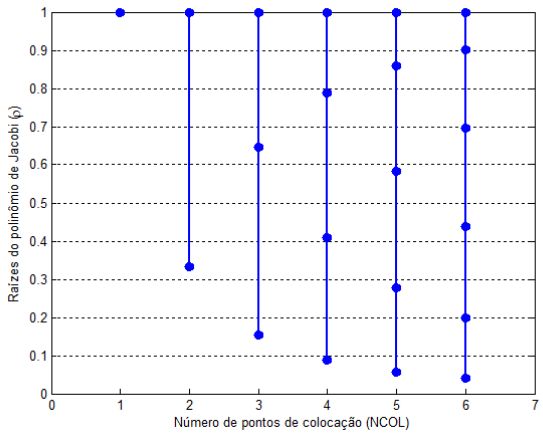


Figura I.10 – Posições dos pontos de colocação.

Na utilização deste método, as soluções de quadratura do sistema *DAE* e a solução do problema *DAOP* são obtidas simultaneamente. A seguir serão apresentados os vetores e matrizes utilizadas na utilização deste método.

O vetor das variáveis de decisão do *NLP* é composto pelas variáveis de estado e suas derivadas no tempo, as variáveis algébricas e as variáveis de controle do *DAOP* em cada ponto de colocação de cada elemento finito *i*. Pode-se escrever o vetor da seguinte maneira:

$$z_i = \begin{bmatrix} \dot{x}_{i,1} & x_{i,1} & y_{i,1} & u_{i,1} & \cdots & \dot{x}_{i,ncol} & x_{i,ncol} & y_{i,ncol} & u_{i,ncol} \end{bmatrix}^T ; \; z_i \in \Re^{(2nx+ny+nu)\times ncol}$$

Os resíduos das restrições, para cada ponto de colocação *j* do elemento finito *i* do estágio *k*, podem ser escritos como (Equação I.7):

$$F_{i,j}^{(k)} = F\left(\dot{x}_{i,j}^{(k)}, x_{i,j}^{(k)}, y_{i,j}^{(k)}, u_{i,j}^{(k)}, p, t_{i,j}^{(k)}\right) \tag{I.7}$$

onde:

$$F = \begin{bmatrix} f(x, y, u, p, t) - \dot{x} \\ h(x, y, u, p, t) - y \end{bmatrix}$$

Para as Condições de Continuidade, tem-se (Equação I.8):

$$x_{i+1} = x_i + h_i \sum_{q=1}^{ncol} \Omega_q(1)\dot{x}_{i,q} \;\text{ sendo }\; C_{i-1}^{(k)} = x_{i-1}^{(k)} - x_i^{(k)} + h_{i-1}^{(k)} \sum_{q=1}^{ncol} \Omega_q(1)\dot{x}_{i-1,q}^{(k)} \tag{I.8}$$

Além disso, há restrições adicionais (*AC*) que são condições impostas aos perfis de controle (ex.: no perfil constante por partes $u_{i,j+1} = u_{i,j}$; onde *j* é um ponto de colocação interno de um elemento *i*).

Desta forma, podem-se escrever os resíduos das restrições, para o elemento finito *i* do estágio *k* como:

$$F_i^{(k)} = \begin{bmatrix} F_{i,1} & \cdots & F_{i,ncol} & AC_i & C_{i,1} & \cdots & C_{i,ncol} \end{bmatrix}^T ;\ RHS_i \in \Re^{(2nx+ny)\times ncol+nac} \ \forall i = 1,\ldots,ne$$

onde $AC_i = \begin{bmatrix} AC_{i,1} & \cdots & AC_{i,nac} \end{bmatrix}^T ;\ AC_i \in \Re^{nac}$ e $C_{i,j} = \begin{bmatrix} C_1^{i,j} & \cdots & C_{nx}^{i,j} \end{bmatrix}^T ;\ C_i \in \Re^{nx}$.

A Jacobiana das restrições, para os elementos finitos do estágio *k*, pode ser escrita como (Equação I.9):

$$\nabla F_{i,k}^P = \begin{bmatrix} \begin{bmatrix} \nabla_{x_0} F_1^i \\ \vdots \\ 0 \end{bmatrix} & \begin{bmatrix} \nabla F_1^i & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \nabla F_{ncol}^i \end{bmatrix} & \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \\ \begin{bmatrix} \nabla AC_0^i \end{bmatrix} & \begin{bmatrix} \nabla AC_0^i & \cdots & \nabla AC_0^i \end{bmatrix} & [0] \\ \begin{bmatrix} -I_{nx} \\ \vdots \\ -I_{nx} \end{bmatrix} & \begin{bmatrix} D_1^i & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & D_{ncol}^i \end{bmatrix} & \begin{bmatrix} I_{nx} \\ \vdots \\ I_{nx} \end{bmatrix} \end{bmatrix} \qquad \text{(I.9)}$$

onde $\nabla F_{i,k}^P \in \Re^{((2nx+ny)\times ncol+nac)\times(2\times nx+(2nx+ny+nu)\times ncol)}$ e:

$$\nabla F_q^i = \begin{bmatrix} \nabla_{\dot{x}_q} F^i & \nabla_{x_q} F^i & \nabla_{y_q} F^i & \nabla_{u_q} F^i \end{bmatrix};\ \nabla F_q^i \in \Re^{((nx+ny))\times(2\times nx+ny+nu)}$$

$$D^i = \begin{bmatrix} I_{nx} & D_1^i & 0_{nx\times ny} & \cdots & 0_{nx} & D_{ncol}^i & 0_{nx\times ny} \end{bmatrix};\ D^i \in \Re^{nx\times((2\times nx+ny)\times ncol)}$$

$$I_{nx} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix};\ D_q^i = \begin{bmatrix} h_i\Omega_q(1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & h_i\Omega_q(1) \end{bmatrix};\ 0_{nx} = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix};\ 0_{nx\times ny} = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

A visão geral da matriz Jacobiana para um estágio do *DAOP* pode ser observada em (Equação I.10):

$$JAC = \begin{bmatrix} -I_{nz} & & & & & & & & \\ dFZ_1^1 & dF\dot{Z}Y_1^1 & 0 & & & & & & \\ \vdots & \vdots & \vdots & & & & & & \\ dFZ_{ncol}^1 & dF\dot{Z}Y_{ncol}^1 & 0 & & & & & & \\ I_{nz} & D^1 & -I_{nz} & & & & & & \\ & & dFZ_1^2 & dF\dot{Z}Y_1^2 & 0 & & & & \\ & & \vdots & \vdots & \vdots & & & & \\ & & dFZ_{ncol}^2 & dF\dot{Z}Y_{ncol}^2 & 0 & & & & \\ & & I_{nz} & D^2 & -I_{nz} & & & & \\ & & & & & \ddots & & & \\ & & & & & & dFZ_1^{ne} & dF\dot{Z}Y_1^{ne} & 0 \\ & & & & & & \vdots & \vdots & \vdots \\ & & & & & & dFZ_{ncol}^{ne} & dF\dot{Z}Y_{ncol}^{ne} & 0 \\ & & & & & & I_{nz} & D^{ne} & -I_{nz} \end{bmatrix} \qquad \text{(I.10)}$$

Apêndice J - Monitoração do DRTO

A ferramenta de monitoração procura acompanhar a evolução das soluções do sistema de *DRTO* cconsiderando os aspectos do acompanhamento gerencial e técnico das soluções de otimização dinâmica.

Acompanhamento gerencial da otimização dinâmica

O relatório gerencial deve conter as informações que fornecem aos gerentes uma visão geral da utilização e dos resultados do uso do *DRTO*, como mostra a Figura J.1. Além disso, pode escolher alguns indicadores a serem visualizados na forma gráfica.

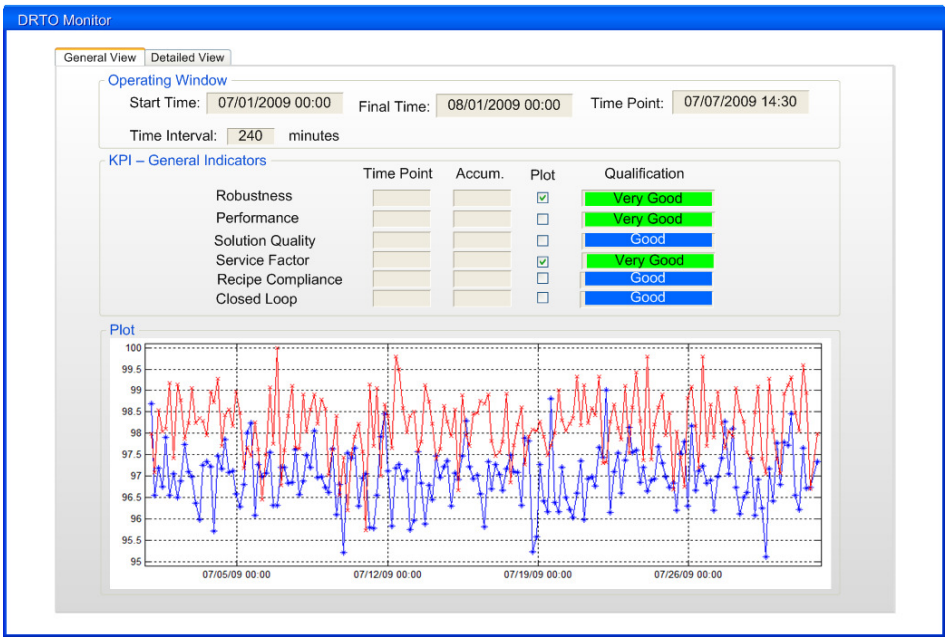


Figura J.1 – Visão geral dos indicadores gerenciais.

Acompanhamento técnico da otimização dinâmica

Na monitoração é realizado o acompanhamento de indicadores técnicos referentes aos diferentes aspectos do sistema de *DRTO*. Os aspectos são: o número de estimação de estados; de reconciliações de dados; de ajustes de parâmetros do modelo; de soluções dos problemas de otimização; e número de implementações dos resultados do otimizador. Num primeiro nível, tem-se uma visão geral das rodadas de cada tarefa do sistema de *DRTO*, como mostra a Figura J.2. O usuário deve escolher o período de acompanhamento e também um ponto específico, para visualizar os dados de uma determinada rodada do sistema.

Dado um horário dentro do período em análise, o sistema de monitoração localiza a rodada passada mais próxima do ponto desejado. Com isso, o usuário pode visualizar os detalhes de cada atividade.

Figura J.2 – Visão geral dos indicadores técnicos do sistema de *DRTO*.

