UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DOUGLAS VILSON OGLIARI

DESENVOLVIMENTO DE UMA INTERFACE GRÁFICA
para o projeto de controladores no espaço de estados

Porto Alegre
2016

DOUGLAS VILSON OGLIARI

DESENVOLVIMENTO DE UMA INTERFACE GRÁFICA
para o projeto de controladores no espaço de estados

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para Graduação em Engenharia Elétrica.

ORIENTADOR: Prof. Dr. Jeferson Vieira Flores

Porto Alegre

2016

DOUGLAS VILSON OGLIARI

DESENVOLVIMENTO DE UMA INTERFACE GRÁFICA
para o projeto de controladores no espaço de estados

Este projeto foi julgado adequado para fazer jus aos créditos da Disciplina de "Projeto de Diplomação", do Departamento de Engenharia Elétrica e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: Prof. Dr. Jeferson Vieira Flores

Coordenador do Curso: Prof. Dr. Ály Ferreira Flores Filho

Aprovado em: 08/12/2016

BANCA EXAMINADORA

Jeferson Vieira Flores (Prof. Dr.) – Universidade Federal do Rio Grande do Sul

Lucíola Campestrini (Profª. Drª.) – Universidade Federal do Rio Grande do Sul

Fausto Bastos Libano (Prof. Dr.) – Universidade Federal do Rio Grande do Sul

Dedico este trabalho aos meus pais
pelo incentivo e apoio incondicional
em todos os momentos.

AGRADECIMENTOS

Aos meus pais e minha irmã pelo incentivo e apoio, sem os quais teria sido impossível alcançar meus objetivos, incluindo a realização deste trabalho. A todos os demais familiares pela ajuda prestada em momentos de necessidade e por serem exemplos a serem seguidos.

Aos amigos e aos colegas da Engenharia Elétrica pelo companheirismo em todos os momentos e auxílio nas atividades da graduação.

Ao meu orientador, pelo auxílio e tempo dedicado, imprescindíveis para realização deste trabalho.

À UFRGS pela oportunidade de estudar numa grande instituição de ensino. A todos os professores, pelos conhecimentos transmitidos e por servirem de inspiração.

A todas as pessoas que, de alguma forma ou outra, contribuíram para minha evolução pessoal e intelectual.

RESUMO

Este trabalho descreve o desenvolvimento de uma GUI (do inglês, *Graphical User Interface*) em MATLAB para projeto e simulação de controladores para seguimento de referências periódicas. Consideraram-se os controladores ressonante, repetitivo e ressonante-repetitivo e o método de projeto baseado na solução de um problema de otimização com restrições na forma LMI (*Linear Matrix Inequality)*. Optou-se pela criação em MATLAB de uma interface que permite ao usuário definir a descrição em espaço de estados da planta, referência e tipo de controlador para calcular os ganhos de realimentação e simular a saída, sinal de controle e erro de seguimento do sistema. Os resultados mostram que a interface gráfica funciona da maneira desejada, calculando os ganhos de realimentação e, a partir dos gráficos da simulação do sistema, é possível notar que os controladores projetados pela GUI seguem a referência com erro de seguimento praticamente nulo em regime permanente. O *layout* escolhido é simplificado e organizado por meio de três abas: Sistema, Projeto e Simulação. Concluiu-se que a GUI desenvolvida é uma ferramenta útil e eficaz para um usuário que visa projetar um controlador via realimentação de estados de um processo que envolve referências periódicas, eliminando a necessidade de conhecer em grande detalhe os métodos modernos de projeto de sistemas de controle que podem, por vezes, ser bastante complexos para o usuário em nível de aplicação.

Palavras chave: GUI, MATLAB, Seguimento de referências periódicas, LMI

ABSTRACT

This work describes the development of a Graphical User Interface (GUI) in MATLAB for design and simulation of controllers with periodic inputs. The resonant, repetitive and resonant-repetitive controllers and the design method based on a solution of an optimization problem with LMI constraints were considered. It was chosen the creation of a MATLAB interface that allows the user to define the state space description of the plant, the reference and controller type to calculate the feedback gains and to simulate the output, control and error signal of the described system. The results show that the graphical interface works as desired, calculating the feedback gains and, from the graphs of the system simulation, it is possible to notice that the controllers designed by the GUI follow the reference with an error signal close to zero in steady state. The chosen layout is simplified and its organized using three tabs named System, Design and Simulation. The conclusion is that the developed GUI is a useful and effective tool for a user that aims to design a controller via state feedback of a process that involves periodic references, eliminating the need to understand in great detail the modern design methods of control systems that can, sometimes, be quite complex for the user at the application level.

Keywords: GUI, MATLAB, Periodic References Tracking, LMI

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

DC	*Direct Current*
GUI	*Graphical User Interface*
GUIDE	*Graphical User Interface Development Environment*
LMI	*Linear Matrix Inequality*
PMI	Princípio do Modelo Interno
UPS	*Uninterruptible Power Supply*

LISTA DE SÍMBOLOS

$\mathbb{R}$	Conjunto dos números reais
$\mathbb{R}^n$	Espaço euclidiano de ordem n
$\dot{x}(t)$	Derivada temporal de primeira ordem de $x(t)$
A'	Transposta da matriz A
$\mathbb{C}$	Conjunto dos números complexos
$Re(a)$	Parte real do número complexo a
$Im(a)$	Parte imaginária do número complexo a
$\mathbb{R}^+$	Conjunto dos números reais não negativos
$\|x(t)\|_2$	Norma-2 do sinal $x(t)$: $\|x(t)\|_2 = \left(\int_0^\infty z(t)'z(t)\,dt\right)^{\frac{1}{2}}$
$\otimes$	Produto de Kronecker
$0_{n\times m}$	Matriz de dimensão $n \times m$ com todos os elementos nulos
$He\{AW\}$	Bloco hermitiano $AW + W'A'$
$\mathbb{R}^{n\times m}$	Espaço das matrizes reais de dimensão $n \times m$

SUMÁRIO

1. INTRODUÇÃO

Devido à presença de sinais periódicos em diversos sistemas elétricos e eletrônicos, como em sistemas ininterruptos de energia (UPS) (ESCOBAR;VALDEZ;LEYVA-RAMOS; MATTAVELLI, 2007), sistemas antivibração (YAO;TSAI;YAMAMOTO, 2013), inversores de potência (ROHOUMA;ZANCHETTA;WHEELER;EMPRINGHAM, 2015), motores DC (WU;XU;CAO;SHE, 2014), dentre outros exemplos, torna-se importante projetar e construir sistemas de controle que sejam capazes de seguir referências ou rejeitar perturbações periódicas de maneira eficiente. Tradicionalmente, os controladores utilizados para seguimento de referências periódicas são os baseados no Princípio do Modelo Interno (PMI) (CHEN, 1970). O projeto de tais controladores pode ser realizado por meio da solução de um problema de otimização onde as restrições estão na forma LMI (do inglês, *Linear Matrix Inequality*). Essa solução não é de simples implementação e pode não ser de conhecimento ou de principal interesse de engenheiros ou outros profissionais que desejam projetar o controlador para uso em um processo ou sistema real.

O termo GUI é a abreviação de *Graphical User Interface*, que pode ser traduzido livremente do inglês como Interface Gráfica de Usuário. Conforme sugere o nome, uma GUI é uma interface gráfica voltada a um usuário final, que utiliza elementos gráficos como botões, imagens, caixas de texto, ícones, janelas e menus como meio de interação entre o usuário e a interface.

Por ser uma interface gráfica, não é necessário que o usuário escreva comandos ou linhas de código para desempenhar as tarefas, nem mesmo que entenda todos os detalhes de como as tarefas são executadas e os algoritmos envolvidos (MATHWORKS, 2015).

Uma GUI criada em MATLAB, como a que será desenvolvida neste trabalho, possui as vantagens de uma GUI unidas às diversas funcionalidades do MATLAB, podendo ser usadas, por exemplo, para processamento e manipulação de arquivos de dados e exibição de gráficos e tabelas.

Tendo em vista o que foi exposto, o objetivo do trabalho aqui descrito é a construção de uma GUI em MATLAB para projeto e simulação de controladores capazes de seguir referências periódicas. Através da GUI, um usuário interessado em projetar controladores pode obter resultados sem conhecer em detalhes o problema de otimização com restrições na forma LMI. Além disso, a GUI possibilita que o usuário defina por meio de elementos gráficos a descrição em espaço de estados da planta, o tipo de referência e o tipo de controlador, exibe os ganhos calculados e realiza uma simulação do sistema com o

controlador projetado e a referência escolhida através de linhas de código, permitindo que o usuário analise os resultados sem necessidade do Simulink ou de outras ferramentas de simulação.

O texto está organizado da forma seguinte: o Capítulo 2 descreve as fundamentações teóricas do trabalho, descrevendo brevemente os controladores Repetitivo, Ressonante, Ressonante-Repetitivo e Integral e os métodos de projeto, juntamente com uma introdução ao projeto de GUIs em MATLAB. O Capítulo 3 apresenta uma descrição mais detalhada dos objetivos do projeto e relata as etapas de construção da GUI, detalhando o funcionamento de cada elemento da mesma e as decisões de projeto para adequar a GUI aos objetivos propostos. O Capítulo 4 discorre sobre os resultados obtidos ao projetarem-se os controladores Repetitivo, Ressonante, Ressonante-Repetitivo e Integral para um dado sistema e referência, mostrando-se os resultados obtidos no cálculo dos ganhos e na simulação. O Capítulo 5 discute as conclusões que se obteve com o trabalho e sugestões de melhorias futuras.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados aspectos teóricos envolvendo controladores para seguimento e rejeição de sinais periódicos, métodos de projeto e uma breve discussão sobre construção de GUIs em MATLAB.

Será considerado que a planta (sistema ao qual se deseja aplicar uma ação de controle) possui uma representação em espaço de estados na forma:

$$\begin{cases} \dot{x}_p(t) = Ax_p(t) + Bu(t) + B_w w(t) \\ y(t) = Cx_p(t) \\ e(t) = r(t) - y(t) \end{cases}, \quad (1)$$

onde $x_p(t) \in \mathbb{R}^n$ é o vetor de estados do sistema, $u(t) \in \mathbb{R}$ é o sinal de controle, $w(t) \in \mathbb{R}$ é um distúrbio a ser rejeitado, $y(t) \in \mathbb{R}$ é a saída do sistema, $r(t) \in \mathbb{R}$ é uma referência a ser seguida, $e(t)$ é o erro de seguimento da referência e A, B, B_w e C são matrizes que descrevem o sistema.

Na Equação (1), assume-se que os pares (A, B) e (C, A) são respectivamente controláveis e observáveis.

2.1 CONTROLADOR RESSONANTE

Os controladores ressonantes são caracterizados por possuírem em sua função de transferência um pico de ressonância de magnitude infinita na frequência a ser seguida ou rejeitada podendo ser representados por:

$$\mathrm{C}(s) = \frac{1}{s^2 + \omega_0^2}. \quad (2)$$

Sendo um sistema estável em malha fechada, um controlador ressonante introduzido neste sistema garantirá que uma referência senoidal com frequência ω_0 seja reproduzida na saída, resultando em um erro de seguimento nulo em regime permanente. Para um sinal de perturbação semelhante, na mesma frequência ω_0, a saída será nula, indicando que há uma rejeição completa em regime permanente do sinal de perturbação (LORENZINI, 2015).

Como o termo da Equação (2) apresenta dois polos sobre o eixo imaginário (em $\pm j\omega_0$) que são marginalmente estáveis, podem-se introduzir dois zeros no denominador para garantir que o sistema seja estável em malha fechada. Levando em conta a adição dos dois zeros, a função de transferência do controlador ressonante pode ser dada por:

$$\mathrm{C}_{rs}(s) = \frac{k_2(s^2 + \omega_0^2) + k_4 s + k_3}{s^2 + \omega_0^2}, \quad (3)$$

onde k_2, k_3 e k_4 são parâmetros a serem projetados.

Uma representação em espaço de estados para a Equação (3) é mostrada abaixo:

$$\begin{cases} \dot{x}_{rs}(t) = A_{rs}x_{rs}(t) + B_{rs}u_{rs}(t) \\ y_{rs}(t) = C_{rs}x_{rs}(t) + D_{rs}u_{rs}(t) \end{cases}. \tag{4}$$

Na Equação (4), $x_{rs}(t) \in \mathbb{R}^2$ representa o vetor de estados do controlador ressonante, $u_{rs}(t) \in \mathbb{R}$ representa o sinal de entrada do controlador, $y_{rs}(t) \in \mathbb{R}$ representa o sinal de saída do controlador e as matrizes A_{rs}, B_{rs}, C_{rs} e D_{rs} são dadas por:

$$A_{rs} = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix}, B_{rs} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, C_{rs} = [k_3 \quad k_4], D_{rs} = k_2, \tag{5}$$

onde ω_0 é a frequência da referência.

Considera-se a lei de controle na forma $u(t) = Kx(t) + k_2 r(t)$, onde $K = [k_1 \quad -k_2 \quad k_3 \quad k_4] \in \mathbb{R}^{1\times 4}$. A partir da representação em espaço de estados do controlador ressonante, dada pela Equação (4) e da representação em espaço de estados da planta, que tem a forma da Equação (1) e da lei de controle, encontra-se a representação no espaço de estados do sistema em malha fechada:

$$\begin{cases} \dot{x}(t) = (A_s + B_s K)x(t) + B_q q(t) \\ y(t) = C_s x(t) \end{cases}, \tag{6}$$

onde $x(t) = [x_p \quad x_{rs}]$ é o vetor de estados aumentado planta-controlador, $q(t)$ é dado por:

$$q(t) = [r(t) \;\; w(t)]' \in \mathbb{R}^2, \tag{7}$$

e as matrizes são dadas por:

$$A_s = \begin{bmatrix} A & 0_{2\times 2} \\ -B_{rs}C & A_{rs} \end{bmatrix}, B_s = \begin{bmatrix} B \\ 0_{2\times 1} \end{bmatrix}, B_q = \begin{bmatrix} Bk_2 & B_w \\ B_{rs} & 0_{2\times 1} \end{bmatrix}, C_s = [C \quad 0_{1\times 2}]. \tag{8}$$

A formulação para o controlador ressonante de múltiplas harmônicas pode ser encontrada em (FLORES, 2012).

Conforme descrito em (LORENZINI, 2015) pode-se desconsiderar $q(t)$ para fins de estabilização, pois a estabilidade interna implica em estabilidade BIBO para sistemas lineares, e a Equação (6) pode ser simplificada para:

$$\dot{x}(t) = (A_s + B_s K)x(t). \tag{9}$$

Para projetar os ganhos de realimentação do sistema, conforme (CHILALI; GAHINET, 1996) posiciona-se os polos de malha fechada, através de um conjunto de restrições na forma de LMIs para garantir que os mesmos estejam contidos na intersecção de três regiões do plano complexo, dadas por:

$$\begin{gathered} R_{CR} = \{p_i \in \mathbb{C}: Re(p_i) \leq -\sigma, \sigma \in \mathbb{R}^+\} \\ R_{DR} = \{p_i \in \mathbb{C}: |p_i| \leq r, r \in \mathbb{R}^+\} \\ R_{NF} = \left\{p_i \in \mathbb{C}: tg^{-1}\left(\frac{|Im(p_i)|}{|Re(p_i)|}\right) \leq \theta, \theta \in \mathbb{R}^+\right\}, \end{gathered} \tag{10}$$

onde p_i são os i polos do sistema em malha fechada, σ define o tempo de acomodação, θ é o ângulo associado ao fator de amortecimento e r restringe a máxima frequência natural.

Ainda, em (PEREIRA;CARVALHO;FLORES, 2013), além do posicionamento dos polos, é proposta a função custo

$$J\big(z(t)\big) := \|z(t)\|_2^2 = \int_0^\infty z(t)'z(t)dt, \tag{11}$$

como medida de desempenho transitório.

Na Equação (11), o termo $z(t)$ é a saída de desempenho:

$$z(t) \coloneqq C_z x(t) + D_z u(t), \tag{12}$$

onde C_z e D_z são matrizes a serem definidas.