Na visão geral da atualização de parâmetros, deve-se acompanhar: os valores históricos da função objetivo do ajuste de parâmetros (representa a diferença do previsto x experimental), como mostra a Figura J.3. O coeficiente de determinação (R^2) entre os dados experimentais e o modelo (representa a qualidade do ajuste); a qualidade da estimação (condicionamento da matriz de covariância dos parâmetros); a robustez da estimação de parâmetros (% problemas resolvidos) e o número de parâmetros ativos nas restrições (indica problemas na estimação);

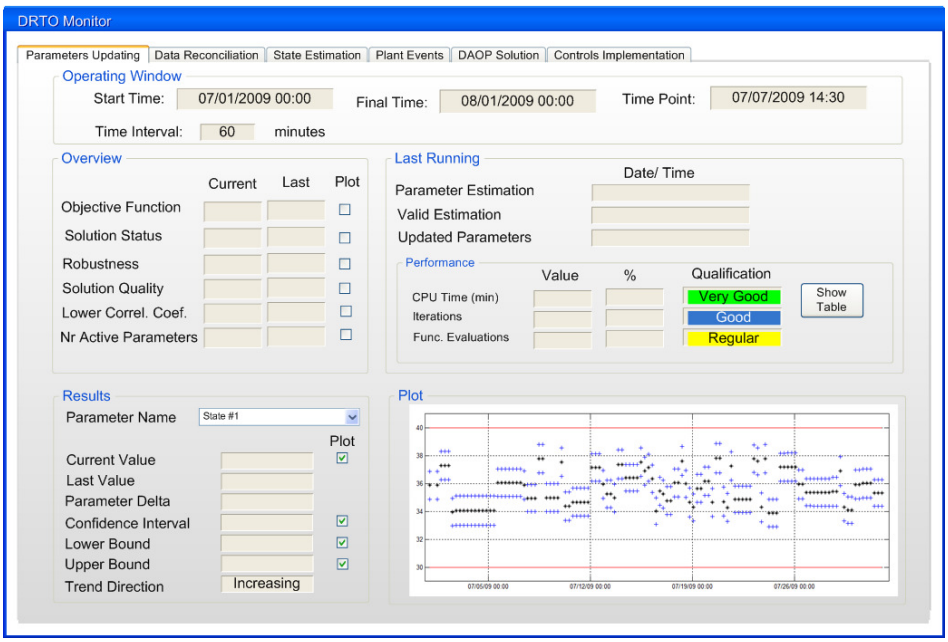


Figura J.3 – Tela de acompanhamento da atividade de atualização de parâmetros.

Na visão da rodada, deve-se acompanhar: as datas, horários e status das estimações de parâmetros (indica a sua freqüência de disparo e falhas); datas, horários das estimações de

parâmetros válidas (indica o seu sucesso); datas, horários das atualizações de parâmetros (indica a variabilidade das estimações); e os tempos gastos nas estimações (indica o seu desempenho);

Da mesma forma da seção anterior, o acompanhamento da atividade de reconciliação de dados é efetuado através da visualização dos indicadores específicos e de gráficos de tendência, como mostra a Figura J.4.

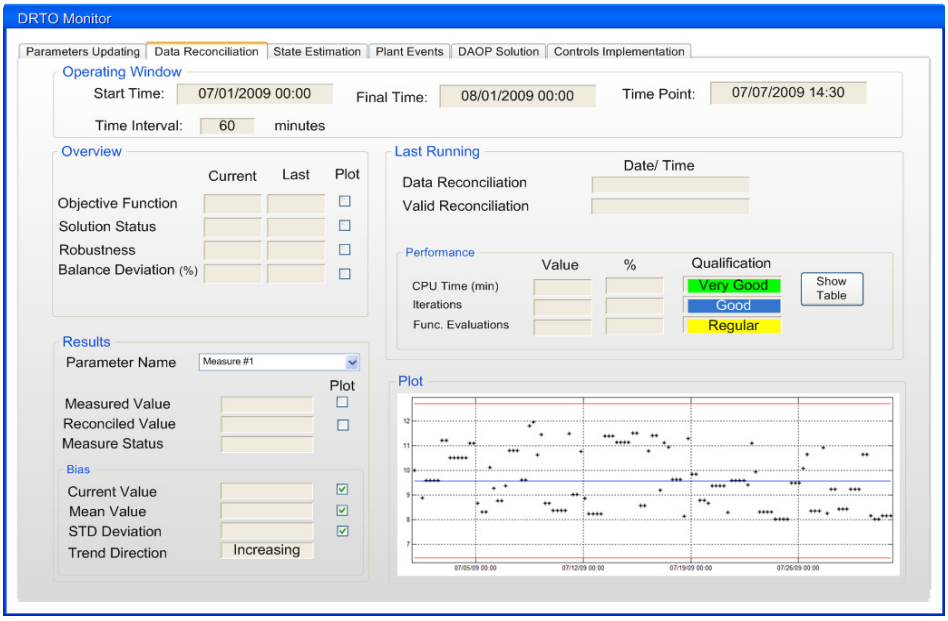


Figura J.4 – Tela de acompanhamento da atividade de reconciliação de dados.

Na visão geral da reconciliação, deve-se acompanhar: os valores históricos da função objetivo da reconciliação de dados (representa a diferença do previsto x experimental); desvios de balanços (% das diferenças no balanço material utilizando as vazões medidas) e a robustez da reconciliação (% problemas resolvidos). Quando a reconciliação for simultânea à estimação, deve-se utilizar a parcela da função objetivo referente aos desvios das medições.

Na visão da rodada, deve-se acompanhar: as datas, horários e status das reconciliações de dados (indica a sua freqüência de disparo e falhas); datas, horários das reconciliações válidas (indica o seu sucesso); e os tempos gastos nas reconciliações (indica o seu desempenho).

O acompanhamento da atividade de estimação de estados é efetuado através da visualização dos indicadores específicos e de gráficos de tendência, como mostra a Figura J.5.

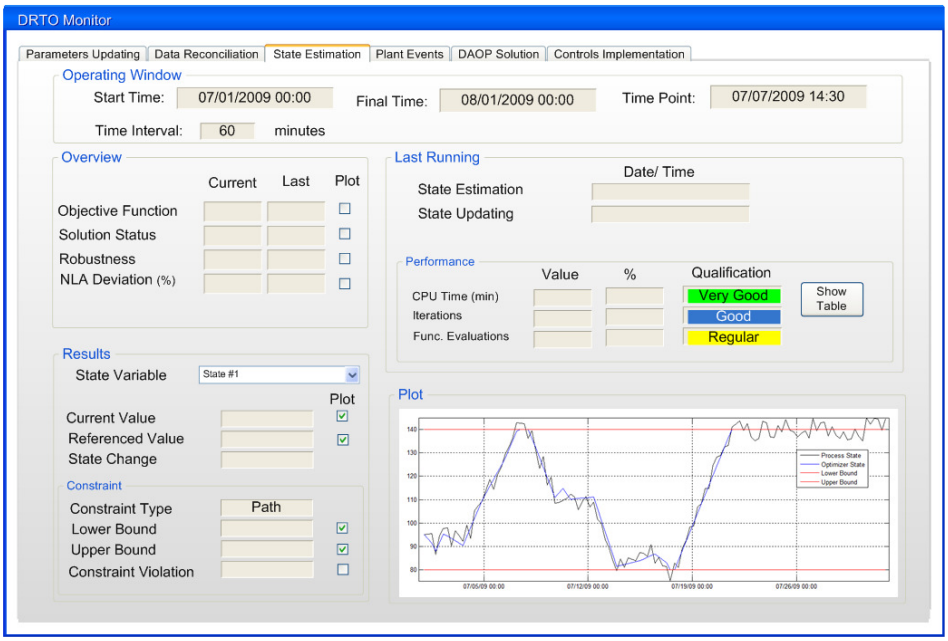


Figura J.5 – Tela de acompanhamento da atividade de estimação de estados.

Na visão da estimação de estados, deve-se acompanhar: os valores históricos da função objetivo da estimação de estados (representa a diferença do previsto x experimental); desvios entre a estimação e a solução do *NLA* (% das diferenças; indica divergências entre a observação do estimador e do otimizador) e a robustez da estimação (% problemas resolvidos).

Na visão da rodada, deve-se acompanhar: as datas, horários e status das estimações de estados (indica a sua freqüência de disparo e falhas); datas, horários das atualizações dos estados (indica a sua atualização); e os tempos gastos nas estimações (indica o seu desempenho);

Da mesma forma da seção anterior, o acompanhamento dos eventos que disparam o otimizador e o ajuste de parâmetros e reconciliação de dados é efetuado através da visualização dos indicadores específicos e de gráficos de tendência, como mostra a Figura J.6.

Na visão da rodada, deve-se acompanhar: as datas, horários dos disparos da otimização dinâmica (indica a sua freqüência de disparo); motivos dos disparos do otimizador (mudança da estrutura do *DAOP*, perturbação na planta, mudança na ordem de produção, não otimalidade); e otimalidade da receita (indica o seu sucesso na obtenção da política ótima de controle);

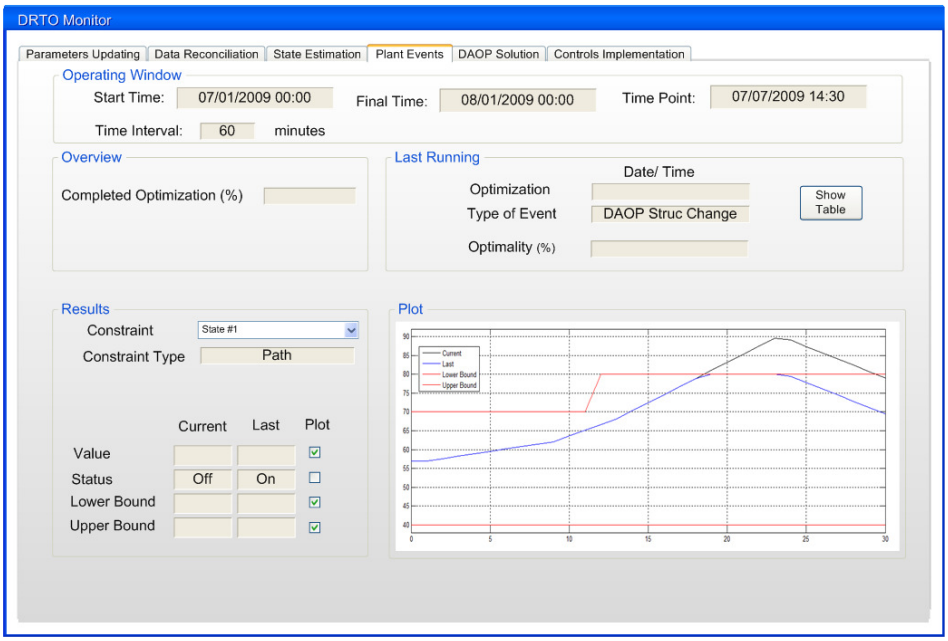


Figura J.6 – Tela de acompanhamento dos eventos da planta.

Da mesma forma da seção anterior, o acompanhamento da solução do problema de otimização dinâmica é efetuado através da visualização dos indicadores específicos e de gráficos de tendência, como mostra a Figura J.7.

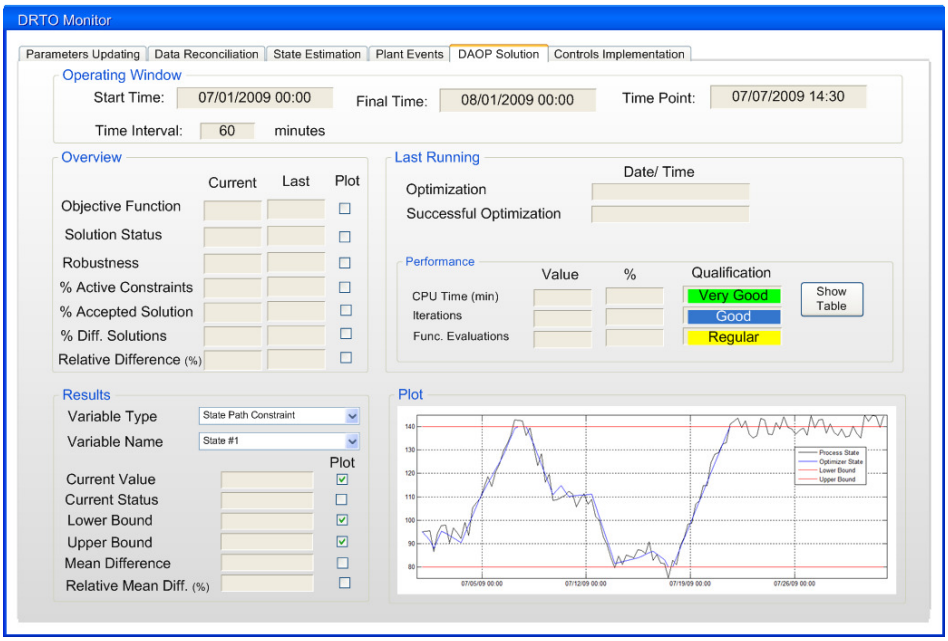


Figura J.7 – Tela de acompanhamento da atividade de otimização.

Na visão da atividade de otimização, deve-se acompanhar: os valores históricos da função objetivo; % das violações das restrições (indica a inviabilidade do problema); % das restrições ativas (indica as estruturas dos arcos da solução); % soluções aceitas (indica o nível de confiança da operação nos resultados do otimizador); evolução das últimas soluções - perfis de controle, estado, falha na rodada, tempo de *CPU*, ... (informações gerais da solução do otimizador); e discriminação entre soluções do otimizador - % de

soluções diferentes, perfis de controle distintos, diferenças relativas das soluções [%] (indica a freqüência e natureza das soluções distintas fornecidas pelo otimizador).

Na visão das implementações das ações de controle, deve-se acompanhar: os valores históricos das % das soluções bem sucedidas implementadas (indica a quantidade de soluções úteis); as conformidades dos perfis ótimos das variáveis de controle e de estado (indicam os níveis de rastreamento das soluções ótimas), como mostra a figura J.8.

Esta monitoração procura detalhar o indicador de conformidade gerencial, apresentando as informações para cada variável de controle e estado.

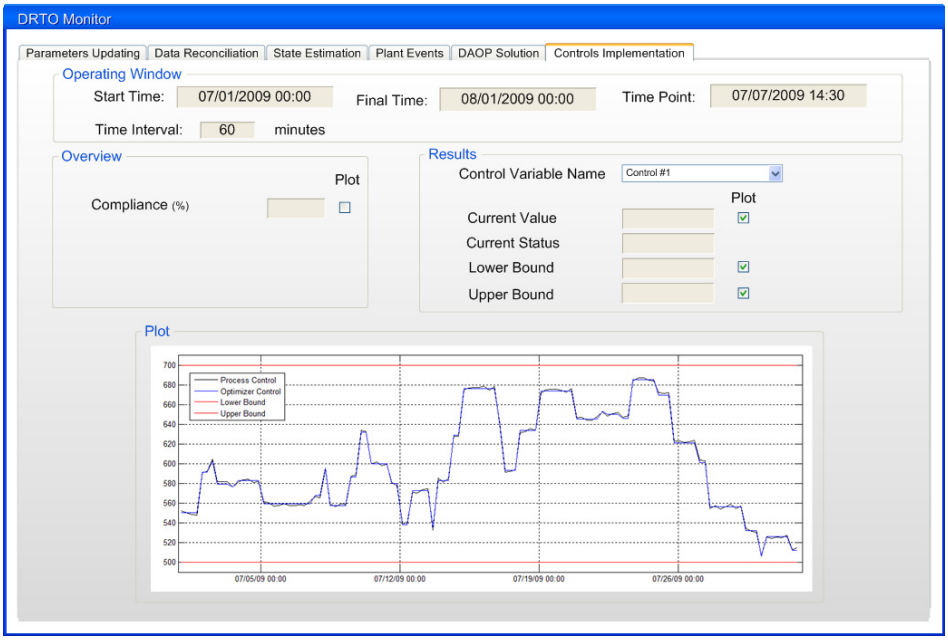


Figura J.8 – Tela de acompanhamento das implementações das ações de controle.

Apêndice K - Diagnóstico do DRTO

O diagnóstico do *DRTO* pode ser realizado através de análises de problemas em três níveis de detalhes (Figura K.1). No nível 1, tem-se uma visão geral do problema de otimização dinâmica, da solução do problema, dos parâmetros do otimizador e dos alarmes e eventos gerados pelo otimizador. No nível 2, têm-se os detalhes dos mesmos aspectos citados acima. E no nível 3, tem-se uma análise detalhada de cada tópico do processo de solução do problema de otimização.

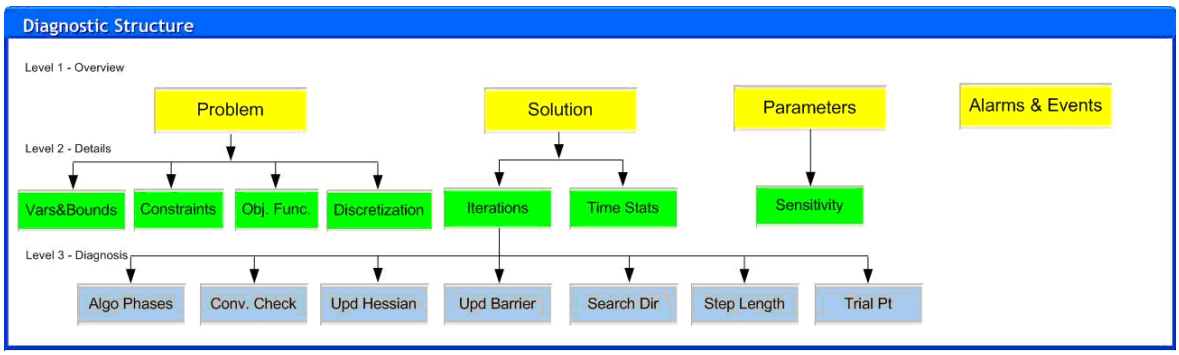


Figura K.1 – Estrutura do sistema de diagnóstico do *DRTO*.

Ao analisar o problema, obtém-se uma visão geral do problema, onde são informadas as dimensões globais das variáveis, restrições e graus de liberdade do problema de otimização. Estes tópicos são mostrados na Figura K.2.

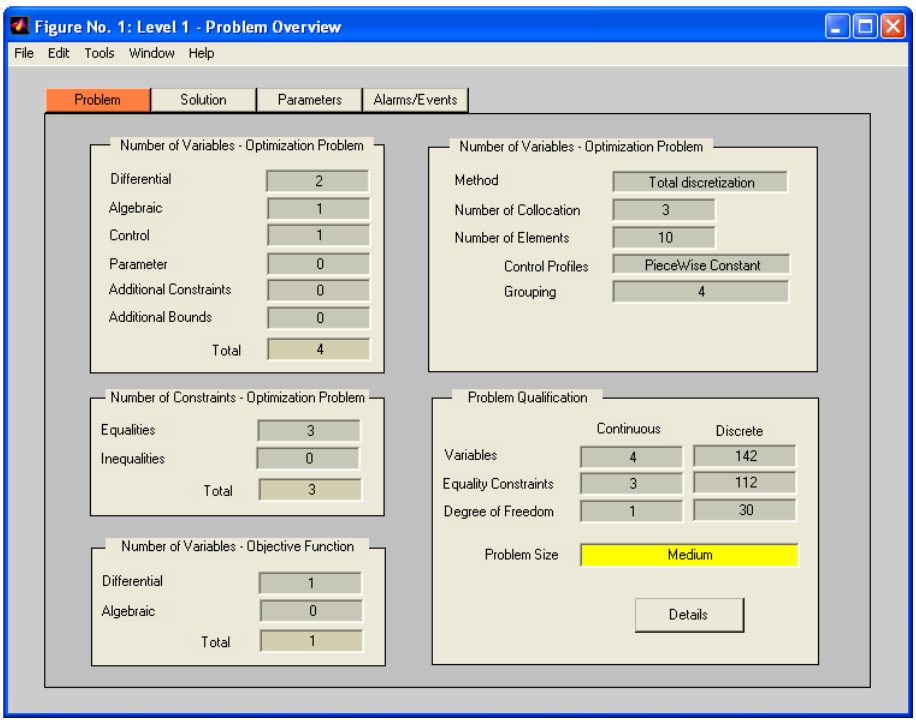


Figura K.2 – Visão geral do problema de otimização dinâmica.

Na tela (Figura K.3), têm-se três seções dando informações gerais sobre a solução do problema de otimização dinâmica.

Na primeira seção têm-se as informações do estado geral da solução do otimizador. Na segunda, podem ser vistos os valores iniciais e finais dos indicadores principais da solução do problema de otimização. Na terceira seção, são apresentados os indicadores gerais de desempenho do otimizador. E nas seções 2 e 3 são apresentados os resultados de uma análise qualitativa dos indicadores. Estas avaliações fornecem informações sobre a natureza do problema ocorrido com o otimizador e sua primeira análise. Se o usuário desejar mais informações, ele pode investigar o problema no nível 2 de detalhes.

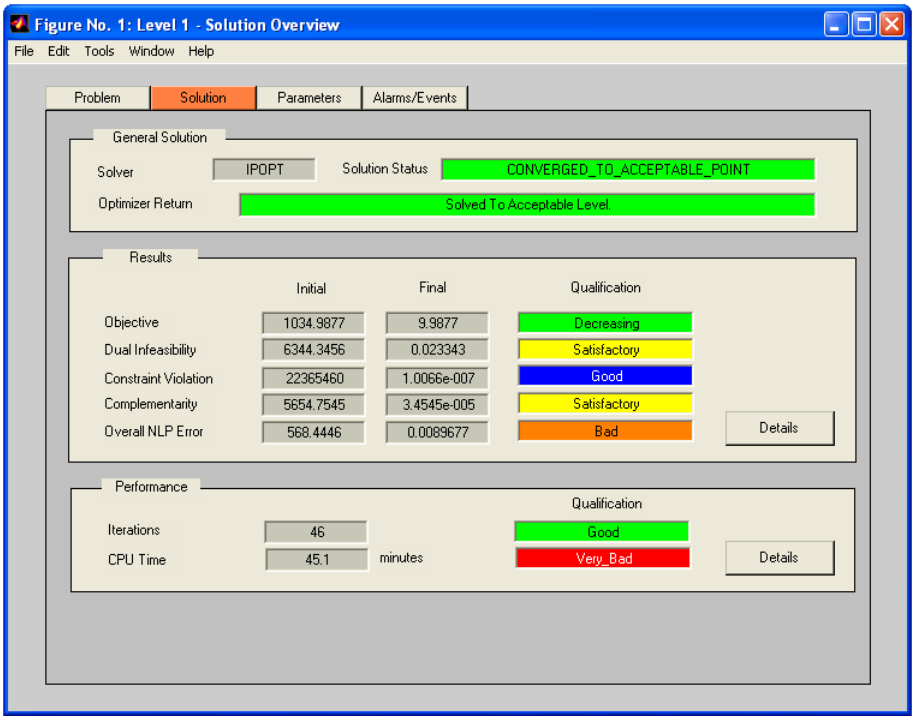


Figura K.3 – Visão geral da solução do problema de otimização dinâmica.