Para obter os ganhos de realimentação do sistema com o controlador ressonante, e para minimizar a função custo da Equação (11) propõe-se o seguinte problema de otimização:

$$\min_{W,Y,\lambda} \lambda \ \ sujeito\ a\ \mathcal{R} \tag{13}$$

Na Equação (13), $\mathcal{R}$ é um conjunto de restrições na forma LMI e λ é um escalar positivo tal que $\mathcal{R}$ seja satisfeito, conforme exemplificado para um sistema UPS em (LORENZINI, 2015). O conjunto de inequações que compõem $\mathcal{R}$ é dado por:

$$\begin{gathered} L_1 \otimes Q + M_1 \otimes (AW + BY) + M_1' \otimes (AW + BY) < 0, \\ L_2 \otimes Q + M_2 \otimes (AW + BY) + M_2' \otimes (AW + BY) < 0, \\ L_3 \otimes Q + M_3 \otimes (AW + BY) + M_3' \otimes (AW + BY) < 0, \\ \begin{bmatrix} He(AW + BY) & WC_z' + Y'D_z' \\ C_z W + D_z Y & -\lambda I_{n_z} \end{bmatrix} < 0, \end{gathered} \tag{14}$$

onde W e Y são matrizes simétricas positivas.

Na Equação (14), o símbolo ⊗ indica o produto de Kronecker e as matrizes L_1, L_2, L_3, M_1, M_2 e M_3 são dadas por:

$$\begin{gathered} L_1 = 2\sigma,\ M_1 = 1, \\ L_2 = \begin{bmatrix} -r & 0 \\ 0 & -r \end{bmatrix},\ M_2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \\ L_3 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, M_3 = \begin{bmatrix} sen(\theta) & cos(\theta) \\ -cos(\theta) & sen(\theta) \end{bmatrix}, \end{gathered} \tag{15}$$

Se existir solução para o problema da Equação (13), determinam-se os ganhos de realimentação através de

$$K = YW^{-1}, \tag{16}$$

onde K é um vetor de ganhos.

2.2 CONTROLADOR REPETITIVO

O controlador repetitivo, assim como o ressonante, é adequado para seguimento e rejeição de sinais periódicos. Apresenta em sua estrutura um elemento de atraso em realimentação positiva, na forma $e^{-s\tau_0}$ onde τ_0 é o período fundamental do sinal a ser seguido ou rejeitado.

Conforme analisado em detalhes em (LORENZINI, 2015) o controlador repetitivo pode ser aproximado por infinitos polos sobre o eixo imaginário, o que compromete a estabilidade do sistema em malha fechada, além de possuir ganho infinito em altas frequências, o que pode levar o sistema a instabilidade e comprometer o desempenho. Para contornar estes comportamentos não desejados, introduz-se um filtro passa-baixas de primeira ordem em série com o elemento de atraso na realimentação positiva. A função de transferência do controlador repetitivo resultante é dada por:

$$C_{rp}(\mathrm{s}) = \frac{1}{1 - \frac{\omega_c}{s + \omega_c} e^{-s\tau_0}}, \tag{17}$$

onde ω_c é a frequência de corte do filtro passa-baixas.

A representação em espaço de estados do controlador repetitivo com filtro passa-baixas da Equação (17) é dada por:

$$\begin{cases} \dot{x}_{rp}(t) = A_{rp}x_{rp}(t) + A_{d_{rp}}x_{rp}(t-\tau_0) + B_{d_{rp}}u_{rp}(t-\tau_0) \\ y_{rs}(t) = C_{rp}x_{rp}(t) + D_{rp}u_{rp}(t) \end{cases}, \tag{18}$$

onde $x_{rs}(t) \in \mathbb{R}$ representa o vetor de estados do controlador ressonante, $u_{rs}(t) \in \mathbb{R}$ representa o sinal de entrada do controlador, $y_{rs}(t) \in \mathbb{R}$ representa o sinal de saída do controlador e as matrizes, definidas por:

$$A_{rp} = -\omega_c, A_{d_{rp}} = \omega_c, B_{d_{rp}} = \omega_c, C_{rp} = 1, D_{rp} = 1. \tag{19}$$

É possível encontrar a representação em espaço de estados do sistema em malha fechada com controlador repetitivo a partir da Equação (18) e da representação da planta no espaço de estados, que tem a forma da Equação (1), resultando em:

$$\begin{cases} \dot{x}(t) = A_s x(t) + A_d x(t-\tau_0) + B_s u(t) + B_q q(t) \\ y(t) = C_s x(t) \end{cases}, \tag{20}$$

onde $q(t)$ é dado por:

$$q(t) = [r(t-\tau_0)\ \ w(t)]' \in \mathbb{R}^2, \tag{21}$$

e as matrizes são dadas por:

$$A_s = \begin{bmatrix} A & 0_{2\times1} \\ 0_{1\times2} & A_{rp} \end{bmatrix}, A_d = \begin{bmatrix} 0_{2\times1} & 0_{1\times1} \\ -B_{d_{rp}}C & A_{d_{rp}} \end{bmatrix}, B_s = \begin{bmatrix} B \\ 0_{1\times1} \end{bmatrix}, B_q = \begin{bmatrix} 0_{2\times2} & B_w \\ B_{d_{rp}} & 0_{1\times1} \end{bmatrix}, \quad C_s = [C \quad 0_{1\times1}] \tag{22}$$

Pode-se definir a lei de controle:

$$u(t) = Kx(t) + \left(k_2 + k_3 D_{rp}\right) r(t), \tag{23}$$

onde $K = \begin{bmatrix} k_1 & -\left(k_2 + k_3 D_{rp}\right) & k_3 C_{rp} \end{bmatrix}$, para que o sistema seja estável em malha fechada. Substituindo a Equação (23) na Equação (0), obtém-se a representação em malha fechada do sistema com controlador repetitivo:

$$\begin{cases} \dot{x}(t) = (A_s + B_s K)x(t) + A_d x(t-\tau_0) + B_h h(t) \\ y(t) = C_s x(t) \end{cases}. \tag{24}$$

O termo $h(t)$ da Equação (24) é dado por:

$$h(t) = [r(t)\ r(t-\tau_0)\ \ w(t)]' \in \mathbb{R}^3, \tag{25}$$

e a matriz B_h é dada por:

$$B_h = \begin{bmatrix} (k_2 + k_3)B & 0_{2\times1} & B_1 \\ 0_{1\times1} & B_{d_{rp}} & 0_{1\times1} \end{bmatrix}. \tag{26}$$

Define-se então a função de transferência do controlador repetitivo em malha fechada, conforme:

$$C_{rp}(\mathrm{s}) = k_2 \frac{k_3}{1 - \frac{\omega_c}{s+\omega_c} e^{-s\tau_0}}. \tag{27}$$

Da mesma forma que o controlador ressonante, para encontrar os ganhos de realimentação pode-se desconsiderar o termo $h(t)$ na Equação (24), resultando em:

$$\dot{x}(t) = (A_s + B_s K)x(t) + A_d x(t-\tau_0). \tag{28}$$

Define-se um problema com restrições em forma de LMI e garante-se um determinado critério de desempenho transitório. Para o caso do controlador repetitivo, conforme (LORENZINI, 2015), o critério de desempenho a ser seguido é:

$$\|x(t)\| \leq \beta \|\varphi\|_{\tau_0} e^{-\alpha t}. \tag{29}$$

Na Equação (29), β é uma constante positiva, α é a taxa de decaimento exponencial para a norma da trajetória de estados e $\|\varphi\|_{\tau_0}$ é dada por:

$$\|\varphi\|_{\tau_0} \coloneqq \sup_{t\in[-\tau_0,0]} \|\varphi(t)\|, \tag{30}$$

onde $\varphi(t)$ é a função inicial que descreve o comportamento de $x(t)$ no intervalo$[-\tau_0, 0]$.

A medida de desempenho transitório proposta é a mesma da Equação (11), apresentada na Seção 2.1. Os ganhos de realimentação para o sistema com controlador repetitivo são encontrados propondo-se um problema de otimização da forma:

$$\min_{W,S,Y,\lambda} \lambda \ sujeito\ a\ \mathcal{R}, \tag{31}$$

onde $\mathcal{R}$ é um conjunto de restrições na forma LMI, conforme descrito detalhadamente em (LORENZINI, 2015) para o caso de um sistema UPS.

Na Equação (31), o termo λ é um escalar que satisfaz as inequações que compõem o conjunto de restrições $\mathcal{R}$, sendo estas últimas expressas na seguinte forma:

$$\begin{bmatrix} He(AW+BY)+S+2\alpha W & e^{\alpha\tau_0}A_dW & WC_z'+Y'D_z' \\ * & -S & 0 \\ * & * & -\lambda I_{n_z} \end{bmatrix} < 0, \tag{32}$$

onde W e S são matrizes positivas simétricas e Y é uma matriz.

Ao resolver o problema sujeito às restrições citadas, deve-se minimizar a função custo dada pela Equação (11), bem como encontrar uma boa relação entre erro de seguimento, distorção harmônica total e resposta transitória do sistema, para determinados α e ω_c dados. Se o problema tiver solução, então é possível calcular os ganhos de realimentação conforme a Equação (16), apresentada na Seção 2.1, onde K é um vetor de ganhos.

2.3 CONTROLADOR RESSONANTE-REPETITIVO

Este tipo de controlador é uma junção do controlador ressonante com o controlador repetitivo, associados em paralelo, proposta em (SALTON et al., 2013). Conforme exposto mais detalhadamente em (LORENZINI, 2015) essa junção traz benefícios, pois para se obter rejeição de sinais periódicos com bom desempenho em controladores ressonantes, faz-se necessária uma estratégia de controle demasiado complexa e de elevada ordem, enquanto os controladores repetitivos não garantem um seguimento de referência com erro nulo. Assim, a junção dos dois tipos de controladores permite que as suas características se complementem, tornando o controlador híbrido mais eficiente. O seguimento da referência com erro nulo é garantido pelo controlador ressonante enquanto a rejeição das harmônicas do sinal de perturbação é garantida pelo controlador repetitivo.

Considerando as representações no espaço de estados obtidas para os controladores ressonante e repetitivo, na Seção 2.1 e 2.2 respectivamente e a Equação (1), encontra-se a representação em espaço de estados para a planta com controlador ressonante-repetitivo:

$$\begin{cases} \dot{x}(t) = A_s x(t) + A_d x(t-\tau_0) + B_s u(t) + B_q h(t) \\ y(t) = C_s x(t) \end{cases}, \tag{33}$$

sendo $h(t)$ dado por:

$$h(t) = [r(t)\ r(t-\tau_0)\ \ w(t)]' \in \mathbb{R}^3, \tag{34}$$

e as matrizes dadas por:

$$A_s = \begin{bmatrix} A & 0_{2\times2} & 0_{2\times1} \\ -B_{rs}C & A_{rs} & 0_{2\times1} \\ 0_{1\times2} & 0_{1\times2} & A_{rp} \end{bmatrix}, A_d = \begin{bmatrix} 0_{2\times2} & 0_{2\times2} & 0_{2\times1} \\ 0_{2\times2} & 0_{2\times2} & 0_{2\times1} \\ -B_{d_{rp}}C & 0_{1\times2} & A_{d_{rp}} \end{bmatrix}, B_s = \begin{bmatrix} B \\ 0_{2\times1} \\ 0_{1\times1} \end{bmatrix},$$
$$B_q = \begin{bmatrix} 0_{2\times1} & 0_{2\times1} & B_w \\ B_{rs} & 0_{2\times1} & 0_{2\times1} \\ 0_{1\times1} & B_{d_{rp}} & 0_{1\times1} \end{bmatrix}, C_s = [C \quad 0_{1\times2} \quad 0_{1\times1}]. \tag{35}$$

Para que o sistema da Equação (33) seja estável em malha fechada, define-se a lei de controle:

$$u(t) = Kx(t) + \left(k_2 + k_5 D_{rp}\right) r(t), \tag{36}$$

onde K é dado por:

$$K = \begin{bmatrix} k_1 & -\left(k_2 + k_5 D_{rp}\right) & k_3 & k_4 & k_5 C_{rp} \end{bmatrix} \in \mathbb{R}^{1\times5}. \tag{37}$$

Substituindo a Equação (36) na Equação (33) obtém-se o sistema em malha fechada:

$$\begin{cases} \dot{x}(t) = (A_s + B_s K)x(t) + A_d x(t-\tau_0) + B_h h(t) \\ y(t) = C_s x(t) \end{cases}, \tag{38}$$

onde a matriz B_h é dada por:

$$B_h = \begin{bmatrix} \left(k_2 + k_5 D_{rp}\right) & 0_{2\times1} & B_q \\ B_{rs} & 0_{2\times1} & 0_{2\times1} \\ 0_{1\times1} & B_{d_{rp}} & 0_{1\times1} \end{bmatrix}. \tag{39}$$

A função de transferência da associação em paralelo dos controladores ressonante e repetitivo é dada por:

$$C_{rr}(\mathrm{s}) = \frac{k_2(s^2 + \omega_0^2) + k_4 s + k_3}{s^2 + \omega_0^2} + \frac{k_5}{1 - \frac{\omega_c}{s + \omega_c} e^{-s\tau_0}}. \tag{40}$$

Os ganhos de realimentação do sistema em malha fechada com o controlador ressonante-repetitivo podem ser calculados através do problema de otimização e das mesmas restrições e função custo utilizada para o controlador repetitivo da Seção 2.2, correspondendo as Equações (31), (32) e (29), respectivamente. Se existir solução, então os ganhos de realimentação são calculados pela Equação (16), onde K é um vetor de ganhos.

2.4 CONTROLADOR DE AÇÃO INTEGRAL

Segundo (BAZANELLA;SILVA, 2000) a ação de controle integral consiste em aplicar um sinal de controle $u(t)$ proporcional à integral do sinal de erro $e(t)$. A função de transferência de um controlador integral é:

$$C_I(\mathrm{s}) = \frac{1}{sT_I} = \frac{K_I}{s}, \tag{41}$$

onde T_I é chamado de tempo integral ou *reset-time.*

Se a partir de um dado instante de tempo t, $e(t) = 0$ o sinal de controle será mantido em um valor constante proporcional à energia armazenada até aquele instante de tempo. Este fato permite que o sistema em malha fechada siga uma referência com erro nulo em regime permanente.

Ainda segundo (BAZANELLA;SILVA, 2000) devido ao fato de que o controlador de ação integral introduz um polo na origem na função de transferência em malha aberta, há uma tendência de piorar a estabilidade relativa do sistema em malha fechada ou até mesmo torná-lo instável.

O procedimento para projeto dos ganhos de realimentação para o sistema com controlador de Ação Integral segue os mesmos passos descritos para o controlador Ressonante, exceto que na representação em espaço de estados da Equação (4), substituem-se as matrizes A_{rs}, B_{rs}, C_{rs} e D_{rs} pelas matrizes abaixo:

$$\mathrm{A}_I = [0],\ B_I = [1],\ C_I = [K_I],\ D_I = [0]. \tag{42}$$

2.5 GUI (*GRAPHICAL USER INTERFACE*)

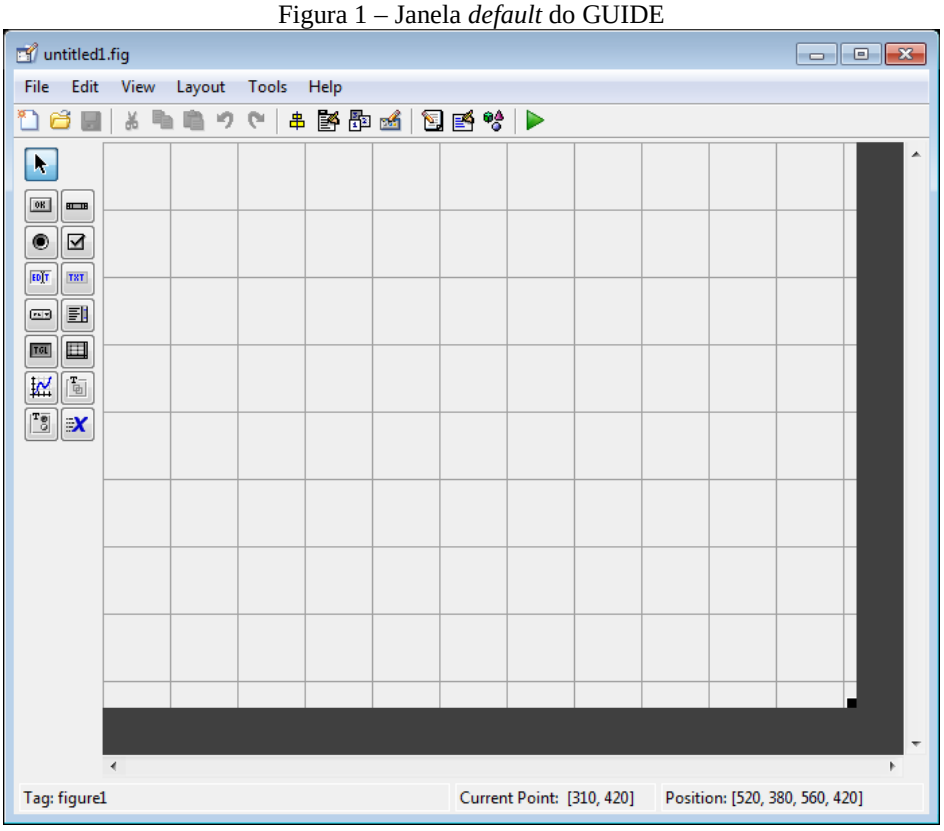
Em MATLAB, existem duas maneiras de desenvolver-se uma GUI: por meio do GUIDE (o termo vem de *Graphical User Interface Development Environment*, ou seja, é um ambiente voltado para desenvolvimento de GUIs, fornecendo ferramentas para criação da parte gráfica) e pela programação direta em um *script*. Desenvolver uma GUI por programação implica em descrever todos os componentes, suas propriedades e comportamento por meio de linhas de código. Quando se utiliza o GUIDE, as informações sobre o *layout* gráfico são armazenadas em um arquivo de extensão .fig e associado a esse arquivo é criado outro, de extensão .m, contendo linhas de código para determinar o comportamento de cada componente gráfico da GUI e suas interações com o usuário.