Nesta tela, têm-se três seções que fornecem informações gerais sobre os parâmetros do otimizador, como mostra a Figura K.4. Na seção geral da tela, o usuário pode selecionar o modo de solução de problema e respectivo solver usado na atividade. O solver disponível é apenas ***IPOPT*.**

Na análise de ***parâmetros***, o usuário deve primeiro selecionar a categoria dos parâmetros a serem listados. Os campos são os seguintes: Categoria – é a categoria de parâmetros do solver, cada categoria representa uma fase do algoritmo que executa uma tarefa específica. Quando o usuário escolhe uma categoria específica de parâmetros, a lista de parâmetros dessa categoria é apresentada.

Para a ***análise de sensibilidade***, quando o usuário seleciona um parâmetro específico, clicando na respectiva linha da tabela, o nome e o comportamento desse parâmetro aparecem na área de análise de sensibilidade. Se o usuário decidir realizar a análise de sensibilidade, é necessário apenas para clicar no botão executar localizado na parte inferior, após selecionar um parâmetro específico.

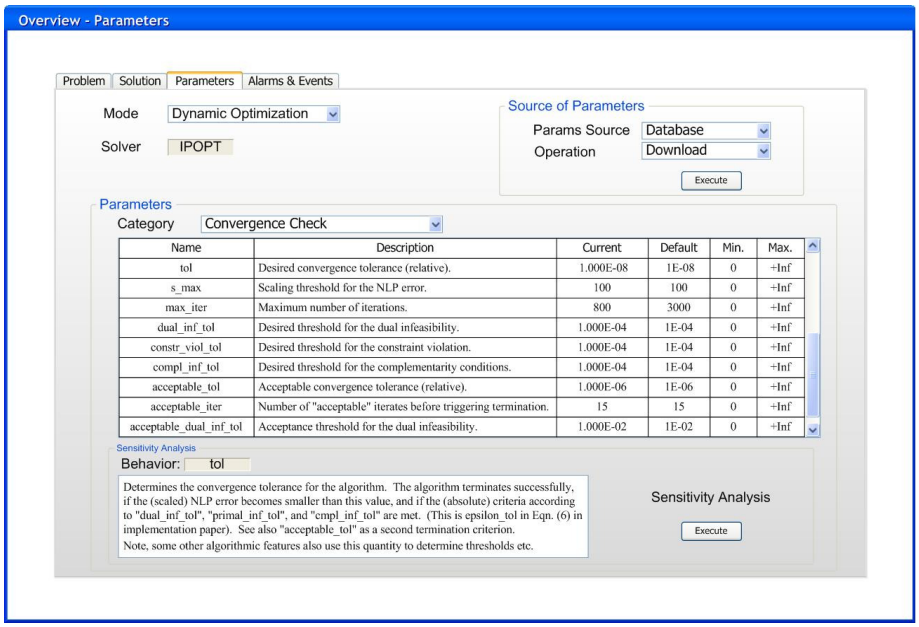


Figura K.4 – Visão geral dos parâmetros do otimizador.

Na visão geral dos alarmes e eventos, nós temos a lista resumida de alarmes e eventos gerados pelo sistema, conforme a Figura K.5. O usuário pode classificar ou filtrá-las. O usuário pode classificar as informações por severidade e filtrar-las por tarefa. Desta forma fica mais simples visualizar o que interessa no momento da análise.

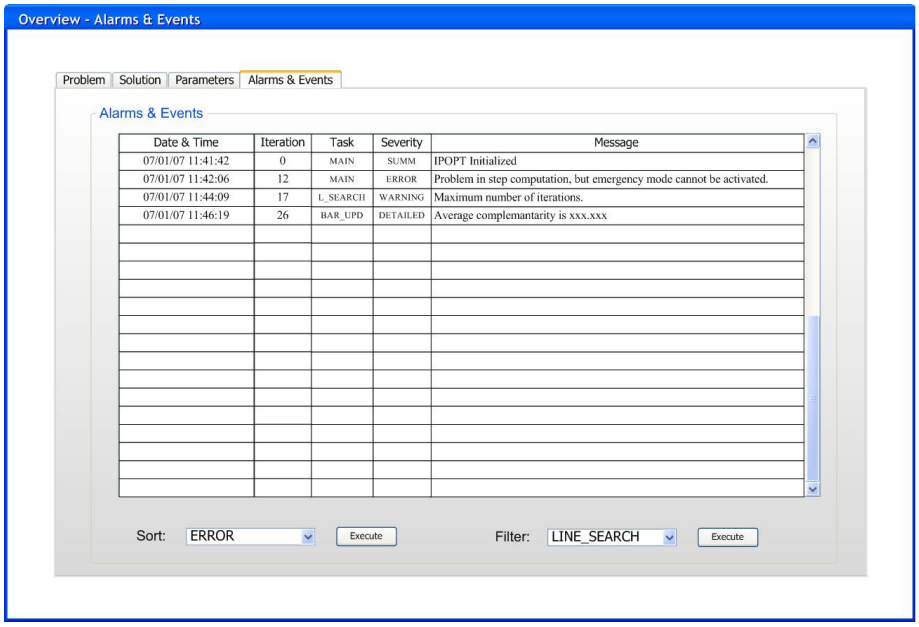


Figura K.5 – Visão geral das mensagens de alarmes e eventos do otimizador.

Estas listas de tarefas e de severidade baseiam-se nas categorias de impressão de mensagens do *IPOPT*.

Nos detalhes das dimensões do problema, são visualizadas informações detalhadas das dimensões do problema (*DAOP*), como mostra a Figura K.6. Aqui são apresentadas as

dimensões do problema contínuo e discreto. Sobre as ***variáveis***, são obtidas as informações sobre o número de variáveis diferenciais, algébricas, de controle, de parâmetros, de restrições e limites adicionais no problema contínuo e discreto. As qualificações das dimensões baseiam-se no tamanho do problema discreto. O critério é o mesmo da qualificação de tamanho problema aplicada para cada variável abaixo. Além disso, são apresentadas informações sobre o número de ***restrições de limites*** no problema contínuo e discreto. Neste caso, tem-se número total de restrições de limites, que é a soma de todos os limites do problema: variáveis com apenas limites inferiores ou superiores e variáveis com ambos os limites. Com estas informações, tem-se uma ordem de grandeza do tamanho do problema resolvido e das expectativas de certas dificuldades para resolvê-lo. Se houver um problema com um número muito grande ou ultra grande número de variáveis e limites, o usuário pode esperar alguns problemas com a alocação de memória e desempenho, sendo necessário mitigar este problema com a melhoria do sistema através de uma parametrização mais adequada dos algoritmos de otimização utilizado.

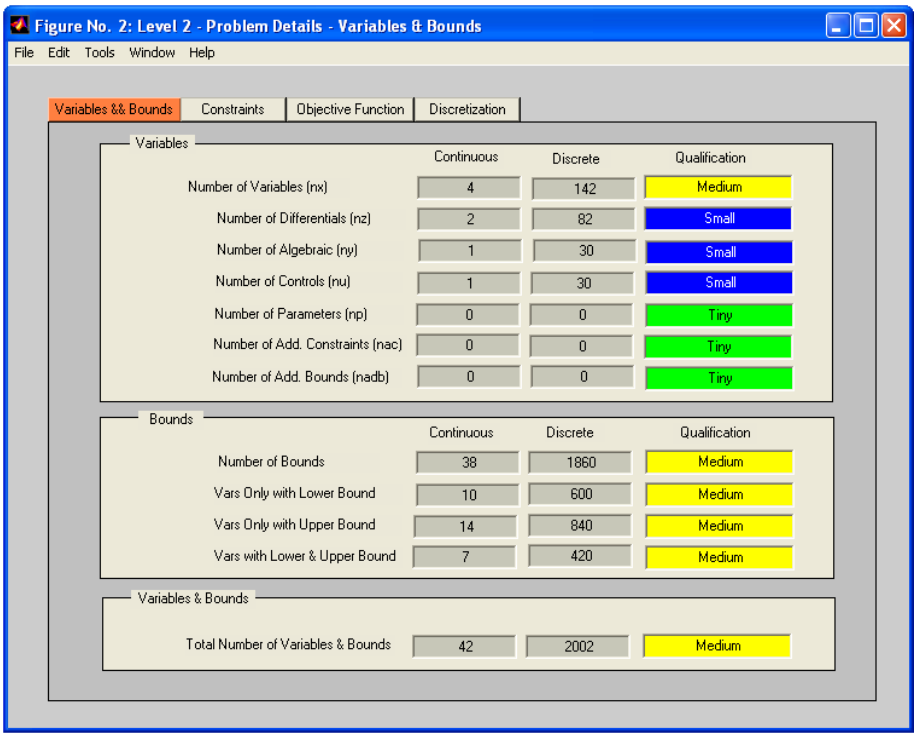


Figura K.6 – Detalhes das dimensões das variáveis e limites do problema (*DAOP*).

Na análise das iterações, tem-se uma tabela das principais informações de cada iteração do algoritmo. As principais informações, no caso do *IPOPT*, são o seguinte: número da iteração com o algoritmo; tipo de iteração - string que relatar algo anormal nessa iteração; valor sem escala da função objetivo; inviabilidade primal e dual; Lg(μ) – logaritmo 10 de μ (para obter a ordem de grandeza do (ex.: Lg(μ) =-1 significa que $\mu = 10^{-1}$); ||***d***|| – a norma da direção de busca ($\|d\| = \max\left(|\delta x|_\infty, |\delta s|_\infty\right)$); Lg(rg) – logaritmo 10 do tamanho de regularização (δx); alfa primal – valor α das variáveis primais (x); tipo de tamanho de passo na etapa na busca em linha; alfa dual – valor de α das variáveis duais (s); número de tentativas de busca em linha para obter a direção de busca; e do tipo de busca em linha – no filtro de busca em linha.

Se o usuário desejar obter mais detalhes de uma iteração, ele pode clicar na linha da iteração desejada na tabela resumo das iterações (Figura K.7). Você pode encontrar aqui o número e a qualificação da iteração requisitada. Depois de selecionar a iteração, o usuário pode obter informações detalhadas, clicando em detalhes localizados na parte inferior na área de detalhes da iteração.

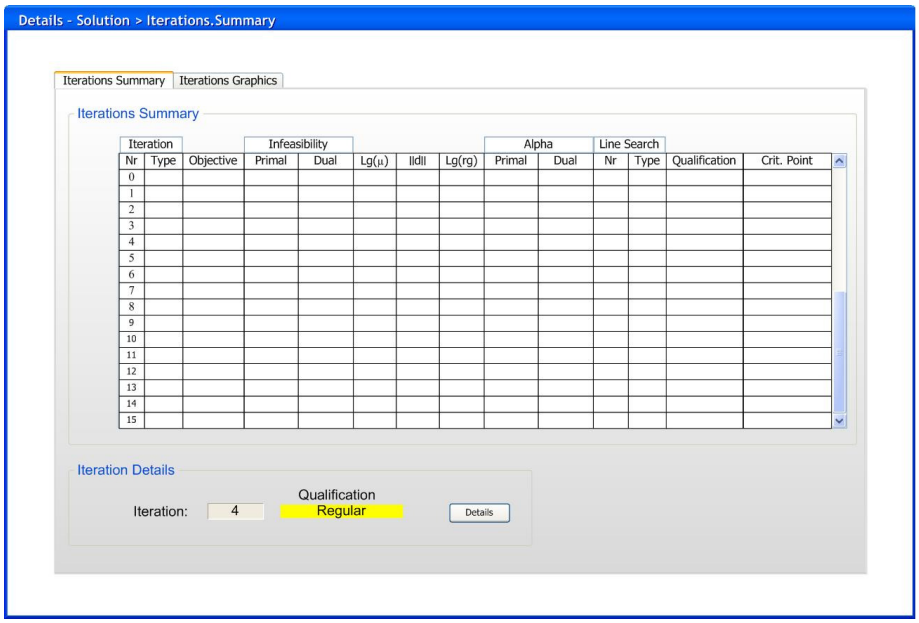


Figura K.7 – Resumo das iterações do otimizador.