Pelo fato do GUIDE apresentar maior simplicidade no desenvolvimento do *layout* gráfico, foi o método escolhido para criar a GUI proposta neste trabalho e o foco será direcionado a ele.

A Figura 1 mostra o GUIDE, assim que é aberto no MATLAB. Conforme mostra a Figura 1, o GUIDE oferece alguns elementos com os quais se pode construir a parte gráfica do GUI. A cada um destes elementos será associado um *callback*, que são linhas de código que descrevem o comportamento de cada elemento e suas interações. Os *callbacks* são executados quando o usuário executa uma ação chamada de evento. Cada elemento aciona o *callback* por um evento em particular, podendo ser um clique, uma tecla sendo pressionada, etc.

Um elemento pode ser utilizado em um *callback* distinto do seu próprio, no entanto, neste caso, a ação será executada quando ocorrer um evento no elemento do *callback* em que é utilizado, e não quando da ocorrência de um evento no elemento em si.

Figura 1 – Janela *default* do GUIDE



Fonte: MathWorks, 2015

Alguns dos elementos importantes para elaboração deste projeto serão listados abaixo, com uma pequena descrição de seu funcionamento.

- *Radio Button*: Botão de seleção, normalmente utilizado em conjunto com outros *Radio Buttons*. O evento se dá ao selecionar o botão.

- *Edit Text*: Caixa de texto editável que pode aceitar uma ou mais linhas de entrada e mostrar um texto inicial. O evento se dá ao pressionar a tecla *Enter* com o cursor na caixa de texto, dentre outras formas.
- *Static Text:* Caixa de texto não editável, normalmente não necessita de um *callback*. Exibe uma *string* estática.
- *Pop-Up Menu:* Menu que pode ser expandido para englobar diversas opções de forma compacta. O evento se dá ao selecionar uma das opções.
- *Axes:* Elemento que permite exibir gráficos e imagens.
- *Panel:* Elemento que cria um painel com bordas e título, tornando a GUI mais organizada e fácil de compreender. Normalmente não necessita de um *callback*.

Cada um dos elementos possui propriedades que devem ser ajustadas de acordo com a funcionalidade desejada. Duas propriedades devem ser destacadas, pois aparecem com frequência neste trabalho e estão listadas abaixo.

- *Tag:* É a propriedade do elemento com a qual será nomeado um *callback* e a partir da qual se define o elemento a ser manipulado pelo código dos *callbacks*.
- *Title:* Propriedade que define o título do elemento, geralmente utilizada em *Panels.*

3. DESENVOLVIMENTO DA GUI

O processo de desenvolvimento da GUI levou em conta um usuário final, portanto tinha como objetivo uma interface amigável, agregando simplicidade, funcionalidade e uma disposição organizada dos elementos da GUI.

A GUI deve conter três componentes principais, resumidos abaixo:

- Entrada de dados para definição do modelo do sistema no espaço de estados, tipo de referência e demais parâmetros.
- Cálculo dos ganhos do controlador selecionado.
- Simulação do sistema em malha fechada.

Para que os componentes acima fossem incorporados na GUI projetada, optou-se por criar um *layout* com três abas: Sistema, Projeto do Controlador e Simulação. A divisão em três abas permite organizar a GUI de maneira clara e objetiva evitando que as informações exibidas ao usuário fiquem congestionadas, devido à falta de espaço na tela.

Os objetivos a serem alcançados no desenvolvimento da GUI foram definidos para nortear o projeto (alguns para garantir que a mesma operasse de maneira mais eficiente levando em conta as interações com o usuário, outros por trazerem vantagens ao projeto), e estão resumidos abaixo:

- Permitir que os dados a serem definidos pelo usuário sejam carregados a partir da *workspace* ou digitados diretamente na GUI.
- Tornar automáticas as tarefas que devem ser executadas pela GUI, sempre que possível.
- Mostrar ao usuário, por meio de uma barra de *status*, quando o problema de cálculo dos ganhos tem ou não solução.
- Permitir que o usuário ajuste o tempo de simulação e a escala dos gráficos.
- Realizar simulação por linha de código, sem dependência do *Simulink*.
- Mostrar o gráfico da saída, do sinal de controle e do erro de seguimento na simulação.
- Para simplificar o projeto, o sinal de distúrbio será considerado nulo.

Para todo o desenvolvimento descrito neste capítulo utilizou-se o *software* MATLAB R2012a. Algumas versões mais atuais do MATLAB apresentam funcionalidades adicionais, que possivelmente teriam tornado o desenvolvimento da GUI muito mais simples. No

entanto, a *release R*2012a foi a escolhida, pois era a que estava disponível para uso pelos alunos da UFRGS.

3.1 CRIAÇÃO DE UMA INTERFACE COM ABAS

A *release* do MATLAB utilizada neste trabalho, a R2012a, apresenta uma função que permite a criação de abas, chamada *uitab*, que não está documentada nos arquivos de ajuda do *software.* No entanto, por se tratar de uma função, deve ser iniciada por linha de código, o que inviabiliza a construção do *layout* desejado através do GUIDE.

Devido a falta de uma ferramenta para criação de abas no GUIDE, utilizou-se um arquivo .m que desempenha esta tarefa, chamado *TabManager*, que pode ser encontrado no *website* da MathWorks, de autoria de (DAVIDSON, 2016).

O *TabManager* cria abas utilizando a função *uitab*, tendo como referência no GUIDE o elemento *Panel*. Para que esse recurso funcione corretamente, é necessário criar *Panels* cujas *Tags* tenham o prefixo *Tab*, de modo que os elementos que forem alocados dentro deles serão exibidos na aba correspondente. O parâmetro *Title* de cada um dos *Panels* definirá o nome da aba que corresponde a ele.

O primeiro *Panel* servirá de *Background*, ou seja, o *TabManager* criará no GUI uma janela com abas que possuem as mesmas dimensões e a mesma posição (na tela) do *background.* Os demais *Panels* criados corresponderão às abas desejadas. Para que essa correspondência se dê da forma correta, é necessário nomear as *Tags* de cada *Panel* da maneira correta. Assim, o *background* deverá ter, por exemplo, TabA como *Tag*, enquanto a primeira aba deverá ter *Tag* TabA01, a segunda TabA02 e assim por diante, respeitando sempre o prefixo e a ordem alfabética ou numeral.

Para construir o GUI desejado, criou-se no GUIDE quatro *Panels*, conforme mostrado na Tabela 1, onde é informado a *Tag* e o *Title* de cada *Panel* bem como uma pequena descrição de sua função.

Tabela 1 - *Panels* criados para construção do GUI

Panel	*Tag*	*Title*	Descrição
1º	TabA	Background	Painel que define o tamanho e a posição da janela
2º	TabA1	Sistema	Aba Sistema
3º	TabA2	Projeto do Controlador	Aba Projeto do Controlador
4º	TabA3	Simulação	Aba Simulação

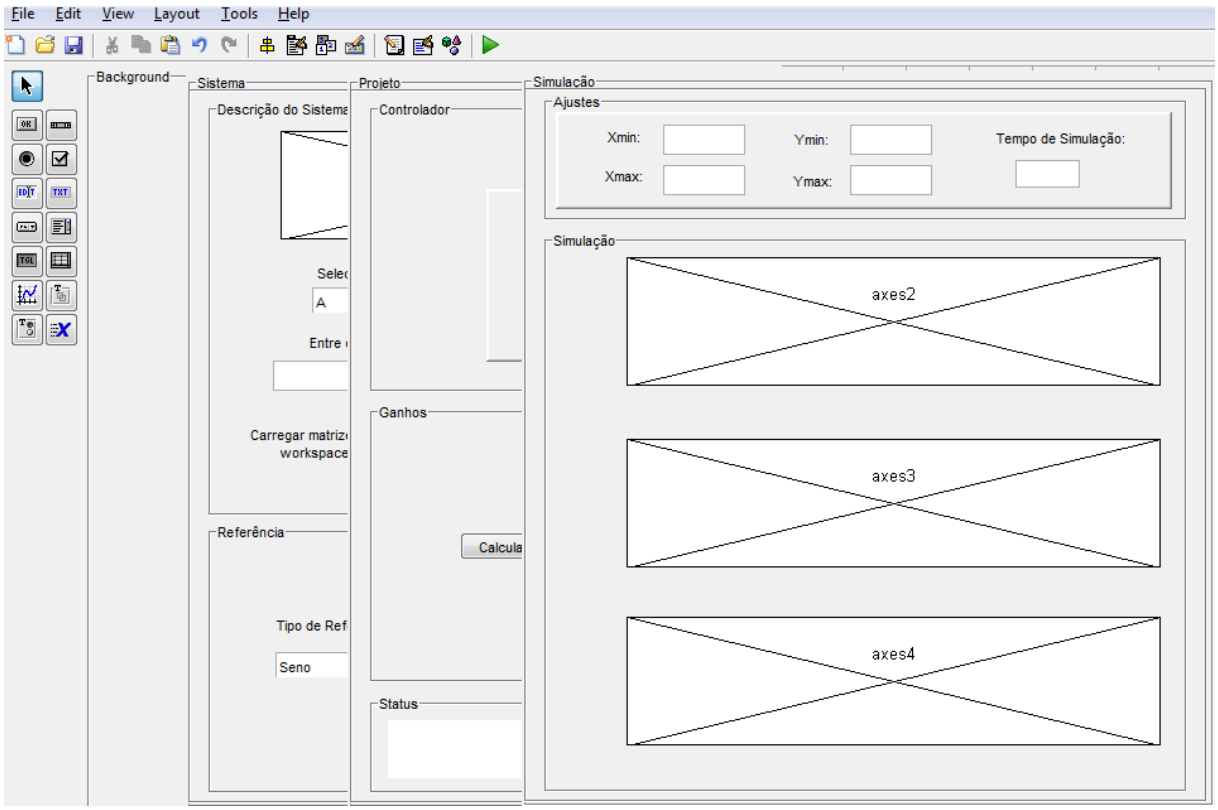
A Figura 2 mostra o GUIDE com os quatro *Panels* já dimensionados, sobrepostos para que possam ser exibidos numa só imagem.

Para que o *TabManager* seja iniciado juntamente com o GUI, o arquivo *TabManager.m* deve ser alocado no mesmo diretório onde está a GUI e utilizou-se a linha de código, mostrada no Quadro 1, na *opening function*, função que é a primeira a ser executada quando aberta a GUI.

Quadro 1 – Inicialização do *TabManager*

```
handles.tabManager = TabManager(hObject);
```

Figura 2 - *Panels* utilizados no GUIDE para implementação de abas



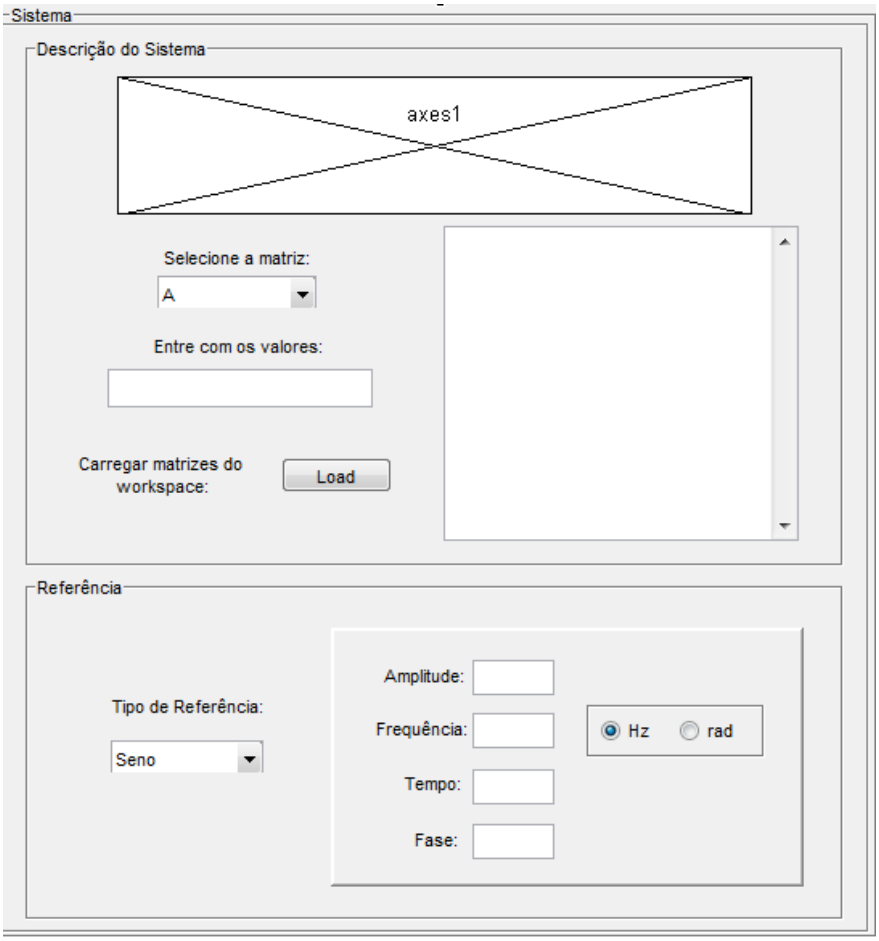
Fonte: O próprio autor

3.2 ABA SISTEMA

A primeira aba, Sistema, foi dimensionada de modo a acomodar dois componentes básicos do projeto: a entrada de matrizes para descrição do sistema no espaço de estados e a

escolha da referência e definição dos seus parâmetros. A Figura 3 mostra a aba Sistema conforme foi construída no GUIDE.

Figura 3 - Aba Sistema, conforme construída no GUIDE



Fonte: O próprio autor

Conforme se observa na Figura 3, a aba Sistema foi ainda dividida em dois *Panels:* Descrição do Sistema e Referência.

No *Panel* Descrição do Sistema, foi colocada uma imagem com a representação matemática de um sistema no espaço de estados, através de um objeto *Axes.* A imagem foi alocada no diretório da GUI, com nome de sistema.jpg e é mostrada no objeto *axes1* através dos comandos mostrados no Quadro 2, que carregam a imagem e a mostram no objeto especificado.

A entrada de matrizes foi feita através de três objetos: um *Pop-Up Menu* (de *Tag popupmenu1*) que permite selecionar a matriz que se deseja definir, um *Edit Text* (de *Tag edit1*) que permite a digitação dos valores da matriz e outro *Edit Text* (de *Tag edit2*) que

Quadro 2 – Comandos para exibição da imagem sistema.jpg

```
img = imread('sistema.jpg');
axes(handles.axes1);
imshow(img)
```

tem como função mostrar os valores das matrizes após o usuário defini-las (optou-se por *Edit Text* em lugar de um simples *Static Text* pois o primeiro permite barra de rolagem, que se faz necessária para visualização completa das matrizes, caso elas possuírem dimensões maiores do que o tamanho definido de um *Static Text* permite mostrar).

O *popupmenu1* indica em qual das matrizes será salva a informação fornecida no *edit1*, A, B, Bw ou C. Uma vez definida qualquer uma das matrizes, o *edit2* mostra seu valor, atribuindo para as demais valor *default* igual a zero para evitar erros devido a variáveis indefinidas durante a execução da GUI, e a matriz é salva no *workspace base* com seu nome correspondente. O *edit1* permite ainda que se entre com o nome de uma variável que já esteja no *workspace base* e atribui seu valor à variável selecionada no *popupmenu1*. O trecho de código mostrado no Quadro 3 implementa as tarefas descritas, para o caso da matriz A.