Quando o usuário precisar analisar mais profundamente as iterações, ele pode usar alguns gráficos para visualizar as tendências dessas iterações. Esta seção permite-lhes apresentar na forma gráfica qualquer variável da tabela de resumo da iteração (vide Figura K.8). O usuário pode escolher uma variável para o eixo x e até duas variáveis para o eixo y.

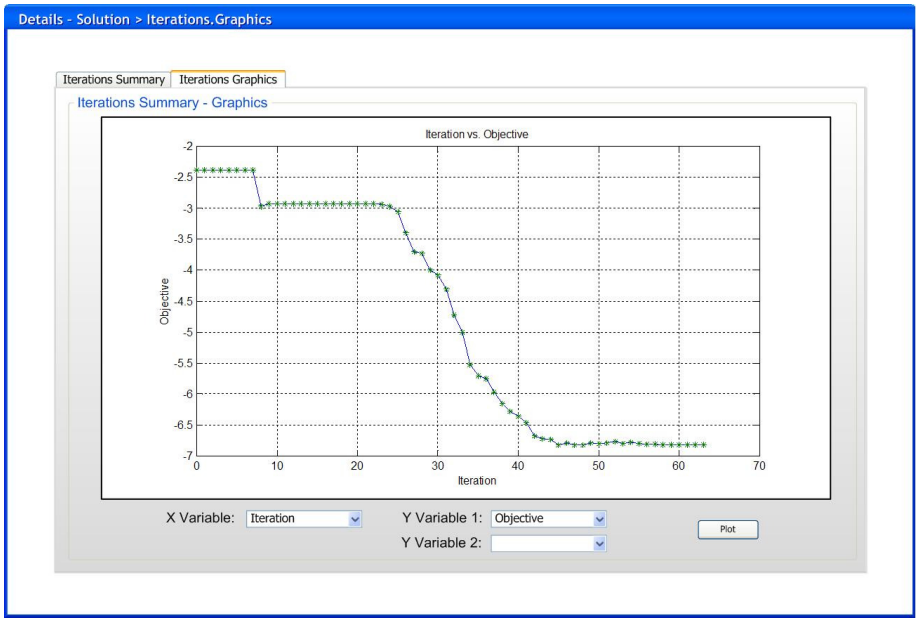


Figura K.8 – Gráfico de itens das iterações do otimizador.

Na seção de análise de desempenho do otimizador, são apresentadas as informações de tempo de *CPU* gasto pelo sistema durante o cálculo de solução. Eles são divididos em dois grupos principais: desempenho nas avaliações de funções e desempenho do solver de otimização (Figura K.9).

Quando não for possível melhorá-lo por métodos numéricos, pode-se recorrer ao processamento paralelo ou alterando o processador da *CPU* para que o desempenho atinja um nível prático para aplicações em tempo real.

Na seção de desempenho, são visualizadas as informações do número de avaliação de funções, o tempo total gasto com essas avaliações, porcentagem do tempo gasto para cada tipo de função, assim como a qualificação deste desempenho. Os tipos de funções envolvidas no cálculo de desempenho são a função objetivo e seu gradiente, os resíduos das restrições de igualdade e desigualdade e suas Jacobianas e Hessiana da Lagrangeana do problema. Estas estatísticas de tempo incluem os seguintes fatores: número de avaliações da função, total de tempo de *CPU* gasto em sua avaliação, percentagem do tempo de *CPU* para a sua avaliação sobre total de *CPU* para a avaliação da função.

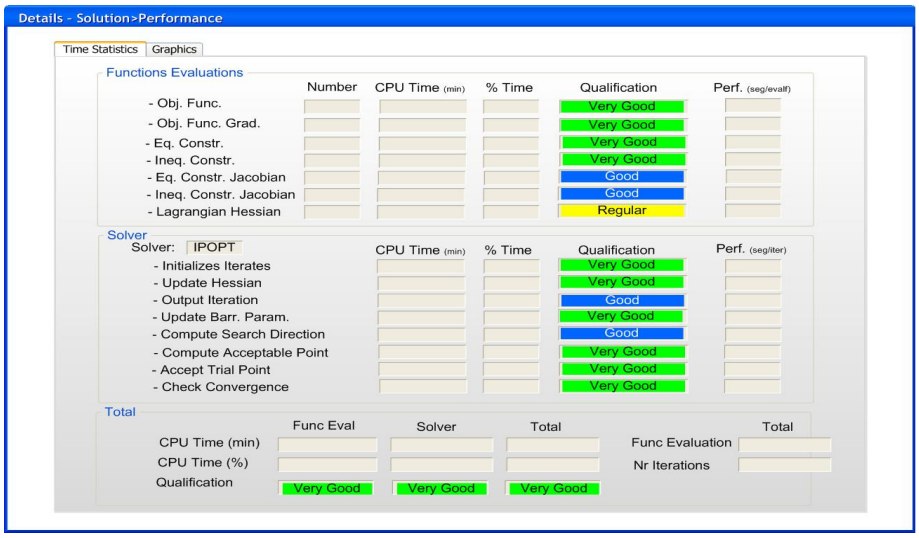


Figura K.9 – Detalhes do desempenho do otimizador.

A avaliação de desempenho do sistema pode ser realizada na forma de gráficos de perfil de desempenho, utilizando a metodologia de Dolan e Moré, 2001 (vide Figura K.10). Este gráfico mostra o perfil de um atributo da tela de desempenho. Para isto, o sistema armazena os dados históricos para todos os desempenhos. Os atributos que podem ser avaliados são os seguintes: desempenho total do sistema de *DRTO*, de todas as avaliações de funções, e do solver.

Ao selecionar o valor e o número de pontos históricos, o usuário pode traçar clicando no fundo correspondente. O eixo x (τ) é o indicador de desempenho (tempo de *CPU*) e $y(\rho)$ é a probabilidade de rodada atual vai ter sobre todos os outros casos.

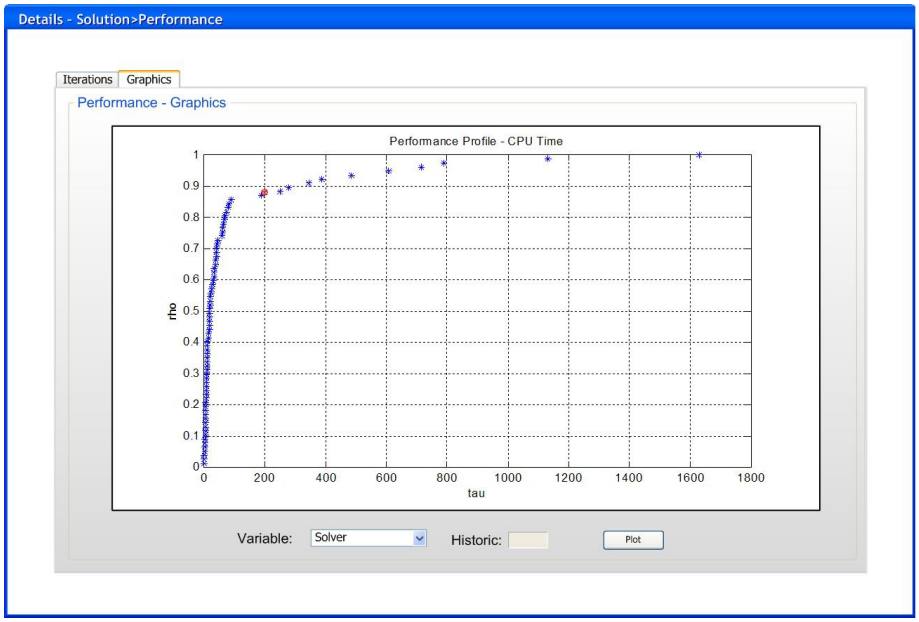


Figura K.10 – Gráficos de perfil de desempenho do otimizador.

Outra atividade importante é a análise de sensibilidade dos algoritmos do otimizador. Esta funcionalidade da ferramenta de diagnóstico visa o projeto e realização da análise de sensibilidade dos parâmetros de sintonia da *DRTO* (vide Figura K.11). Nesta seção são visualizadas as mesmas informações apresentadas na lista de parâmetros, adicionando os valores possíveis, quando for o caso. Com base nessas informações, o usuário pode projetar a análise de sensibilidade do parâmetro do otimizador selecionado. Neste projeto, o usuário precisa definir: o número de estudos de caso, o nome do indicador chave escolhido. O *KPI* (Indicador chave de desempenho - *Key Performance Indicator*) é uma das variáveis apresentadas no resumo da iteração ou na tela de desempenho. O projeto de experimentos é realizado usando uma tabela onde se preenche os valores do parâmetro a ser estudado e executar esta operação. Já, a análise de desempenho dos resultados pode ser efetuada utilizando a mesma tabela ou na forma gráfica.

Com base na análise dos resultados, o usuário pode escolher um melhor valor para o parâmetro de sintonia selecionado, e exportar o parâmetro para arquivo ou banco de dados.

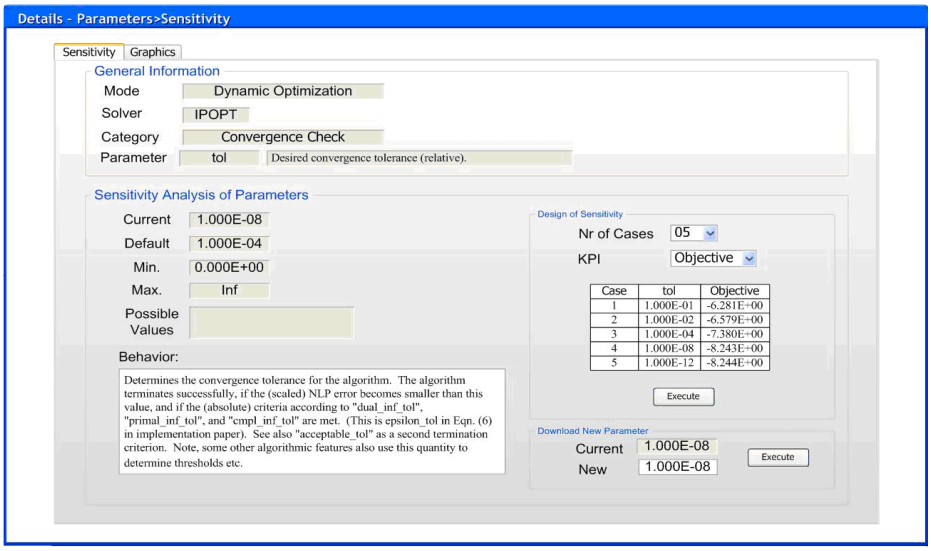


Figura K.11 – Análise de sensibilidade dos parâmetros de sintonia do otimizador.