Quadro 3 – Linhas de código para inicialização das matrizes que descrevem o sistema

```
matr = get(handles.popupmenu1,'Value');
if (matr == 1)
    e1s = get(handles.edit1, 'String');
    snt = str2num(e1s);
    if(isempty(snt))
        mma = evalin('base',e1s);
        assignin('base','A',mma);
    else
        m_a = str2num(get(handles.edit1,'String'));
        assignin('base','A',m_a);
```

No código do Quadro 3, o parâmetro *Value* do objeto *popupmenu1* corresponde à matriz selecionada (para a matriz A, *Value* = 1, para a B, *Value* = 2 e assim sucessivamente). Um laço *if* verifica qual a matriz selecionada e atribui à matriz A o valor digitado em *edit1* ou o valor da variável da *workspace base* caso existir. Esta parte do código foi inserida no *callback* de *edit1*, para que as variáveis sejam armazenadas automaticamente sempre que o valor de uma das matrizes for alterado.

O botão *Load*, de *Tag pushbutton4* carrega as matrizes A, B, Bw e C e exibe as mesmas para o usuário no elemento *edit2*, caso as mesmas já estejam definidas no *workspace base*. No *callback* do *pushbutton4* utilizou-se as linhas de código mostradas no Quadro 4.

Quadro 4 – Comandos utilizados no botão *Load*

```
     mostraa = num2str(evalin('base','A'));
     mostrab = num2str(evalin('base','B'));
     mostrabw = num2str(evalin('base','Bw'));
     mostrac = num2str(evalin('base','C'));

     nm2 = {'A=' mostraa 'B=' mostrab 'Bw=' mostrabw 'C='
mostrac};
     set(handles.edit2,'String',nm2);
```

As linhas do Quadro 4 carregam os valores das matrizes através do comando *evalin* e montam o *string nm2* com o nome e o valor de cada matriz. O *string* é então exibido ao usuário através do *edit2*, utilizando-se um comando *set.* As mesmas linhas de código são utilizadas no *callback* de *edit1*, para exibir as matrizes caso o usuário deseje definir seus valores manualmente.

No *Panel* Referência, utilizou-se um objeto *Pop-Up Menu* (de *Tag popupmenu2)* para selecionar o tipo de referência, Seno, Triangular ou Step, quatro objetos *Edit Text (*com *Tags edit3*, *edit4*, *edit5*, *edit6)* para editar os seus parâmetros, Amplitude, Frequência, Tempo e Fase, um objeto *Button Group* e dois objetos *Radio Button* (com *Tags radiobutton1* e *radiobutton2*) para seleção da unidade de frequência, Hz ou rad.

Como a referência *Step* não tem frequência e fase, os dois *Edit Text* correspondentes são desabilitados ao selecioná-la, por meio das linhas de comando mostradas no Quadro 5.

Quadro 5 – Comandos que desabilitam frequência e fase para referência step

```
fref = get(handles.popupmenu2,'Value');
if (fref == 3)
    set(handles.edit4,'Enable','Off');
    set(handles.edit6,'Enable','Off');
else
    set(handles.edit4,'Enable','On');
    set(handles.edit6,'Enable','On');
```

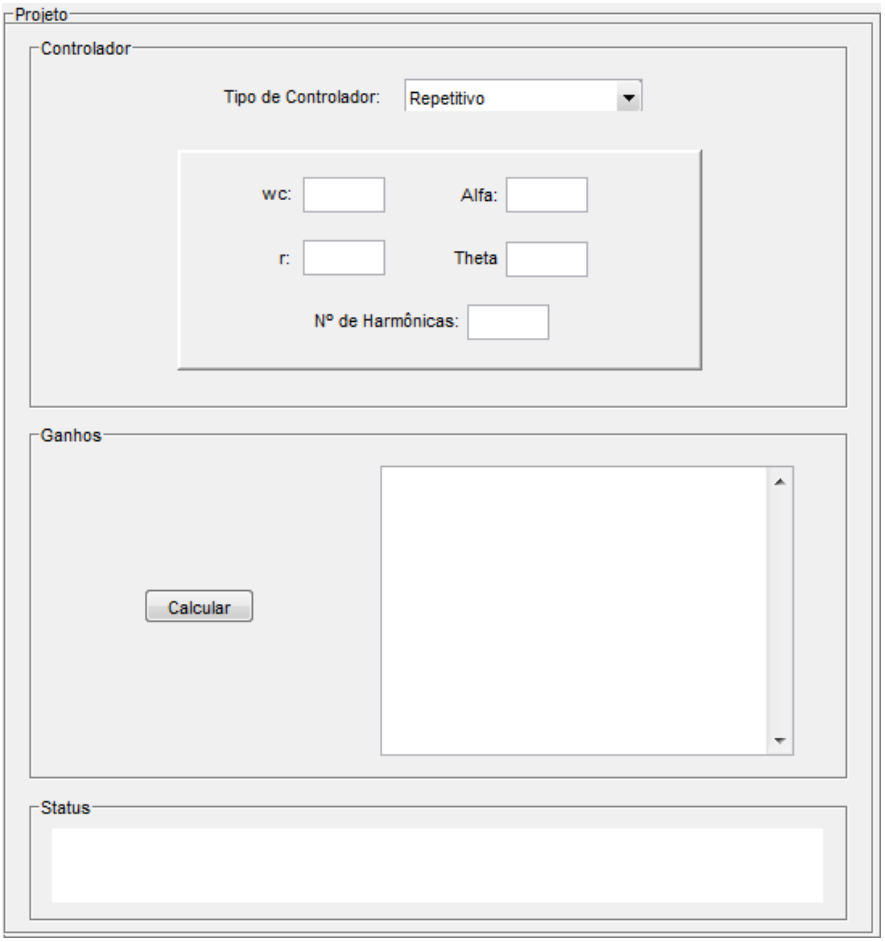
Conforme se observa no Quadro 5, se *Value* do *popupmenu2* for 3, correspondendo a referência Step, então se desabilita *edit4* e *edit6*, correspondentes a frequência e fase, ou habilita caso contrário. Nas *callbacks* dos objetos *edit3* até *edit6* não utilizou-se nenhum comando, pois os valores das variáveis correspondentes serão manipulados em outras partes do código.

Os botões de seleção da unidade de frequência devem conter uma restrição para que apenas uma unidade esteja selecionada, nunca as duas simultaneamente. Isto é garantido inserindo-se ambos em um *Button Group*, que já possui por *default do MATLAB* essas condições, sem que se faça necessário qualquer linha de código.

3.3 ABA PROJETO

A segunda aba, chamada de Projeto do Controlador, é aquela que permite a seleção do tipo de controlador a ser utilizado e definição dos parâmetros de desempenho e cálculo e determina os ganhos de realimentação. A Figura 4 mostra a aba Projeto do Controlador conforme foi construída no GUIDE.

Figura 4 - Aba Projeto, conforme construída no GUIDE



Fonte: O próprio autor

Conforme se observa na Figura 4, a Aba Sistema foi ainda dividida em três *Panels:* Controlador, Ganhos e Status.

O *Panel* Controlador permite ao usuário selecionar o tipo de controlador a ser utilizado no projeto a partir de um *Pop-Up Menu* (de *Tag popupmenu3)* com as opções Repetitivo, Ressonante, Ressonante-Repetitivo e Integral. Permite ainda definir os parâmetros projeto, ω_c, Alpha, r, Theta e Nº de harmônicas através de cinco objetos *Edit Text (*com *Tags edit8* até *edit11).*

Alguns dos parâmetros de desempenho e cálculo devem ser desabilitados conforme o tipo de controlador, pois não serão utilizados no cálculo dos ganhos. A Tabela 2 mostra os parâmetros utilizados no projeto, conforme o tipo de controlador.

A Tabela 2 foi implementada na GUI através de laços *if* no *callback* do *popupmenu3*, de modo que, ao selecionar um controlador, as entradas de parâmetros são anuladas e desabilitadas conforme mostra o Quadro 6.

Tabela 2 – Parâmetros utilizados no projeto dos controladores

Tipo de Controlador	ω_c	Alpha	r	Theta	Nº de harmônicas
Repetitivo	Sim	Sim	Não	Não	Não
Ressonante	Não	Sim	Sim	Sim	Sim
Ressonante-Repetitivo	Sim	Sim	Não	Não	Não
Integral	Não	Sim	Sim	Sim	Não

Quadro 6 – Código que habilita ω_c e Alpha e desabilita os demais parâmetros no caso do controlador repetitivo

```
tcont = get(handles.popupmenu3,'Value');

if (tcont == 1)

    set(handles.edit7,'Enable','On');

    set(handles.edit8,'Enable','On');

    set(handles.edit9,'Enable','Off');

    set(handles.edit10,'Enable','Off');

    set(handles.edit11,'Enable','Off');
```

No trecho mostrado no Quadro 6, a variável tcont armazena o *Value* do *popupmenu3*. Em seguida um laço de i*f* verifica por meio do *Value* qual controlador foi selecionado e desabilita os *edits* correspondentes, conforme Tabela 2.

O *Panel* Ganhos utiliza-se apenas de um *Edit Text (Tag edit12)* e um *Push Button* (*Tag pushbutton1*, nomeado botão Calcular*)*. O *edit12* exibe os ganhos calculados quando o botão Calcular for pressionado, desde que todos os parâmetros necessários estejam definidos e que o problema seja factível.

Para o cálculo dos ganhos de realimentação utilizou-se um arquivo .m adaptado de (LORENZINI, 2015). Este arquivo, chamado de *lmi_res_pep.m* implementa em MATLAB os métodos de projeto descritos no Capítulo 2 e portanto não será explicado em detalhes neste capítulo. Basicamente o arquivo *lmi_res_rep.m* recebe as matrizes que descrevem o sistema aumentado (que são geradas pelo código inserido no *callback* do botão Calcular), os parâmetros de projeto e desempenho e o tipo de controlador como argumento de entrada, e a partir disso monta as restrições na forma LMI e resolve o problema de otimização, retornando um vetor de ganhos.

No *callback* do botão Calcular, uma variável armazena o tipo de controlador selecionado, e laços de *if* verificam qual controlador selecionado e executam os comandos

para calcular os ganhos correspondentes. As linhas mostradas no Quadro 7 exemplificam o código utilizado para o caso do controlador repetitivo. Para os outros controladores, utilizou-se o mesmo código com algumas alterações de acordo com o tipo de controlador, conforme descrito no Capítulo 2.

Quadro 7 – Verificação do tipo de controlador selecionado e unidade de frequência

```
      tcont = get(handles.popupmenu3,'Value');
      if (tcont == 1)
          frad = get(handles.radiobutton2,'Value');
          if (frad == 1)
               w0  =  str2num(get(handles.edit4,'String'));
%frequencia da referencia, em radianos, edit4
               wr = w0;
               tau = (2*pi)/w0;
          else
               w0 = str2num(get(handles.edit4,'String'));
               wr = 2*pi*w0;
               tau = 1/w0;
```

O trecho do Quadro 7 verifica se o controlador repetitivo está selecionado, quando a variável *tcont* possui valor igual a 1. A variável *frad* armazena o parâmetro *Value* do elemento *radiobutton2*, correspondente ao botão que seleciona a frequência em radianos. Um laço *if* verifica se a frequência em radianos foi selecionada através da variável *frad* e atribui valores para ω_r e tau adequados para um ω_0 em radianos caso a condição do laço *if* for verdadeira. Caso contrário, os valores de ω_r e tau são atribuídos para um ω_0 em Hz.

O trecho do Quadro 8 define ω_c a partir do valor fornecido pelo usuário, e a seguir monta as matrizes do controlador e a matriz de atraso do sistema aumentado.

O trecho do Quadro 9 monta as matrizes do sistema aumentado e executa o arquivo *lmi_res_rep.m*, que está no mesmo diretório da GUI, com os argumentos de entrada adequados. A saída (os ganhos calculados) é armazenada na variável de nome F. Para

complementar a descrição dos comandos utilizados na Aba Projeto, primeiramente é necessária uma descrição do *Panel Status*.

Quadro 8 – Código que monta as matrizes do controlador e matriz de atraso do sistema aumentado

```
[n,m] = size(Bp);
[p,lixo] = size(Cp);


wc = str2num(get(handles.edit7,'String')); %wc ,
alfa = str2num(get(handles.edit8,'String'));
r = 0;
theta = 0;
Ac = -wc;
Bc = 0;
Adrc = wc;
Brc = wc;
Ad = [zeros(n,n+1);-Brc*Cp Adrc];
nc = length(Ac);
```

Quadro 9 – Código que monta as matrizes do sistema aumentado e executa a função *lmi_res_rep*

```
A = [Ap zeros(n,nc);-Bc*Cp Ac];
B = [Bp;zeros(nc,m)];
C = [Cp zeros(p,nc)];
D = 1;
ctrle = tcont;
F = lmi_res_rep(A,B,Ad,tau,alfa,r,theta,ctrle);
```

O *Panel Status* é composto apenas de um elemento do tipo *Static Text*, que exibe ao usuário as seguintes mensagens:

- "Controlador (Repetitivo/Ressonante/Ressonante-Repetitivo/Integral) selecionado. Aguardando parâmetros", caso alguns dos parâmetros necessários ainda não tenha sido definido.
- "Ganhos calculados!", ao pressionar o botão Calcular, se o problema for factível.
- "Problema infactível, redefina os parâmetros.", ao pressionar o botão Calcular, se o problema for infactível.

Os comandos reponsáveis pelos avisos acima foram inseridos no *callback* do botão Calcular e na função *lmi_res_rep.m*. O trecho de código mostrado no Quadro 10 mostra as linhas que foram inseridas na função *lmi_res_rep.m* para detectar se o problema é factível ou não.

Quadro 10 – Linhas de código para detecção da factibilidade do problema de cálculo dos ganhos

```
[copt,xopt] = mincx(lmisys,co,otim);
xoptl = length(xopt);
if (xoptl >= 1)
      W_o = dec2mat(lmisys,xopt,W);
      S_o = dec2mat(lmisys,xopt,S);
      Y_o = dec2mat(lmisys,xopt,Y);
      lambda_o = dec2mat(lmisys,xopt,lambda);
      P = inv(W_o);
      F = Y_o*P;
else
      F = 0;
end
```

O trecho do Quadro 10 verifica o tamanho do vetor *xopt*, armazenando o valor na variável *xoptl*. O vetor *xopt* é relacionado ao problema de otimização descrito no Capítulo 2. Se sua dimensão é maior ou igual a 1, o problema tem solução e a função *lmi_res_rep.m* retorna o vetor de ganhos *F*, caso contrário, o problema não tem solução e é retornado $F = 0$.

No *callback* do botão Calcular, a variável *F* é retornada pela função *lmi_res_rep.m* com os ganhos calculados ou com valor zero. O código do Quadro 11 foi inserido neste *callback* para mostrar a mensagem adequada na *Status Bar* de acordo com o valor de *F* retornado.

Quadro 11 – Comandos que retornam ao usuário os avisos sobre a factibilidade do cálculo dos ganhos

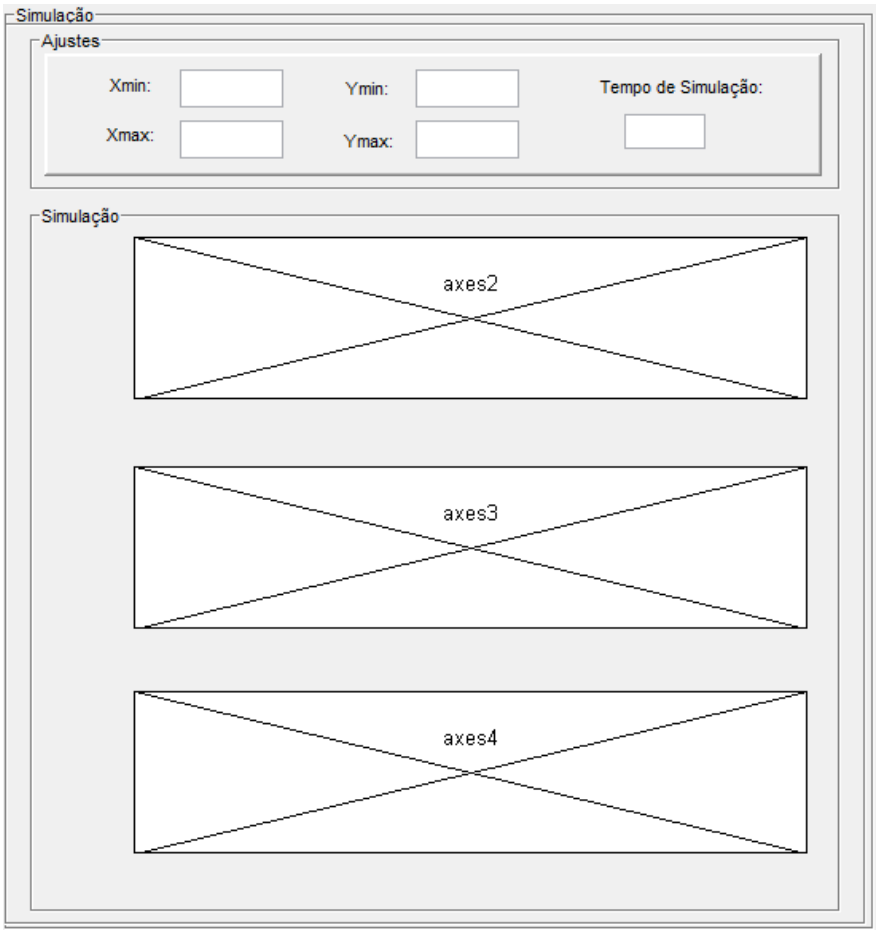
```
            if (F ~= 0)
              K_p = F(:,1:n);
              K_c = F(:,n+1:n+nc);
              nm3 = {'Kp=' K_p 'Kc=' K_c};
              set(handles.edit12,'String',nm3);
              vgc = {'Ganhos calculados!'};
              set(handles.text25,'String',vgc);
            else
              es1 = {'Problema infactível, redefina os
parâmetros.'};
              set(handles.text25,'String',es1);
            end
```

No Quadro 11, um laço *if* verifica se a variável *F* possui valor diferente de 0 e caso verdadeiro, define os vetores *Kp* e *Kc*, ganhos de realimentação associados à planta e ganhos de realimentação associados ao controlador, respectivamente, a partir de *F*. Uma *string* contendo os valores dos ganhos é mostrada ao usuário através do *edit12*, e uma *string* contendo a frase "Ganhos calculados!" é mostrada no *Panel Status* através do elemento *text25*, indicando que o problema foi solucionado. Caso contrário, a frase "Problema infactível, redefina os parâmetros." é mostrada no *text25*, indicando que os parâmetros de projeto e desempenho escolhidos resultaram em um problema infactível.

3.4 ABA SIMULAÇÃO

A Aba Simulação é onde serão exibidos ao usuário os gráficos resultantes da simulação do sistema com o controlador projetado. A Figura 5 mostra a Aba Simulação conforme foi construída no GUIDE.

Figura 5 - Aba Simulação, conforme construída no GUIDE



Fonte: O próprio autor

Conforme mostra a Figura 5, a Aba Simulação foi dividida em dois *Panels*, Ajustes e Gráficos.

O *Panel* Ajustes permite ao usuário ajustar os eixos dos gráficos e o tempo de simulação, sendo composto de cinco elementos *Edit Text*: *Xmin*, *Xmax*, *Ymin*, *Ymax* e *Tempo de Simulação*, com *Tags edit14*, *edit15*, *edit16*, *edit17* e *edit13* respectivamente. O ajuste no eixo do tempo é feito por *Xmin* e *Xmax*, seus valores determinam qual o valor mínimo e máximo do eixo que será mostrado no gráfico, o que possibilita ao usuário visualizar detalhes das curvas quando desejado. *Ymin* e *Ymax* desempenham a mesma função, mas em relação ao eixo das amplitudes. Os ajustes afetam os três gráficos da Aba Simulação, para que o usuário

consiga analisar simultaneamente a saída do sistema, o sinal de controle e o erro de seguimento no segmento de gráfico escolhido. Os comandos responsáveis por essa funcionalidade foram inseridos no *callback* de *edit14* até *edit17*. O Quadro 12 mostra o código utilizado em *edit14*.

Quadro 12 – Código utilizado para ajuste dos limites dos eixos

```
xmax = str2num(get(handles.edit15,'String'));
xmin = str2num(get(handles.edit14,'String'));
set(handles.axes2,'XLim',[xmin xmax]);
set(handles.axes3,'XLim',[xmin xmax]);
set(handles.axes4,'XLim',[xmin xmax]);
```

Conforme observa-se no Quadro 12, armazena-se nas variáveis *xmin* e *xmax* os valores que são inseridos pelo usuário em *edit14* e *edit15*. Em seguida, o comando *set* define que os valores das variáveis definirão os limites do eixo X, ou eixo do tempo neste caso, para os três elementos *Axes* utilizados na Aba simulação. O código acima foi adaptado para o *callback* de *edit16* e *edit17* alterando-se as variáveis, mas preservando a mesma estrutura. Para que não ocorra um erro de variável indefinida ao ser executado o *callback* acima, definiu-se na *opening function* da GUI valores iniciais conforme os comandos mostrados no Quadro 13.

Quadro 13 – Definição de valores *default* para ajuste dos eixos

```
set(handles.edit14,'String','0');
set(handles.edit16,'String','0');
set(handles.edit15,'String','1');
set(handles.edit17,'String','1');
```

O elemento *edit13* ajusta o tempo de simulação por meio da alteração do vetor de tempo que posteriormente é utilizado por outras funções. As linhas de código do Quadro 14 foram utilizadas no *callback* de *edit13.*

Quadro 14 – Definição do vetor de tempo para a simulação

```
tf = str2num(get(handles.edit13,'String'));
t = 0:0.0001:tf;
```

Conforme visto nas linhas de código do Quadro 14, define-se um vetor de dados representando o tempo, com limite superior igual ao valor que é inserido no *edit13*. Isso garante que a simulação será feita respeitando esse limite superior de tempo.

O *Panel* Gráficos mostra os gráficos dos sinais pertinentes ao sistema em malha fechada com o controlador projetado. O primeiro gráfico, na parte superior do *Panel* Gráficos, representa o sinal de saída $y(t)$ obtido ao simular o sistema em malha fechada em conjunto com o sinal de referência escolhido pelo usuário, $r(t)$. O segundo gráfico mostra o sinal de controle $u(t)$ e o terceiro gráfico mostra o erro de seguimento, $e(t) = y(t) - r(t)$. Esses gráficos são exibidos ao usuário por meio de elementos do tipo *Axes*, de *Tags axes2, axes3* e *axes4*, respectivamente.

Para mostrar os gráficos é necessário que se obtenham os sinais definidos, por meio da solução do sistema de equações diferenciais dado pela representação em espaço de estados do sistema em malha fechada. Em MATLAB, tal solução pode ser obtida numericamente por meio da função *ode45* caso o sistema seja de equações diferenciais ordinárias ou por meio da função *dde23* caso o sistema seja de equações diferenciais com atraso. De fato, esses dois casos surgem no contexto dos controladores escolhidos.

Primeiramente, definiu-se no *callback* do elemento *edit13* laços *if* que verificam qual a referência selecionada, resultando em três laços *if*. O vetor de dados que representa a referência é gerado internamente a esses laços de *if*, utilizando-se os parâmetros que o usuário definiu na Aba Sistema e é verificado qual o tipo de controlador selecionado para executar o algoritmo adequado. O trecho de código do Quadro 15 ajuda a entender a estrutura descrita.

Quadro 15 – Trecho do código que verifica qual referência e controlador é selecionado e executa a simulação

```
      vref = get(handles.popupmenu2,'Value');
      tcont = get(handles.popupmenu3,'Value');
      if (vref == 1)
           %define a referencia seno
         (...)
         if (tcont == 1)
           %resolve o sistema e plota os gráficos para ref
seno e controlador repetitivo
         (...)
         if (tcont == 2)
           %resolve o sistema e plota os gráficos para ref
seno e controlador ressonante
         (...)
      if (vref == 2)
         %define a referencia triangular
         if (tcont == 1)
         %resolve o sistema e plota os gráficos para ref
triangular e controlador repetitivo
      (...)
```

No Quadro 15, *vref* e *tcont* armazenam o tipo de referência e de controlador, através do parâmetro *Value* do *popupmenu2* e *popupmenu3*. Cada *Value* corresponde a um tipo de referência e de controlador, conforme o que já foi exposto anteriormente.

Para definir-se a referência internamente, utilizam-se linhas de código como as mostradas no Quadro 16.

As linhas do Quadro 16 definem uma referência seno, e podem ser facilmente adaptadas para as demais referências. A amplitude, frequência, tempo de inicio e fase da referencia são armazenados em variáveis, a partir dos valores que o usuário forneceu nos elementos *edit3* até *edit6* da Aba Sistema. A partir do vetor de tempo definido anteriormente, cria-se um vetor de dados para a referência, que tem início em *tsen*.

Quadro 16 – Definição da referência seno, conforme parâmetros fornecidos pelo usário

```
        ampsen = str2num(get(handles.edit3,'String'));
        fsen = str2num(get(handles.edit4,'String'));
        tsen = str2num(get(handles.edit5,'String'));
        phisen = str2num(get(handles.edit6,'String'));
        r = ampsen*sin(fsen*t + phisen);
        rs = r.*(t>=tsen)
```

Uma vez montada a estrutura que verifica o tipo de controlador e gera as referências, definem-se os comandos que resolvem o sistema de equações no espaço de estados e plotam os gráficos. Para o controlador ressonante e o controlador integral, a descrição no espaço de estados do sistema aumentado que contempla a união entre controlador e planta é um sistema de equações diferenciais ordinárias, e é resolvido através da função *ode45*. Para o controlador repetitivo e o controlador ressonante-repetitivo, o sistema aumentado é um sistema de equações diferenciais com atraso e é resolvido utilizando-se a função *dde23.*

As funções *ode45* e *dde23* utilizam o método de Runge-Kutta para resolver numericamente as equações (SHAMPINE;THOMPSON, 2000). As linhas de código do Quadro 17 exemplificam a utilização da *ode45* para solução do sistema em malha fechada com controlador ressonante.

Quadro 17 – Código que executa a *ode45* para resolver o sistema de EDO e calcula o sinal de controle e de erro

```
            tspan = t;
            iniCon = 0*[10;10;zeros((length(Kc)),1)];
            [t,dx] = ode45(@sys_multiressonante_tr, tspan,
iniCon);
            Ced = evalin('base','Ced');
            F = [Kp Kc]*dx';
            E = rs - Ced*dx'
```

As linhas do Quadro 17 mostram a forma padrão para se utilizar a função *ode45*, definindo-se primeiramente *tspan*, que representa o vetor de tempo e *iniCon*, as condições inicias do sistema de EDO (definidos como um vetor de zeros, pois deseja-se obter a resposta em regime permanente do sistema). A função *ode45* então é chamada com um argumento que representa o sistema a ser resolvido e o vetor de tempo e das condições iniciais. A solução obtida pela *ode45* é então utilizada para calcular o erro de seguimento. A expressão *sys_multiressonante_tr* refere-se a um arquivo .m de mesmo nome no qual estão definidas as equações que representam o sistema em malha fechada para o controlador escolhido bem como uma referência que é gerada interna ao arquivo *sys_multiressonante_tr* para evitar problemas com variáveis utilizadas nos métodos iterativos da *ode45* e *dde23*. As linhas de código mais relevantes do arquivo *sys_multiressonante_tr.m* estão no Quadro 18.

Quadro 18 – Linhas de código do arquivo *sys_multiressonante_tr.m*

```
function  dx = sys_multiressonante_sen(t,x)

Aa = (Aed + Bed*[Kp Kc]);

B1 = [zeros(n,1);Bc];

B2 = [Bw;zeros(nc,1)];

Bb = B1*rs' + B2*wd;

dx = Aa*x + Bb;
```

As linhas do Quadro 18 são simplesmente a representação em espaço de estados do sistema escritas em linhas de código. Para plotar os sinais encontrados a partir da solução da *ode45* utilizam-se linhas de código conforme mostra o Quadro 19.

Define-se em qual gráfico a função *plot* irá desenhar a curva de interesse e qual os nomes dos eixos e da legenda do gráfico e habilita-se o *grid*.

A função *dde23* é exemplificada nas linhas de código do Quadro 20, utilizadas no arquivo .m de nome *solv_rep_sen.m* que define uma função que resolve o sistema de equações com atraso para o controlador repetitivo e com referência senoidal.

Quadro 19 – Exemplo de código que plota os gráficos dos sinais obtidos na simulação

```
axes(handles.axes2);
plot(t,rs,t,Ced*dx');
legend('Referência','Saída');
ylabel(handles.axes2,'Amplitude');
xlabel(handles.axes2,'t(s)');
grid on
```

Quadro 20 – Exemplo da função *dde23* utilizada no arquivo *solv_rep_sen.m*

```
function sol = solv_rep_sen
history = zeros(length([Kp Kc]),1);
tspan = [0,0.1];
sol = dde23(@solv_rep_senf,tau,history,tspan);
plot(sol.x,sol.y(1,:),sol.x,rs_a);


function yp = solv_rep_senf(t,y,Z)
wd = rs;
Aa = Aed + Bed*[Kp Kc];
B1 = [zeros(n,1);Bc];
B2 = [Bw;zeros(nc,1)];
yp = Aa*y + Ad*Z + B1*rs + B2*wd;
```

O funcionamento do código mostrado no Quadro 20 pode ser explicado de maneira análoga ao caso da *ode45*, exceto que no caso da *dde23* a função que resolve a equação diferencial com atraso está definida no mesmo arquivo .m onde é definido o sistema. A expressão *function yp = solv_rep_senf(t,y,Z)* define a função onde será especificado o

sistema, enquanto a expressão *function sol = solv_rep_sen* define a função que retorna a solução do sistema de equações.

Para que a GUI execute o arquivo .m especificado, é necessário que se insira uma linha de código chamando tal arquivo, mostrada no Quadro 21.

Quadro 21 – Comando que executa o arquivo *solv_rep_sen.m*

```
sol = solv_rep_sen;
```

Os gráficos são plotados utilizando-se as mesmas linhas de código especificadas anteriormente para a função *ode45*.

4. RESULTADOS

Neste capítulo serão mostrados os resultados obtidos quando se utiliza a GUI desenvolvida para projetar os ganhos de realimentação e simular o sistema com o controlador escolhido em malha fechada. Utilizou-se como referência o sistema UPS descrito em (LORENZINI, 2015) cuja descrição em espaço de estados tem a forma da Equação (1) e as matrizes que completam a descrição no espaço de estados possuem valores numéricos conforme mostrado abaixo:

$$\mathrm{A} = \begin{bmatrix} -15 & -1000 \\ 3333{,}33 & 0 \end{bmatrix}, B = \begin{bmatrix} 1000 \\ 0 \end{bmatrix}, B_w = \begin{bmatrix} 0 \\ -3333{,}33 \end{bmatrix}, C = [0 \quad 1]. \tag{43}$$

As seções a seguir mostram os resultados obtidos utilizando as referências seno e triangular para os controladores ressonante, repetitivo e ressonante-repetitivo e *step* para o controlador de ação integral.

4.1 CONTROLADOR RESSONANTE

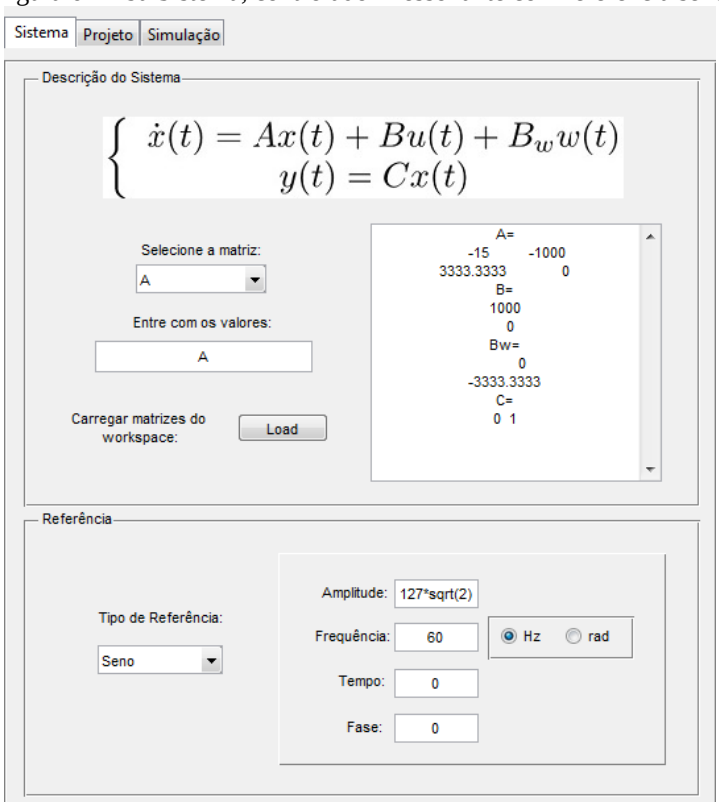
4.1.1. Referência Seno

Para o controlador ressonante com referência senoidal, inicia-se definindo as matrizes dadas pela Equação (42). Escolhe-se a referência seno, com amplitude de $127\sqrt{2}$ e frequência de 60Hz, por coincidir com amplitude e frequência da tensão elétrica com as quais normalmente trabalha um sistema UPS como o que é descrito pelas matrizes da Equação (1). O tempo em que a referência passará a assumir valores foi definido como zero, assim como sua fase. A Figura 6 mostra a aba Sistema após os valores citados serem definidos.