Quando o usuário executa a análise de sensibilidade, o gráfico de sensibilidade pode ajudar a identificar o melhor valor do parâmetro de sintonia, com base no *KPI* escolhido como critério de desempenho (vide Figura K.12). No eixo X é colocado o parâmetro de sintonia estudado e no eixo Y o *KPI* usado para medir o desempenho. Para facilitar a análise dos resultados do experimento, os mesmos podem ser apresentados em diferentes tipos de escalas (*linear, semilogX, semilogY* ou *loglog*).

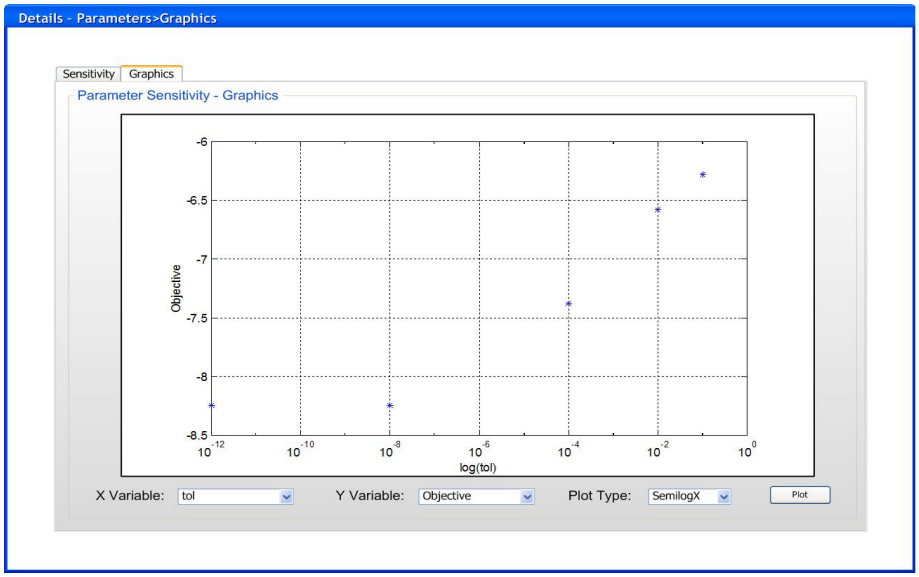


Figura K.12 – Gráficos de análise de sensibilidade dos parâmetros do otimizador.

Este nível de diagnóstico tem o objetivo de fornecer ao usuário mais detalhes sobre uma iteração específica do otimizador, para permitir que o usuário execute diagnóstico nessa iteração. Estas informações permitem que o se identifique a parte do algoritmo que apresentou problemas. Nesta análise, são visualizados os seguintes aspectos do algoritmo de *NLP* (no caso do *IPOPT*): fases do programa, verificação da convergência, atualização da Hessiana, atualização do parâmetro de barreira, direção de busca, comprimento do passo e definição do ponto tentativa.

As informações possíveis são as seguintes: valor do parâmetro barreira (μ), constante do filtro de busca em linha (τ), máximo valor absoluto das variáveis diferencial, algébrica, de controle e parâmetros, máximo valor absoluto das variáveis de folga relacionadas às restrições de desigualdade, valor máximo absoluto de multiplicadores de Lagrange relacionados às restrições de igualdade $c(x)$ (λ_c) e desigualdade $d(x)$ (λ_d), máximo valor absoluto da condição de complementaridade relacionada aos limites inferiores e superiores das variáveis de estado, controle e parâmetros x (Z_L e Z_U), dos limites inferiores e superiores das variáveis de folga s (V_L e V_U).

O usuário pode obter mais detalhes da solução do *NLP* através da seleção de uma iteração específica para ajudá-lo a diagnosticar o problema ocorrido, como mostra a Figura K.13. Nesta seção são apresentados o resumo da iteração, e as posições de algumas variáveis no início e no final da iteração atual.

Na seção onde estão os valores atuais do *NLP*, têm-se os mesmos atributos presentes na seção de visão geral dos resultados. Desta forma, o usuário poderá observar a evolução dos critérios de convergência do *NLP*, através da comparação com os valores da última iteração, primeira iteração ou alguma iteração intermediária. Esta análise é acompanhada da qualificação das direções desses atributos. Desta forma têm-se os valores: decrescendo (delta negativo), estável (delta dentro da respectiva tolerância) e crescendo (delta positivo). Uma condição de cor é apresentada de acordo com a direção certa (Verde – direção certa, Amarelo – estável ou Vermelho – direção errada). A idéia principal deste modelo é para dar uma idéia do movimento do otimizador. Ele pode ser útil definir se o otimizador está seguindo na direção certa.

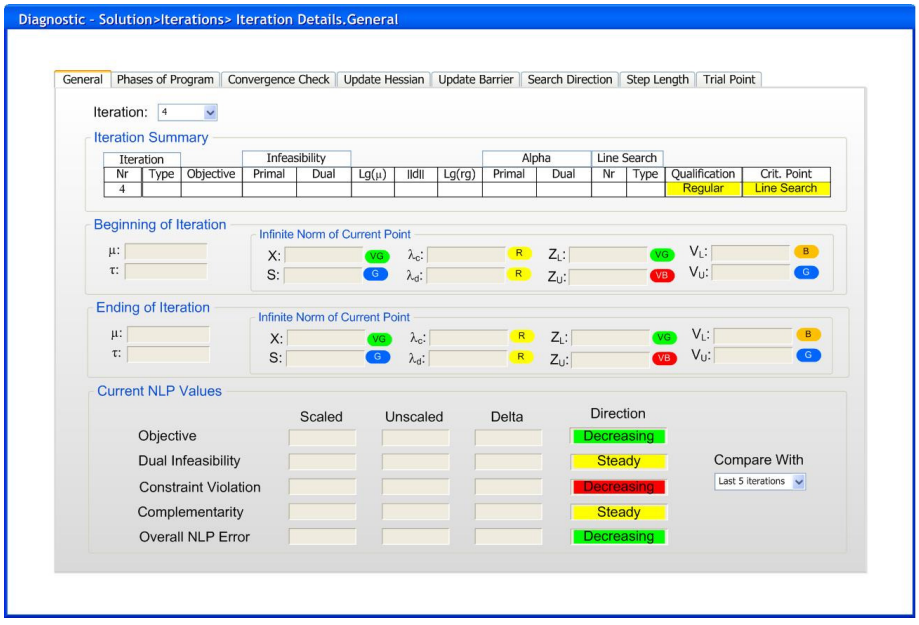


Figura K.13 – Diagnóstico das iterações do otimizador.

As fases do algoritmo *IPOPT* são os seguintes: inicialização das variáveis & parâmetros, verificação da convergência & definir iteração, atualização da Hessiana, atualização do parâmetro de barreira, computação da direção de busca, cálculo do comprimento do passo, computação do ponto tentativa, finalização do algoritmo. A cor é exibida de acordo com

para o estado seguinte: Cinza – tarefas não executadas; Verde – status normal; Amarelo – tarefa concluída com sucesso não, mas não problemático; Laranja – tarefa concluída com sucesso não, problemático, mas aceitável; Vermelho – tarefa concluída com falha.

A seção onde se visualizam as fases do algoritmo de otimização fornece ao usuário uma visão geral das suas tarefas (vide Figura K.14). Os blocos neste diagrama fornecem uma idéia de que partes do algoritmo de tem problemas. Esta informação se baseia o status de retorno de cada tarefa. Este ponto de vista, o usuário pode identificar diretamente a tarefa problemática e navegar diretamente para essa pasta.

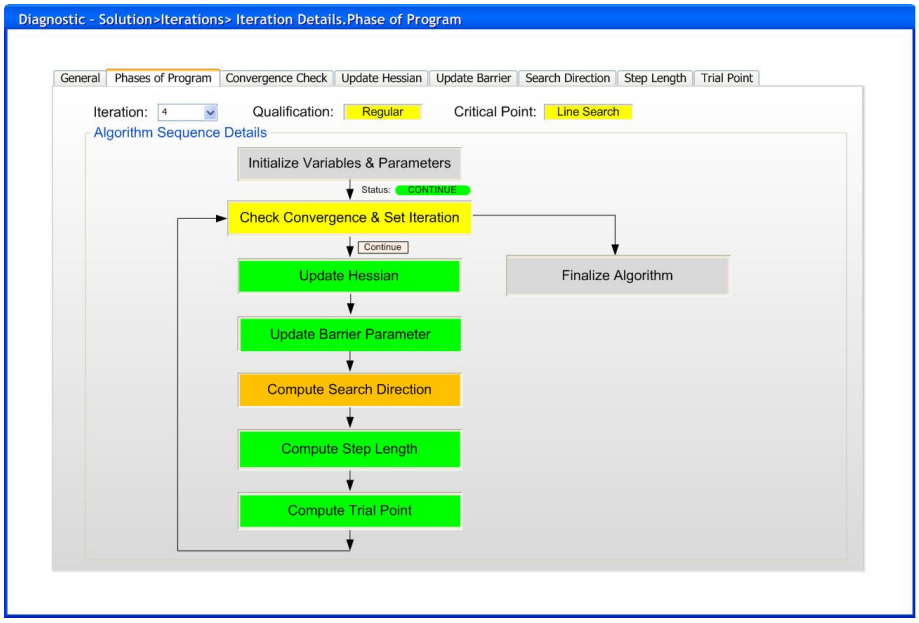


Figura K.14 – Visão dos estados das tarefas do algoritmo.

Na visão dos detalhes da verificação da convergência pelo algoritmo (Vide Figura K.15), as informações sobre os critérios de convergência do algoritmo são apresentadas na forma de erro do *NLP*, inviabilidade dual, violação de restrição e complementaridade. Com estas informações, é possível avaliar a convergência do algoritmo, e detectar se o mesmo está tendendo a divergir ou se a iteração é aceitável. O *DAOP* é considerado convergido se todos os critérios são aceitos com base nas suas respectivas tolerâncias.

A violação das restrições é definida como o número máximo de três normas infinitas. Eles são a norma infinita de restrições de igualdade $c(x)$ ($Max(|c(x)|)$) e a norma infinita das restrições de desigualdade relacionadas com os limites inferiores ($Max(|d_{viol}^{L}|)$) e superiores ($Max(|d_{viol}^{U}|)$) do sistema ($d(x) - d^{L\ ou\ U}$). A inviabilidade dual é o valor máximo de duas normas infinitas. Uma é a norma infinita do gradiente da Lagrange sem escala da função objetivo relacionados com a variável primal x ($Max(|\nabla_x L|)$) e outro com o mesmo gradiente relacionados com o variável de folga s ($Max(|\nabla_s L|)$). A complementaridade é o valor máximo de quatro normas infinitas. Eles são as condições de complementaridade relacionadas com os limites inferiores e superiores da variável primal x ($Max(|S_{xL}Z_L|)$ e $Max(|S_{xU}Z_U|)$) e a variável de folga s ($Max(|S_{sL}Z_L|)$ e $Max(|S_{sU}Z_U|)$).