Na Aba Projeto, seleciona-se o Controlador Ressonante no menu *Pop-Up* e após define-se os parâmetros de desempenho e projeto. Escolheram-se os valores numéricos 600, 6000, π/2 e 4 para Alfa, r, Theta e Nº de harmônicas respectivamente, por se tratarem de valores que tornam factível o cálculo dos ganhos de realimentação. Após pressionar o botão Calcular, os ganhos calculados são exibidos ao usuário, enquanto a *Status Bar* exibe a frase "Ganhos Calculados!", conforme mostra a Figura 7.

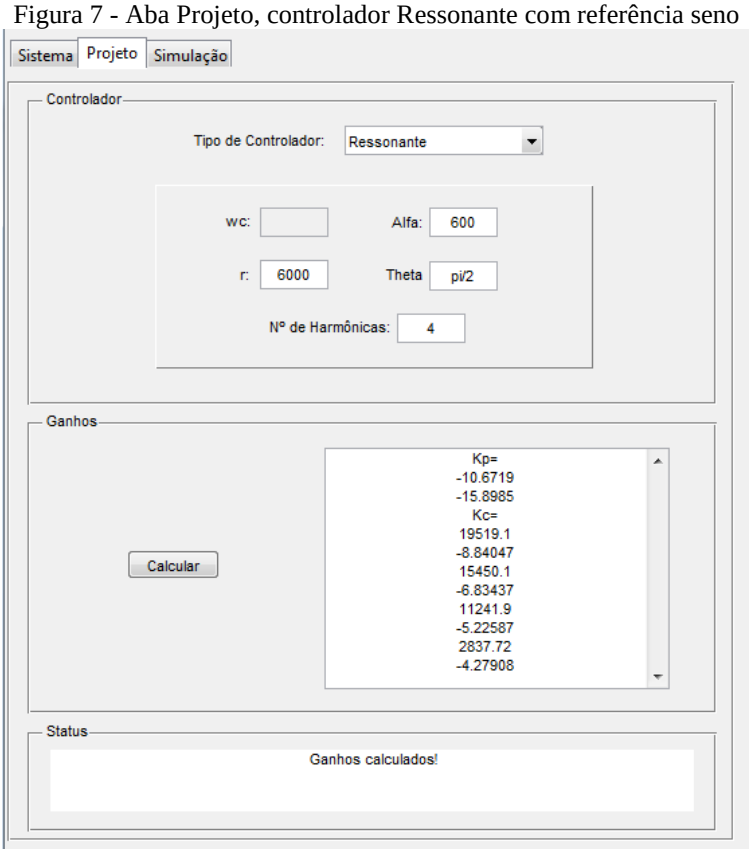
Para simular o sistema definido com o controlador projetado, definiu-se o tempo de simulação 0,1s para contemplar apenas alguns ciclos do sinal de saída, deixando-se os ajustes dos eixos no valor default. O resultado pode ser visto na Figura 8, que mostra os gráficos do sinal de saída e referência (diferenciados pela legenda), sinal de controle e erro.

Figura 6 - Aba Sistema, controlador Ressonante com referência seno



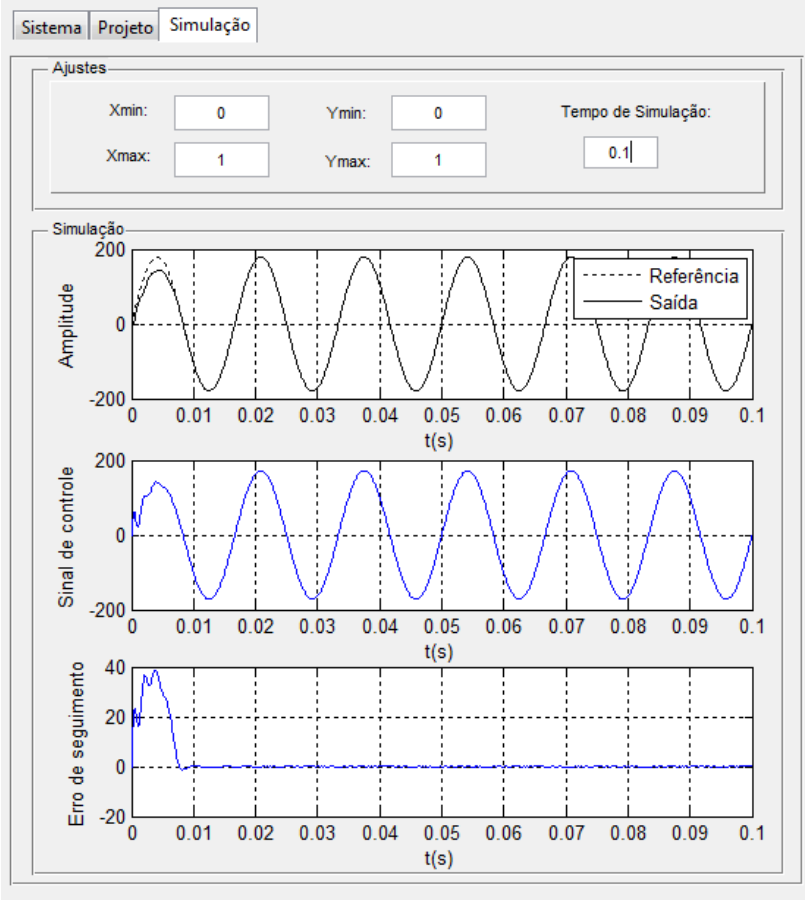
Fonte: O próprio autor

Figura 7 - Aba Projeto, controlador Ressonante com referência seno



Fonte: O próprio autor

Figura 8 - Aba Simulação, controlador Ressonante com referência seno



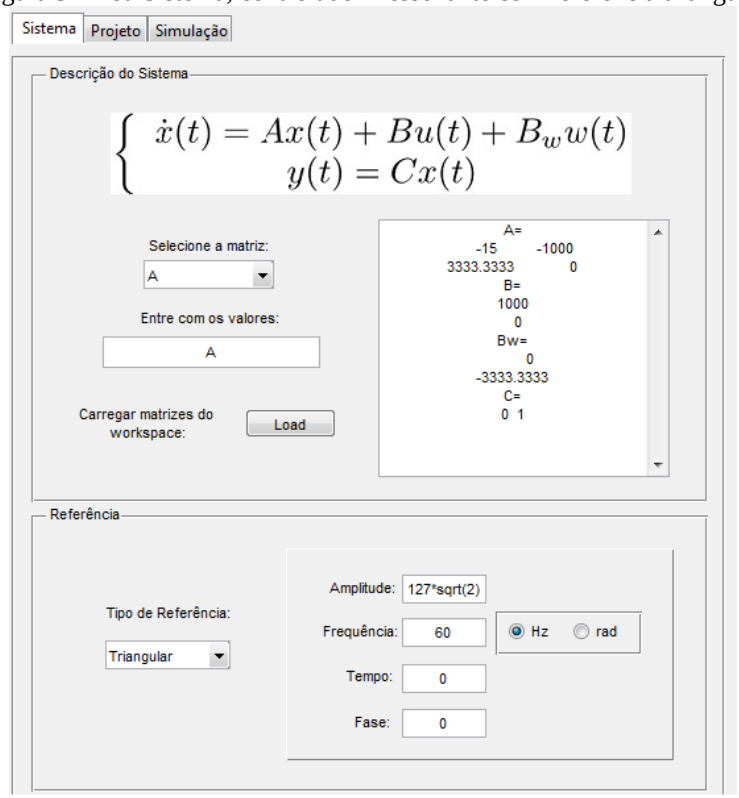
Fonte: O próprio autor

4.1.2. Referência Triangular

Para o controlador ressonante com referência triangular, escolhe-se a referência triangular, com amplitude de $127\sqrt{2}$ e frequência de 60Hz, semelhante ao que foi feito para a referência senoidal. O tempo em que a referência passará a assumir valores foi definido como 0, assim como sua fase. A Figura 9 mostra a aba Sistema após os valores citados serem definidos.

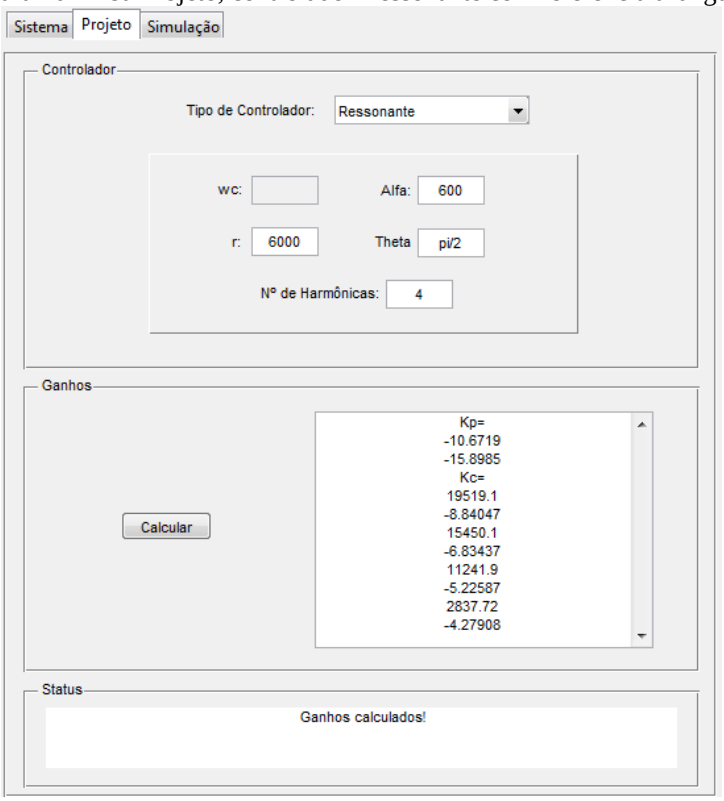
Na Aba Projeto, definiram-se os valores numéricos 600, 6000, π/2 e 4 para Alfa, r, Theta e Nº de harmônicas respectivamente, pelos motivos citados na subseção anterior. A Figura 10 mostra o resultado obtido ao pressionar-se o botão Calcular. A Figura 11 mostra a Aba Simulação com os gráficos do sinal de referência, saída, controle e erro de seguimento para um tempo de simulação de 0,1s.

Figura 9 - Aba Sistema, controlador Ressonante com referência triangular



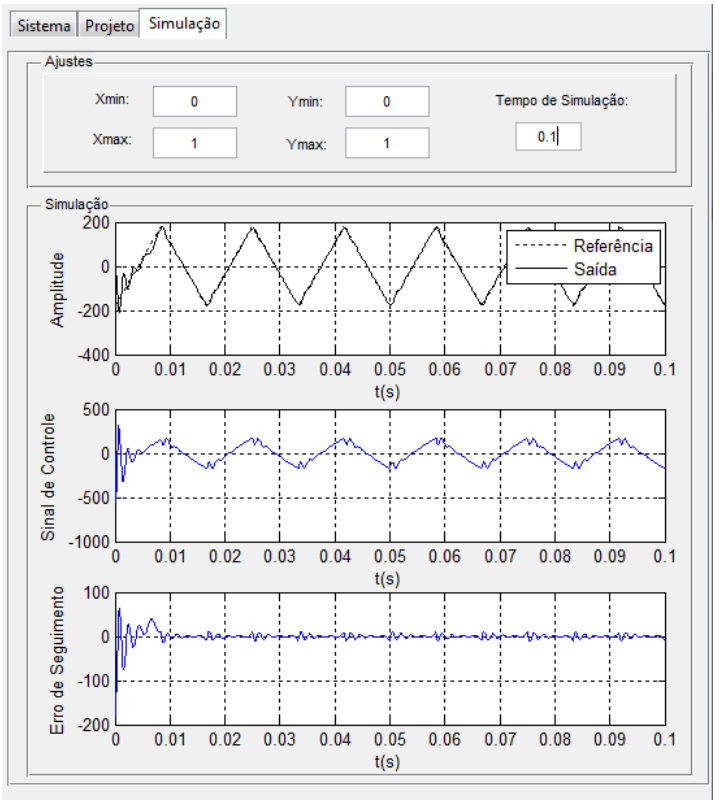
Fonte: O próprio autor

Figura 10 - Aba Projeto, controlador Ressonante com referência triangular



Fonte: O próprio autor

Figura 11. Aba Simulação, controlador Ressonante com referência triangular



Fonte: O próprio autor

4.2 CONTROLADOR REPETITIVO

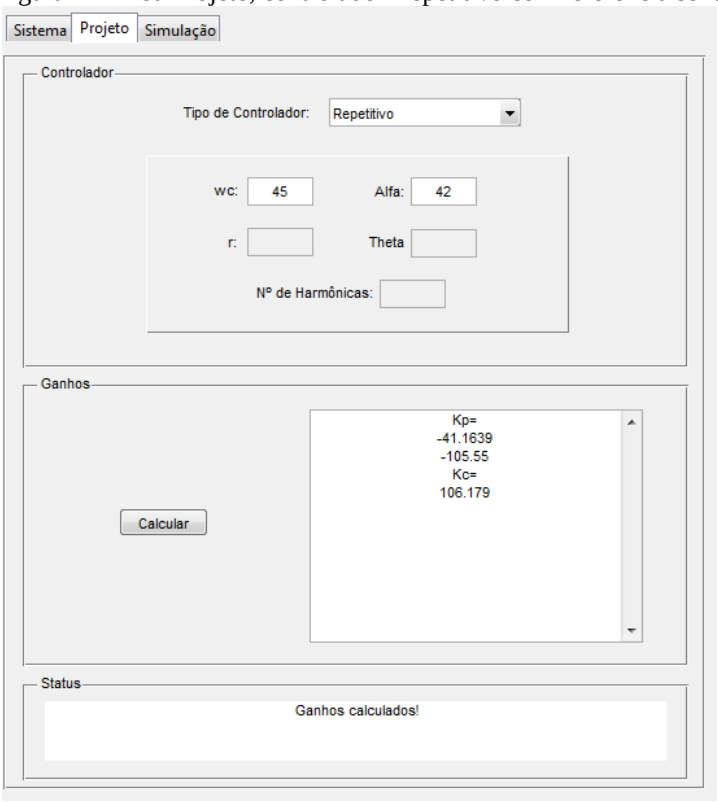
4.2.1 Referência Seno

A Aba Sistema, no projeto do controlador repetitivo com referência seno, foi definida de maneira idêntica ao controlador ressonante, com uma referencia senoidal de 60Hz, amplitude de $127\sqrt{2}$ e tempo de inicio e fase igual a 0, conforme pode ser visto na Figura 6.

Na Aba Projeto, definiu-se ω_c igual a 45 e Alfa igual a 42 por serem valores que tornam o problema factível. Ao pressionar o botão Calcular, as informações exibidas ao usuário podem ser vistas na Figura 12.

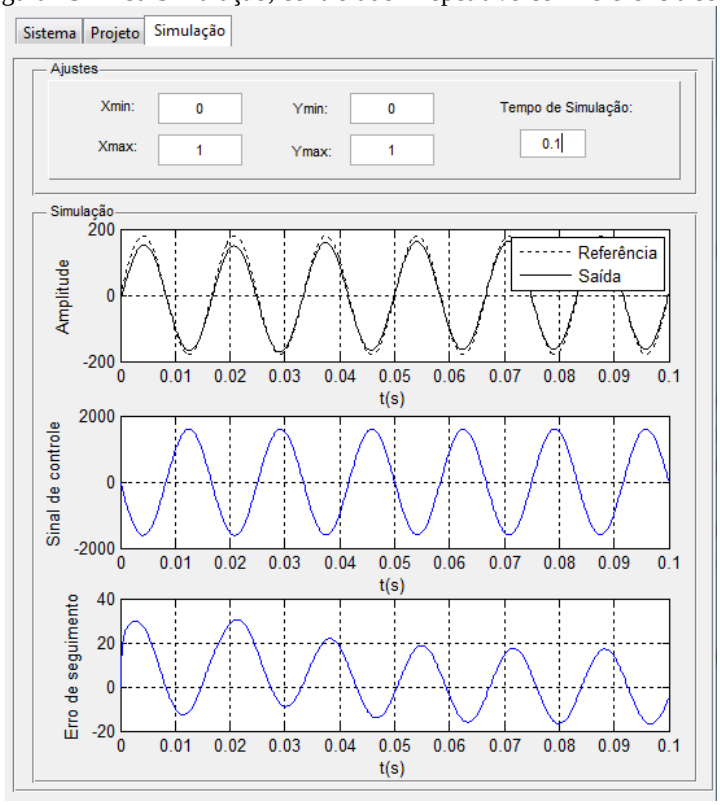
A Figura 13 mostra a aba Simulação, com os gráficos do sinal de referência, saída, controle e erro de seguimento para um tempo de simulação de 0,1s e demais parâmetros deixados em seu valor *default*.

Figura 12 - Aba Projeto, controlador Repetitivo com referência seno



Fonte: O próprio autor

Figura 13 - Aba Simulação, controlador Repetitivo com referência seno



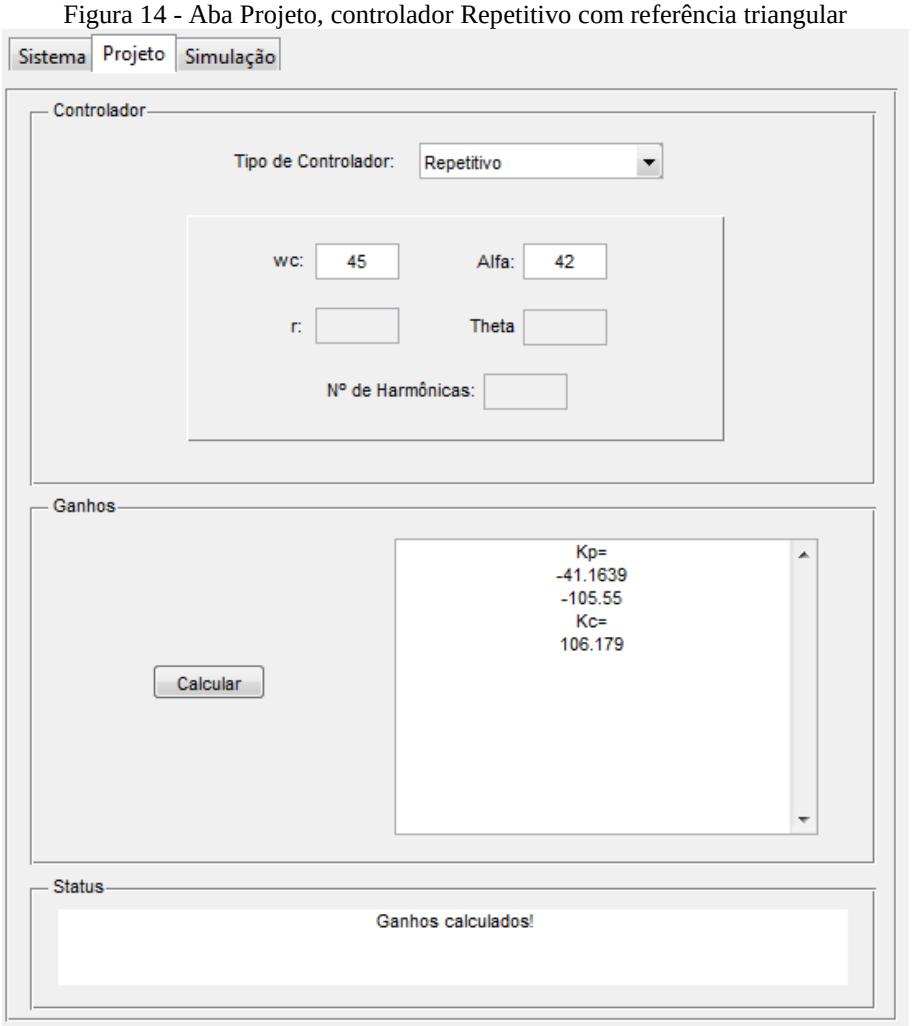
Fonte: O próprio autor

4.2.2 Referência Triangular

Com referência triangular, a Aba Sistema para o controlador Repetitivo é idêntica a Aba Sistema para o controlador Ressonante, pois se escolheu uma referência triangular com os mesmos valores de amplitude, frequência, tempo de início e fase, conforme mostrado na Figura 9.

Na Aba Projeto, escolhe-se o controlador Repetitivo, e define-se se ω_c igual a 45 e Alfa igual a 42, como anteriormente. Ao pressionar-se o botão Calcular, o resultado é o que pode ser visto na Figura 14.

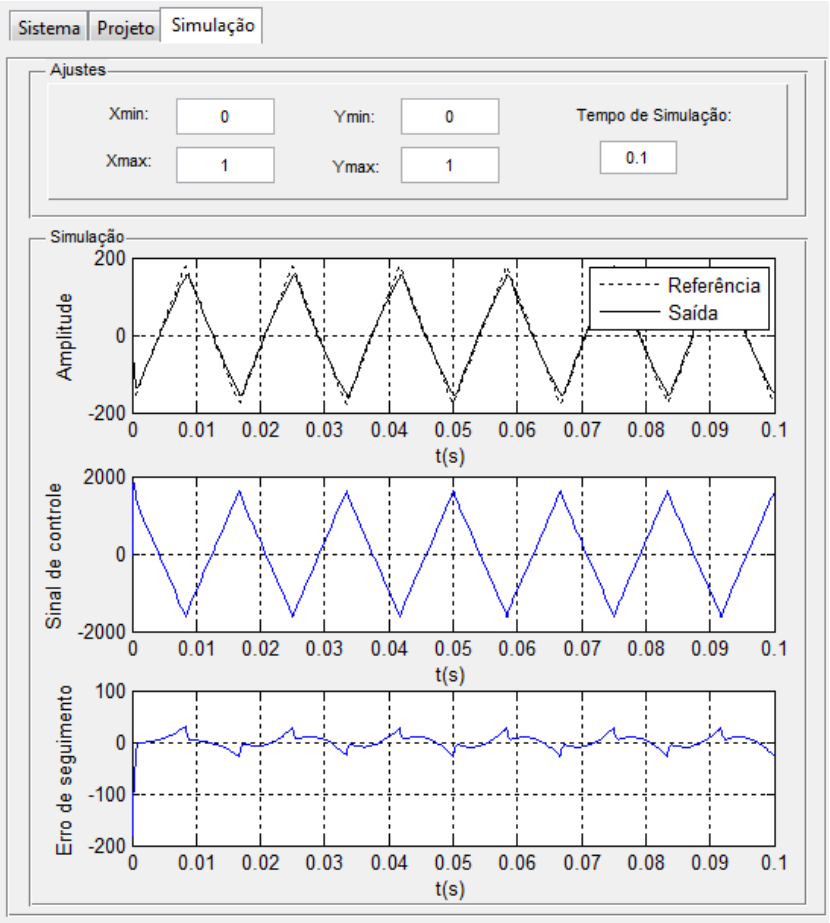
Figura 14 - Aba Projeto, controlador Repetitivo com referência triangular



Fonte: O próprio autor

A Figura 15 mostra a aba Simulação, com os gráficos do sinal de referência, saída, controle e erro de seguimento para um tempo de simulação de 0,1s e demais parâmetros deixados em seu valor *default.*

Figura 15 - Aba Simulação, controlador Repetitivo com referência triangular



Fonte: O próprio autor

4.3 CONTROLADOR RESSONANTE-REPETITIVO

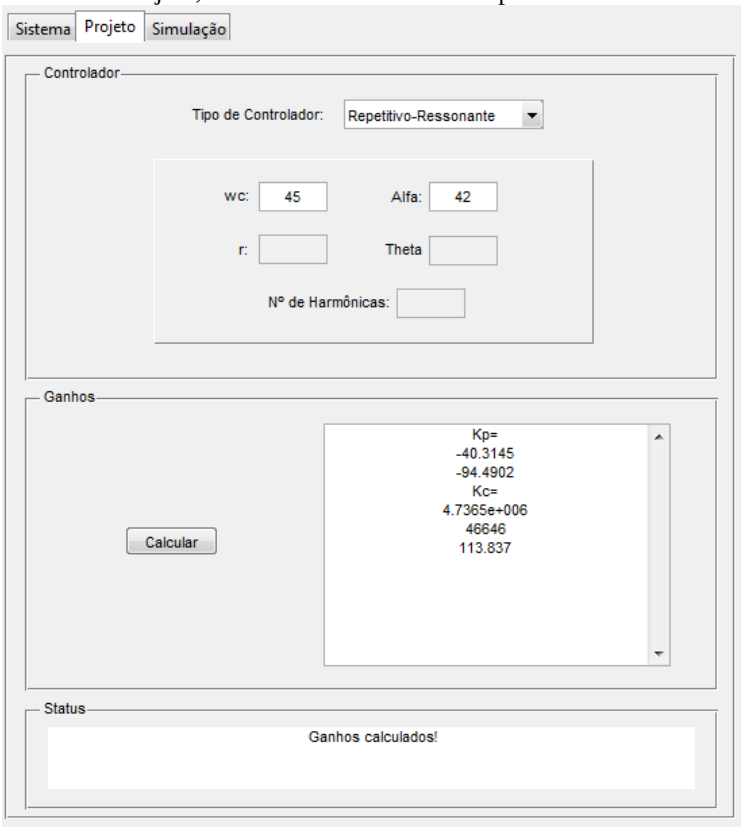
4.3.1 Referência Seno

Com o controlador Ressonante-Repetitivo, as entradas da Aba Sistema são definidas de forma idêntica aos controladores Ressonante e Repetitivo, conforme pode ser visto na Figura 6.

Na Aba Projeto, novamente escolhe-se os valores 45 e 42 para ω_c e Alfa, por implicarem num problema factível. Os ganhos exibidos ao usuário podem ser vistos na Figura 16.

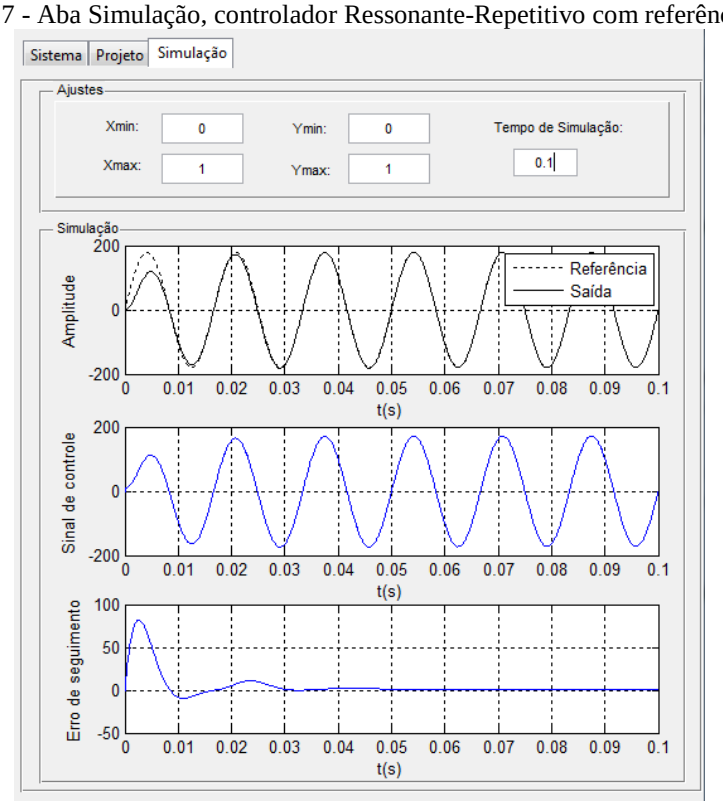
A Figura 17 mostra a Aba Simulação com os gráficos do sinal de referência, saída, controle e erro de seguimento para um tempo de simulação de 0,1s e os ajustes dos eixos deixados em seu valor *default*.

Figura 16 - Aba Projeto, controlador Ressonante-Repetitivo com referência seno



Fonte: O próprio autor

Figura 17 - Aba Simulação, controlador Ressonante-Repetitivo com referência seno



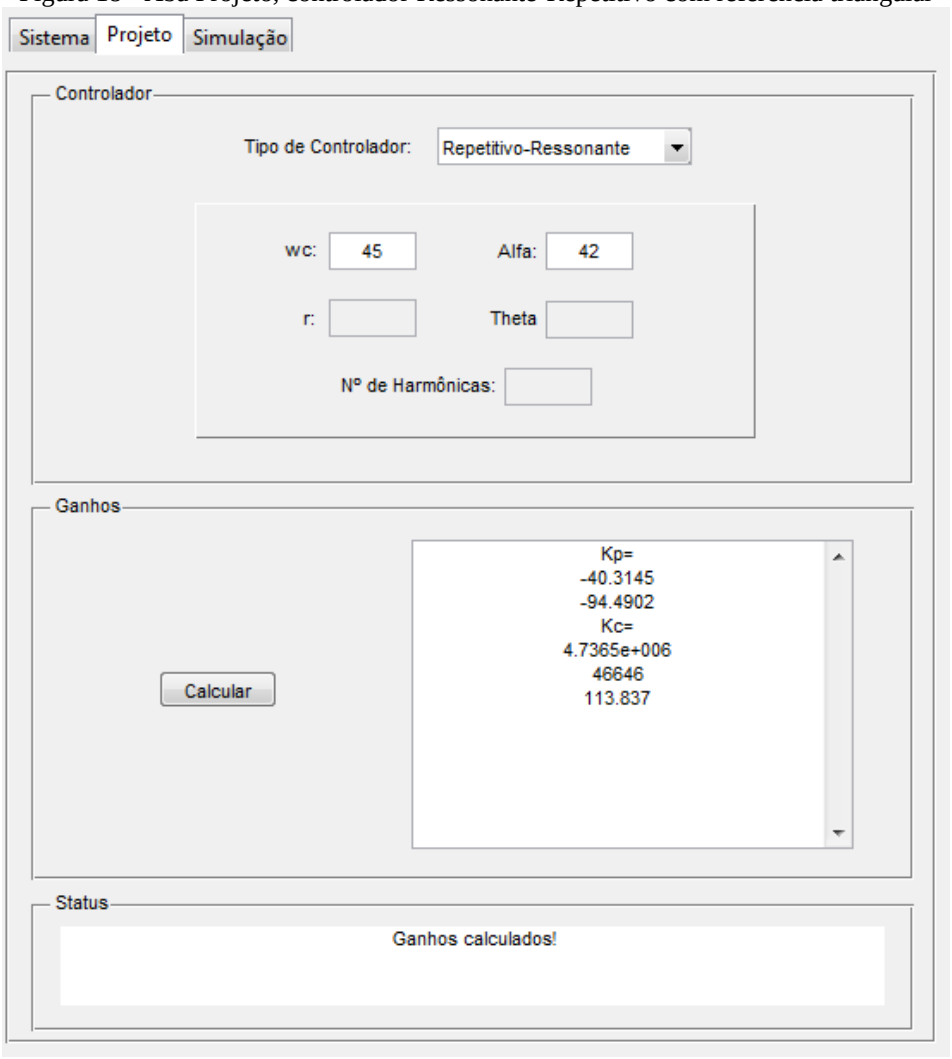
Fonte: O próprio autor

4.3.2 Referência Triangular

Com referência triangular, para o controlador Ressonante-Repetitivo, as entradas da Aba Sistema são definidas de forma idêntica aos controladores Ressonante e Repetitivo, conforme pode ser visto na Figura 9.

Na Aba Projeto, novamente escolhe-se os valores 45 e 42 para ω_c e Alfa, por implicarem num problema factível. Ao pressionar o botão Calcular, os ganhos são exibidos, conforme mostra a Figura 18.