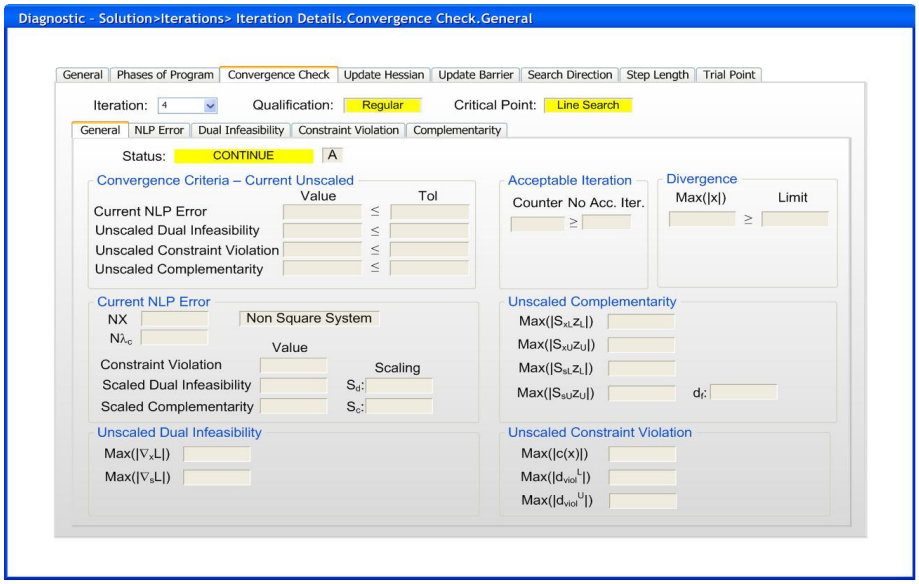


Figura K.15 – Diagnóstico da verificação da convergência.

Uma iteração aceitável quando o otimizador finalizar com uma solução viável, mas não ótima. Se o otimizador obtiver uma solução viável sem melhoria da função objetivo durante um determinado número de iterações aceitáveis, ele é paralisado com um estado CONVERGIU PARA UM PONTO ACEITÁVEL. A detecção de divergência é definida quando a norma infinita da variável primal (*Max(|x|)*) explode, ou seja, torna-se maior que o limite estabelecido pelo usuário. Quando esta situação acontece, o algoritmo pára com o status ITERAÇÕES DIVERGENTES.

Outro aspecto da avaliação da convergência é a análise o erro global do *NLP* (vide Figura K.16). Esta seção repete algumas informações, tal como foi apresentado na aba em geral, e adicionando os números, nomes e tempo dessas variáveis correspondentes. O tempo corresponde ao instante tempo discreto (no caso, referente ao ponto de colocação em um elemento finito). Com essas informações, o usuário pode identificar a variável e o momento em que ocorre a maior violação da restrição, a inviabilidade dual e a condição de complementaridade. Além disso, é possível ver as distribuições desses erros através do gráfico de Pareto, que permite concluir se a violação do critério de convergência está localizada ou distribuída em todo o seu domínio. Os conjuntos são baseados nos mesmos critérios da qualificação de erro com ambos os lados, valores positivos e negativos. Se o usuário precisar identificar melhor a causa do problema ocorrido, ele pode verificar esses erros na forma de tabela clicando no fundo mais detalhes.

Esta tela fornece uma idéia de qual tipo de variável está causando desvios mais elevados nas condições de complementaridade. Se o usuário quiser de obter mais detalhes sobre a condição de complementaridade, ele pode verificar cada parcela da condição de complementaridade e localizar o problema.

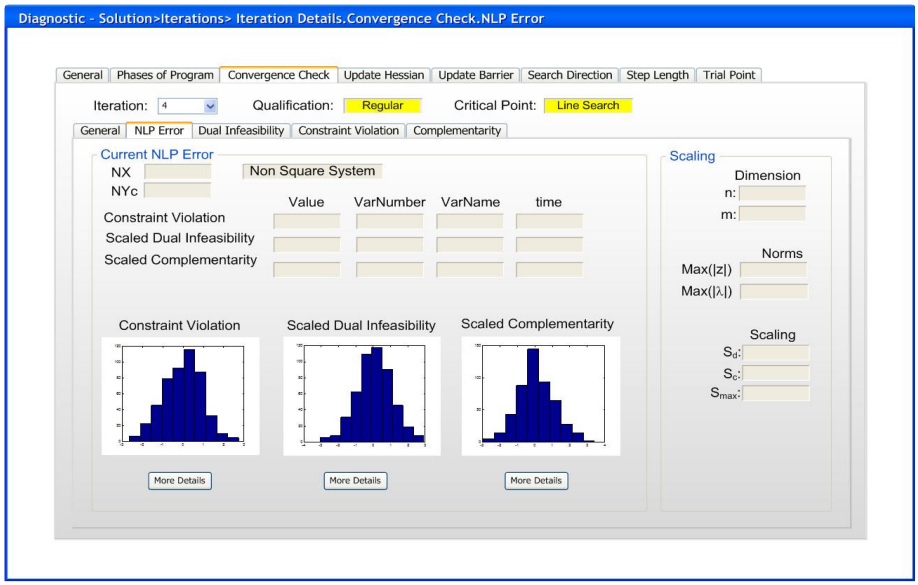


Figura K.16 – Diagnóstico da verificação da convergência do erro do *NLP*.

Se o usuário desejar obter mais detalhes sobre os critérios de convergência (as violações de restrições, inviabilidade dual e das condições de complementaridade), o mesmo pode clicar no botão "*More Details*". Neste caso, a ferramenta de diagnóstico fornece ao usuário alguns detalhes mais profundos sobre os perfis desses critérios e seus efeitos na convergência de erros de *NLP*. O objetivo desta visão é fornecer ao usuário estas informações de cada variável a cada intervalo de tempo do problema discreto. As colunas correspondem ao intervalo de tempo em termos de ponto de colocação e de elementos finitos. E as linhas são os resíduos das restrições, inviabilidades e complementaridade. Estas matrizes fornecem ao usuário os perfis exatos dos critérios citados acima. E com essas informações, o usuário pode analisar esses perfis e localizar os momentos nos quais cada tipo de variável que está causando as maiores violações dos critérios (vide Figura K.17). Pode ser necessário associar esta análise com outras informações de diagnóstico para identificar as causa do problema.

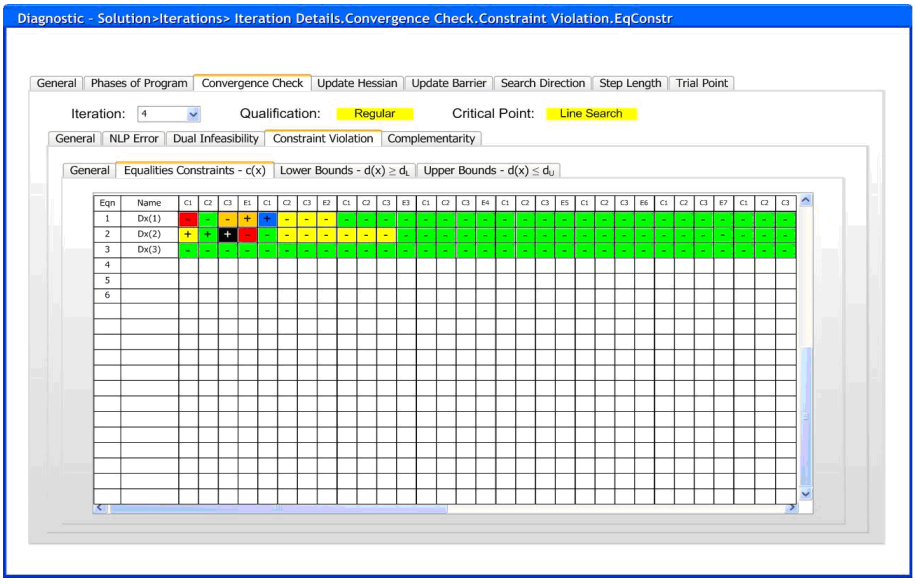


Figura K.17 – Visualização das violações das restrições ao longo do tempo.

O usuário também pode visualizar alguns detalhes sobre a inviabilidade dual na convergência de erros de *NLP*. O objetivo desta tela é fornecer ao usuário mais informações sobre inviabilidade dual. Podemos localizar o número e os nomes das variáveis que causam mais inviabilidade. Também instante de tempo referente ao problema de otimização dinâmica. Além de que se pode analisar a distribuição da inviabilidade dual por meio de gráfico de Pareto. Esta tela dá uma idéia de que tipo de variável que está causando valores mais altos. Se o usuário desejar obter mais detalhes sobre as inviabilidades duais, ele pode verificar em cada parte os gradientes e localizar o problema (vide Figura K.18).

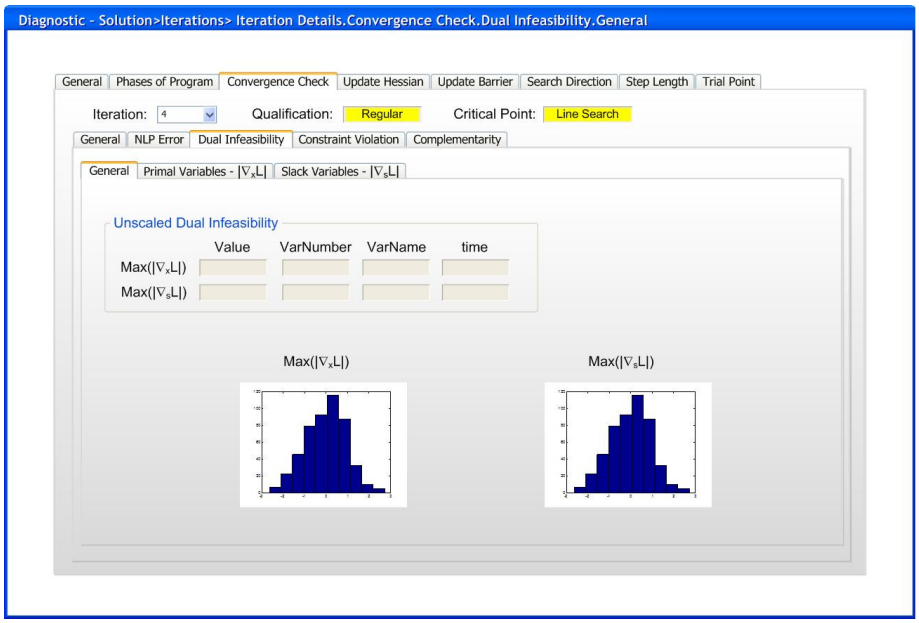


Figura K.18 – Visualização da distribuição da inviabilidade dual.

Nesta seção podem ser obtidos alguns detalhes sobre a violação de restrições na convergência de erros de *NLP*. O objetivo desta seção é fornecer ao usuário mais informações sobre as violações das restrições. Pode-se localizar o número e os nomes das variáveis que violam mais as restrições. Também é fornecido o tempo referente ao problema de otimização dinâmica contínuo, além de poder analisar a distribuição da violação das restrições por meio de gráfico de Pareto (vide Figura K.19).