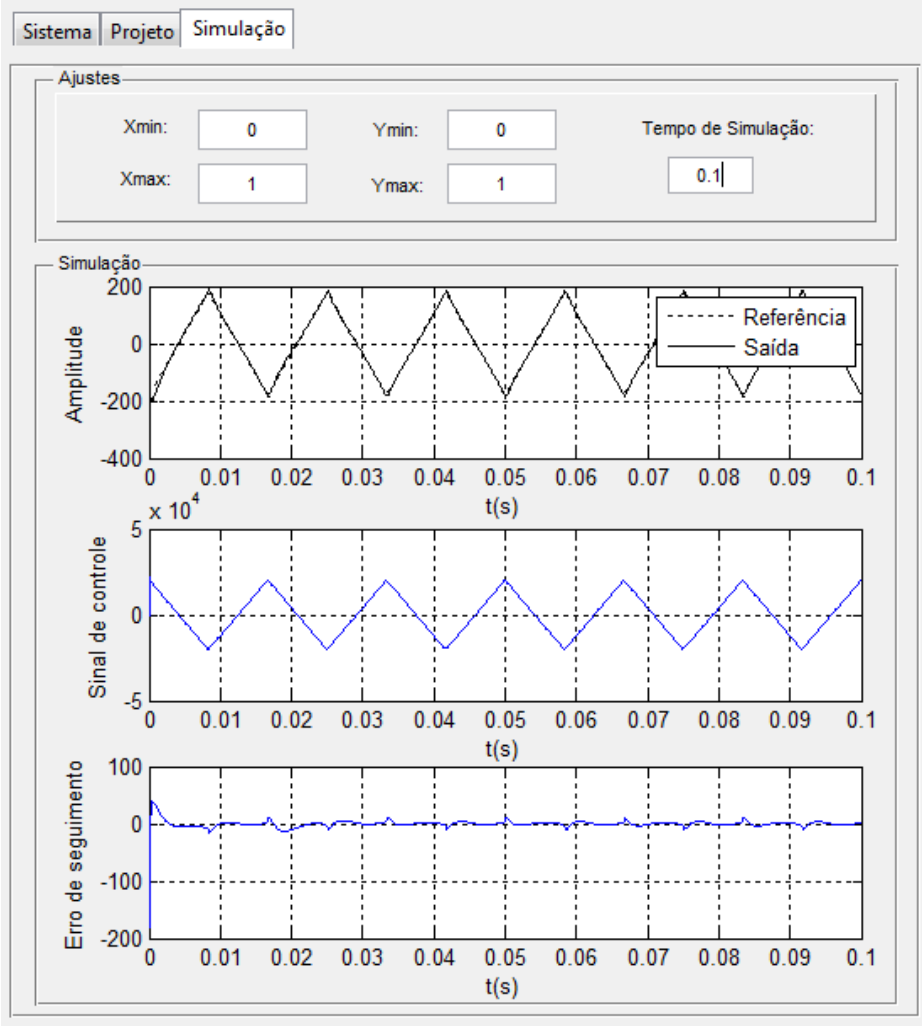
Figura 18 - Aba Projeto, controlador Ressonante-Repetitivo com referência triangular



Fonte: O próprio autor

A Figura 19 mostra a Aba Simulação com os gráficos do sinal de referência, saída, controle e erro de seguimento para um tempo de simulação de 0,1s e demais ajustes deixados em seu valor *default.*

Figura 19 - Aba Simulação, controlador Ressonante-Repetitivo com referência triangular



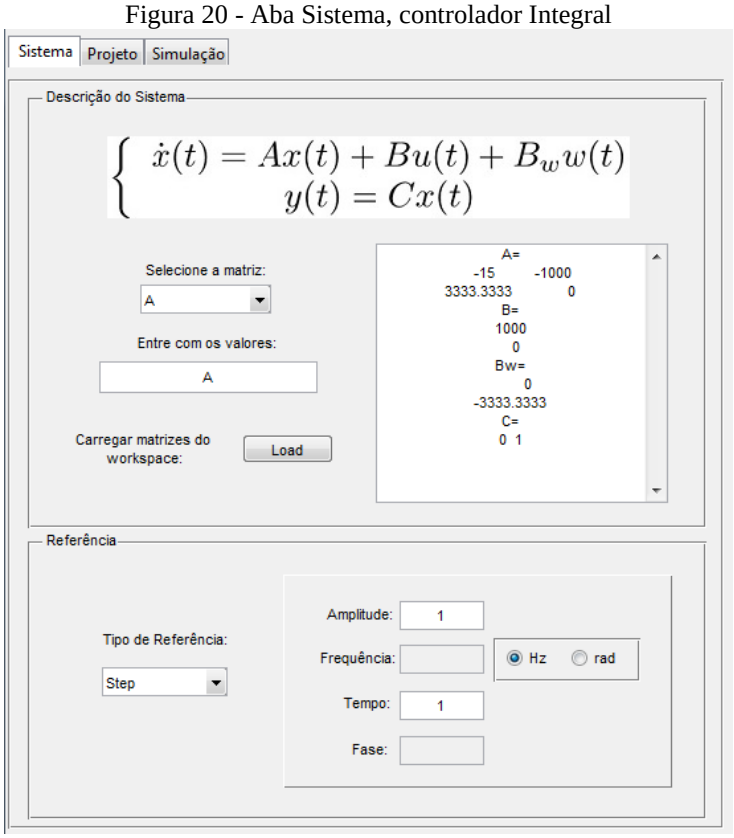
Fonte: O próprio autor

4.4 CONTROLADOR DE AÇÃO INTEGRAL

Para o controlador de Ação Integral, definiram-se na Aba Sistema as matrizes dadas pela Equação (42) e escolheu-se a referência *step*, com amplitude e tempo de inicio igual a 1. A Figura 20 mostra a Aba Sistema após a definição desses valores.

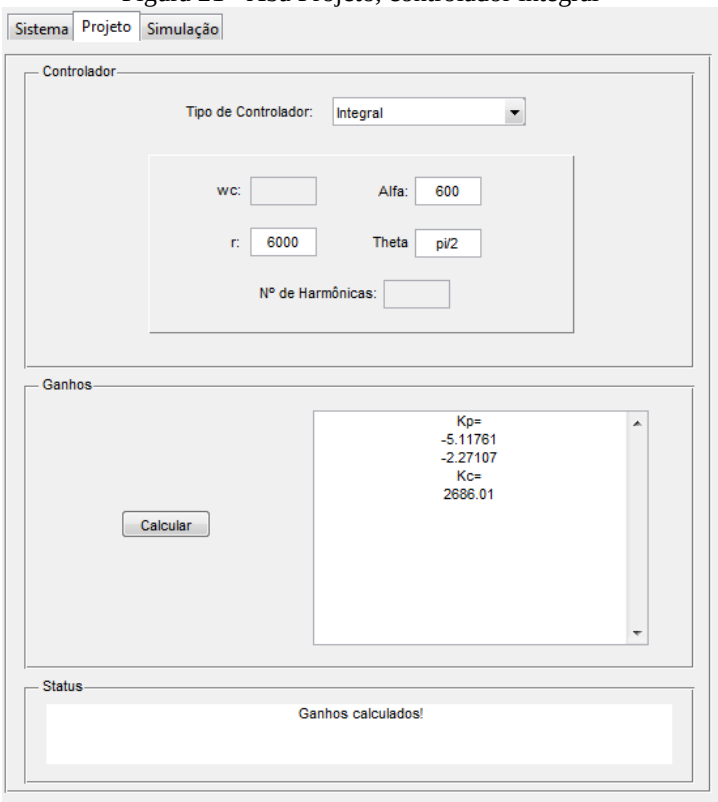
Na Aba Projeto, definiram-se os valores 600, 6000 e $\pi/2$ para Alfa, r e Theta respectivamente, por tornarem o problema (do cálculo dos ganhos de realimentação) factível. A Figura 21 mostra a Aba Projeto, enquanto a Figura 22 mostra a Aba Simulação, com tempo de simulação ajustada para 2s e os limites do eixo do tempo ajustados para 0,98s e 1,1s, enquanto os limites do eixo da amplitude são deixados no valor *default.*

Figura 20 - Aba Sistema, controlador Integral



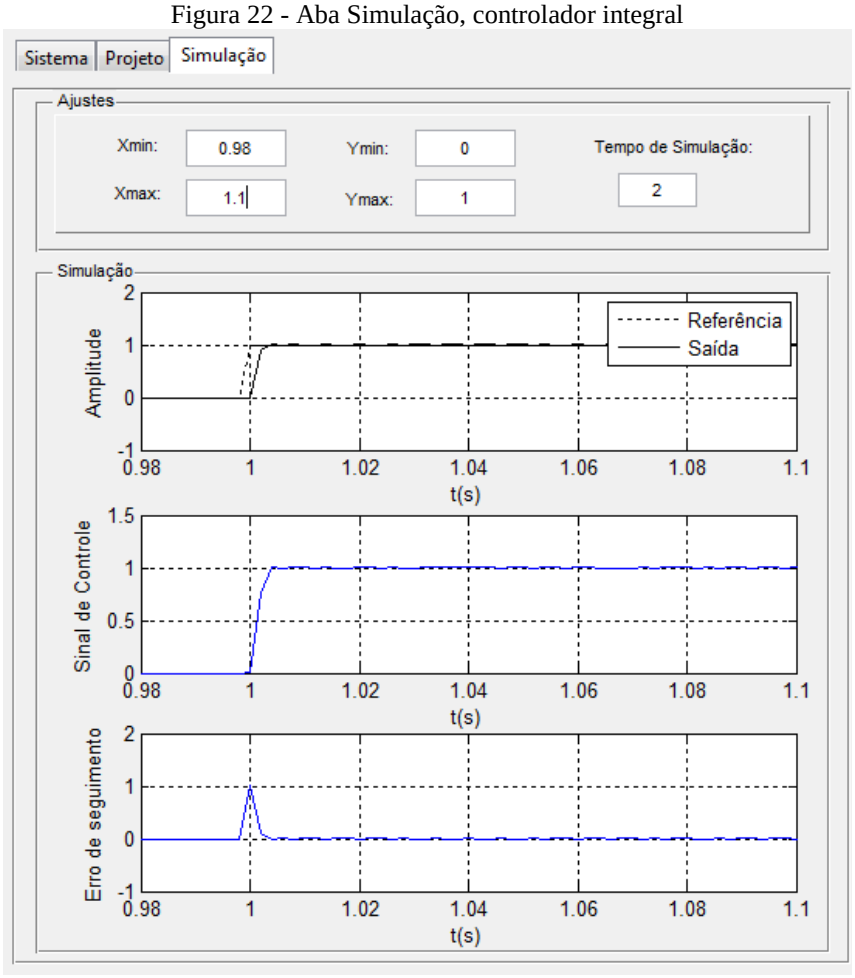
Fonte: O próprio autor

Figura 21 - Aba Projeto, controlador integral



Fonte: O próprio autor

Figura 22 - Aba Simulação, controlador integral



Fonte: O próprio autor

5. CONCLUSÃO

A interface gráfica construída apresenta ao usuário um *layout* simples e limpo, ao mesmo tempo incorporando os principais elementos necessários para a finalidade para a qual foi proposta: possibilitar o projeto e simulação de controladores com referências periódicas, sem exigir do usuário conhecimento profundo dos métodos de projeto envolvidos.

As três abas propostas no Capítulo 3 seguem os objetivos apresentados naquele capítulo. A Aba Sistema permite a entrada dos valores das matrizes manualmente ou por meio do botão *Load* que carrega as matrizes se elas estiverem na *workspace*. A escolha da referência é feita de forma simples, selecionando-se o tipo e os parâmetros da mesma.

Na Aba Projeto, o tipo de controlador é escolhido pela seleção em um menu e os parâmetros de projeto e desempenho são definidos pelo usuário, enquanto os ganhos são calculados e então exibidos ao usuário ao pressionar-se o botão Calcular. A barra de *status* mostrou-se de fundamental importância, pois permite ao usuário visualizar informações sobre a factibilidade do problema que deve ser resolvido.

Na Aba Simulação pode-se comprovar que os ganhos calculados na Aba Projeto resultam em um sistema capaz de seguir a referência definida com erro praticamente nulo em regime permanente, conforme pode ser verificado nas figuras da Aba Simulação, mostradas no Capítulo 4.

Alguns problemas de desempenho foram observados ao executar-se a GUI sob determinadas condições especificas. O *TabManager* pode demorar para atualizar os elementos gráficos quando é realizada a troca de abas, ocasionando uma ligeira sobreposição dos elementos da aba atual e da aba a ser selecionada. Na Aba Simulação, a escolha de um tempo de simulação elevado pode resultar num vetor de tempo com excessivo número de pontos que implica em um tempo de geração dos gráficos por parte do MATLAB que é proporcional ao tempo de simulação, causando atrasos ou até mesmo travamento do *software* caso se definam valores demasiadamente grandes.

A partir do que foi exposto acima, ficam propostas algumas sugestões de trabalhos futuras visando melhorar o presente projeto:

- Permitir que se considere uma perturbação a ser rejeitada pelo sistema, que seja diferente de zero.
- Transformar a GUI em um programa *standalone*, ou seja, um programa autônomo que funcione sem dependência do MATLAB.

- Permitir a combinação dos controladores Ressonante, Repetitivo e Ressonante-Repetitivos com uma Ação Integral.
- Melhorar o desempenho da Aba Simulação e adicionar algumas funcionalidades que facilitem a interação do usuário com a GUI, como novas barras de *status* e ferramentas de ajuda.
- Permitir que o usuário defina outras referências periódicas além das já consideradas.

REFERÊNCIAS

BAZANELLA, A. S.;SILVA, J. M. G. da. **Ajuste de Controladores PID.** Disponível em: < http://www.ece.ufrgs.br/~jmgomes/pid/Apostila/apostila/node22.html>. Acesso em: 7 set. 2016.

CHEN, C. **Linear system theory and design.** 3. ed. NewYork: Oxford University Press, 1999.

CHILALI,M;GAHINET,P. H_∞design with pole placement constraints: an LMI approach. **IEEE Transaction on Automatic Control,** New York, v41, n. 3, p.358-367, Mar. 1996.

DAVIDSON, G. **MathWorks File Exchange.** Disponível em: < https://www.mathworks.com/matlabcentral/fileexchange/54705-tabmanager-create-tab-panels--uitabgroup--from-a-guide-gui>. Acesso em: 25 set. 2016.

ESCOBAR, G.;VALDEZ, A. A.;LEYVA-RAMOS, J.;MATTAVELLI, P. Repetitive-based controller for a UPS inverter to compensate unbalance and harmonic distortion. **IEEE Transactions on Industrial Electronics.** v54, n. 1. p.504-510. Feb. 2007.

FLORES, J. V. **Projeto de controladores para o seguimento de referências periódicas em sistemas com atuadores saturantes.** 2012. 138f. Tese (Doutorado) – Universidade Federal do Rio Grande do Sul. Escola de Engenharia. Programa de Pós-Graduação em Engenharia Elétrica, Porto Alegre, 2012.

LORENZINI, C. **Desenvolvimento de um controlador Ressonante-Repetitivo aplicado a fontes ininterruptas de energia.** 2015. 117f. Dissertação (Mestrado) – Universidade Federal do Rio Grande do Sul. Escola de Engenharia. Programa de Pós-Graduação em Engenharia Elétrica, Porto Alegre, 2015.

MATHWORKS. MATLAB® Creating Graphical User Interfaces. Natick, MA: The MathWorks Inc, 2015.

PEREIRA, L. F. A.;CARVALHO, F. M.;FLORES, J. V. Alternative resonant controller design for Uninterruptable Power Supplies (UPS). In: ANNUAL CONFERENCE OF THE IEEE INDUSTRIAL ELECTRONICS SOCIETY (IECON'13), 39., 2013, Vienna. **Proceedings…** New York: IEEE, 2013. p.3311-3316.

ROHOUMA, W.; ZANCHETTA, P.; WHEELER, P.;EMPRINGHAM, L. A four-leg matrix converter ground power unit with repetitive voltage control. **IEEE Transactions on Industrial Electronics.** v62, n. 4. p.2032-2040. Dec. 2014.

SALTON, A. T. et al. A resonant-repetitive control scheme applied to uninterruptable power supplies (UPS). **Journal of Control, Autmation and Electrical Systems,** New York, v.24, n.3, p.253-262, Apr. 2013.

SHAMPINE, L. F.;THOMPSON, S. Solving Delay Differential Equations with *dde23.* Radford University, 2000. 44p.

WU, M.;XU, B.;CAO, W.;SHE, J. Aperiodic disturbance rejection in repetitive control systems. **IEEE Transactions on Control Systems Technology.** v.22, n. 3, p.1044-1051. 2014.

YAO, W.-S.;TSAI, M.-C.;YAMAMOTO, Y. Implementation of repetitive controller for rejection of position-based periodic disturbances. **Control Engineering Practice.** v. 21, n. 9. p. 1226-1237. 2013.