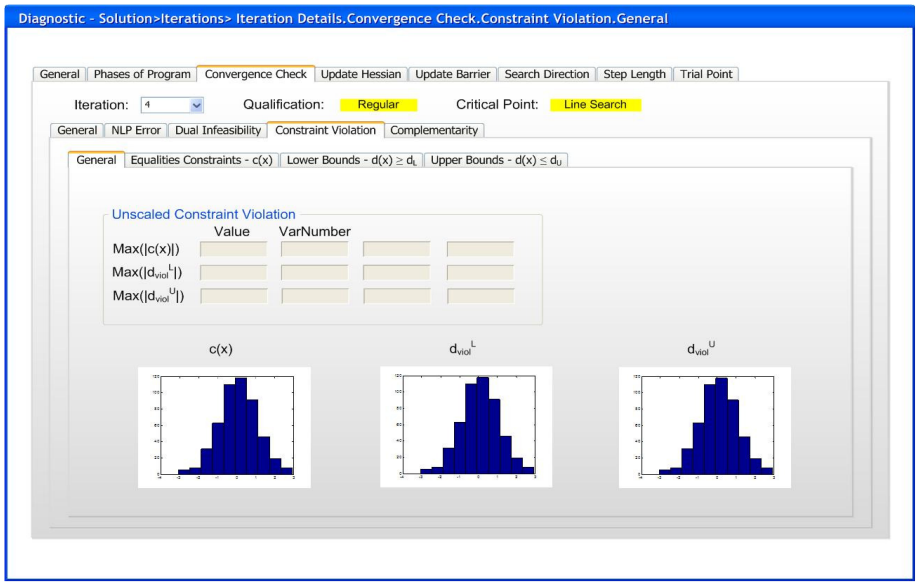


Figura K.19 – Visualização da distribuição das violações das restrições.

Esta seção apresenta alguns detalhes sobre a condição de complementaridade. Da mesma forma que o caso anterior, pode-se localizar o número e os nomes das variáveis que causam mais violações nas condições de complementaridade. E também fornece o tempo referente ao problema de otimização dinâmica. Além de poder analisar a distribuição da complementaridade por meio de gráfico de Pareto (vide Figura K.20).

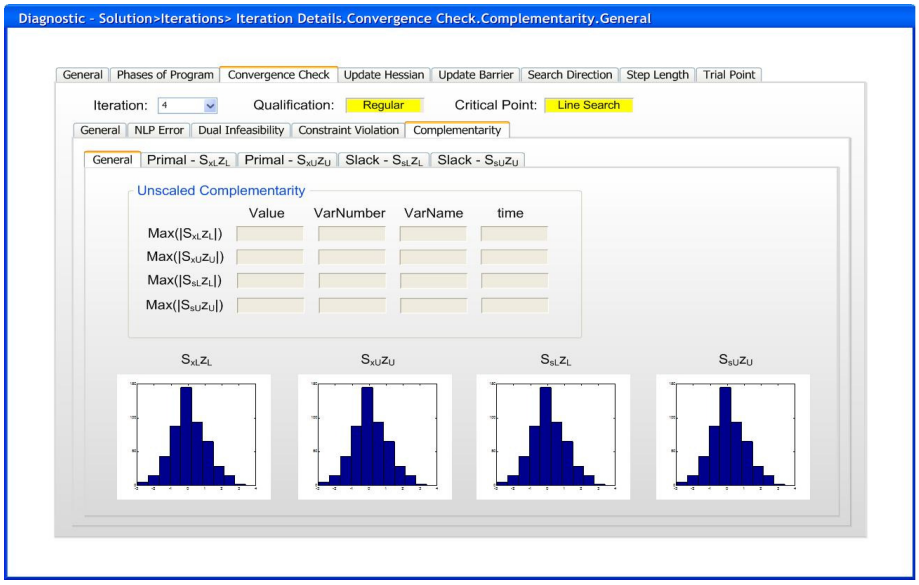


Figura K.20 – Visualização da distribuição das condições de complementaridade.

A atualização do parâmetro de barreira é uma tarefa importante do algoritmo. Este parâmetro controla a aproximação da solução para as restrições do problema e controla o progresso da busca em linha. Quando o passo se torna muito pequeno, o otimizador apresenta dificuldades em avançar na busca do ponto ótimo. Este parâmetro é reduzido drasticamente quando o problema se torna inviável. A evolução deste parâmetro, iteração a iteração indica a capacidade avanço do otimizador na busca da solução. Este fator pode

afetar diretamente o passo do otimizador. Uma pequena visão desta análise pode ser vista na Figura K.21.

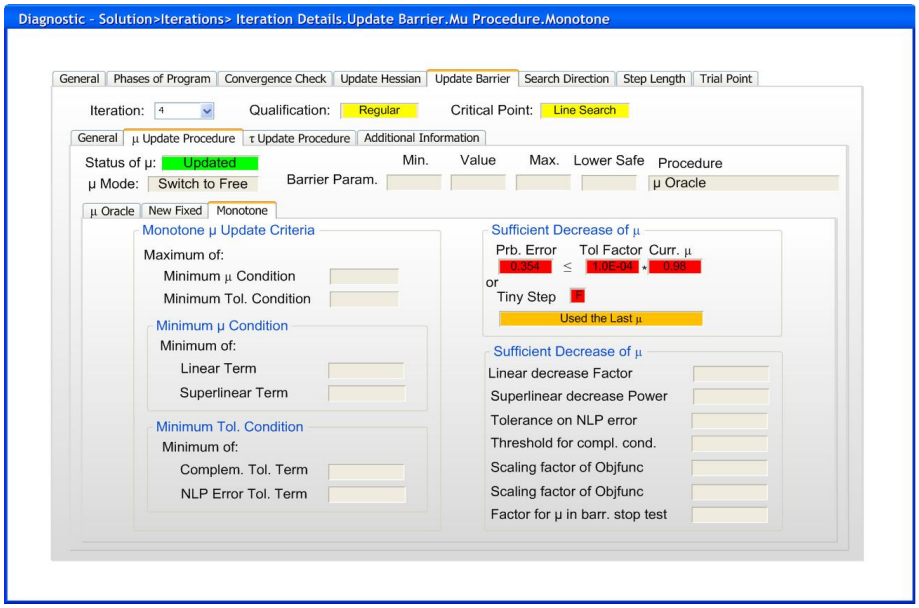


Figura K.21 – Diagnóstico da atualização do parâmetro de barreira do *IPOPT*.

A direção de busca consiste da solução do sistema resultante do desdobramento das condições de otimalidade do problema. Esta solução é obtida quando os resíduos do lado esquerdo do sistema de equações (deltas das variáveis de decisão) são zerados. Esta análise consiste basicamente em acompanhar estes resíduos. Estas informações serão utilizadas no filtro de busca em linha para verificar se o ponto proposto é aceitável. Desta forma, inspeciona-se o valor máximo da norma 2 de cada tipo de variável envolvida no problema (*x* - variáveis primais, *s* - variáveis de folga, *c* & *d* - violações das restrições primais e duais, *zL* & *zU* - limites superior e inferior das variáveis primais e *vL* & *vU* - limites superior e inferior das variáveis duais). A Figura K.22 apresenta uma visão do sistema primal-dual.

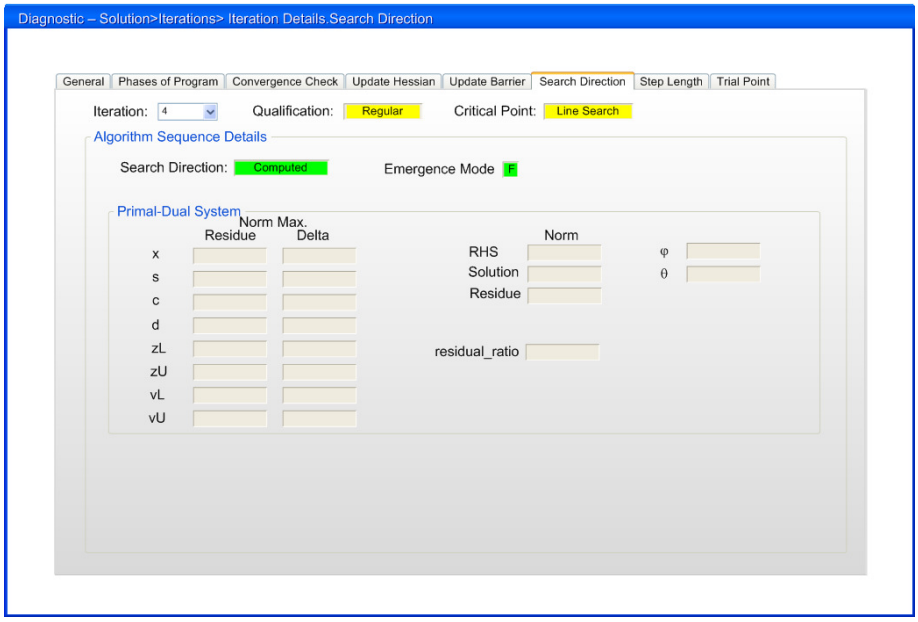


Figura K.22 – Diagnóstico do cômputo da direção de busca do *IPOPT*.

Na definição do tamanho do passo, controla-se a viabilidade do avanço da solução através do uso do filtro de busca em linha. Se houver tendência de violação de restrições, o algoritmo executa a restauração do sistema *primal-dual*. Isto pode definir que o algoritmo deva recuar na busca ou não. Todos estes fatores são analisados nesta tarefa do algoritmo.

Na decisão sobre a aceitação do ponto proposto pela busca em linha é tomada nesta tarefa do algoritmo. Neste ponto, novos valores propostos para as variáveis primais e duais são calculadas e verificado se o progresso da função objetivo e das violações das restrições foram sensíveis. Se houver correções sensíveis nas violações de restrições e progresso na função objetivo, o novo ponto proposto pode ser considerado como aceitável, e caso contrario é possível verificar qual o tipo de restrição está impedindo este progresso, e posteriormente identificar a variável sua posição no *DAOP* através da análise de violações de restrições. A Figura K.23 fornece uma amostra desta análise.

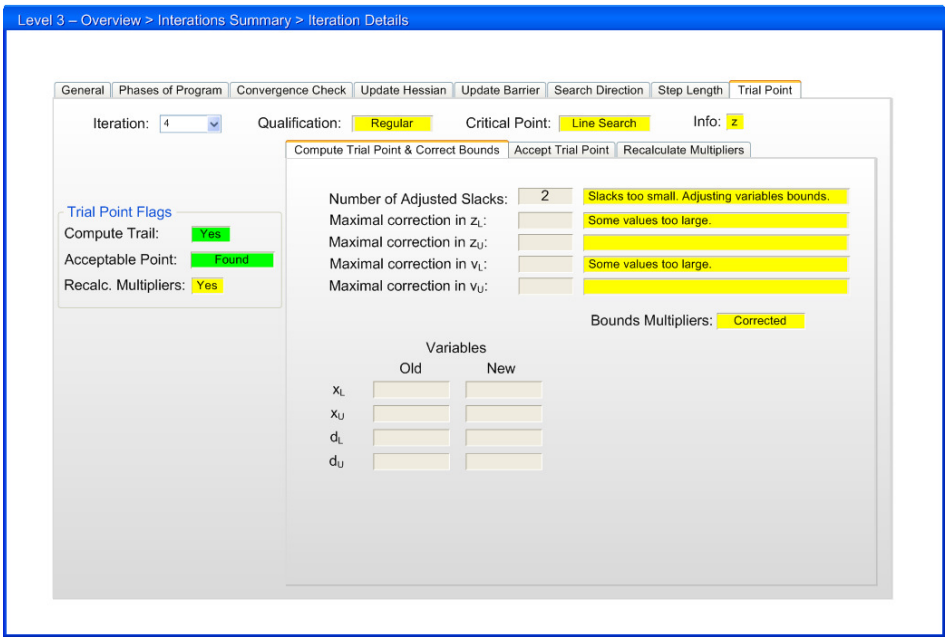


Figura K.23 – Diagnóstico do teste de aceitação do ponto tentativa do *IPOPT*.