MINISTÉRIO DA EDUCAÇÃO

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Escola de Engenharia

Programa de Pós-graduação em Engenharia de Minas, Metalúrgica e de Materiais-PPGE3M

SOFTWARE EDUCATIVO PARA O ESTUDO DE CRITÉRIOS E ENVELOPES DE FALHA EM MATERIAIS COMPÓSITOS

Julian Gamboa

Dissertação para obtenção do Título de Mestre em Engenharia

Porto Alegre

2018

MINISTÉRIO DA EDUCAÇÃO

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Escola de Engenharia

Programa de Pós-graduação em Engenharia de Minas, Metalúrgica e de Materiais-PPGE3M

SOFTWARE EDUCATIVO PARA O ESTUDO DE CRITÉRIOS E ENVELOPES DE FALHA EM MATERIAIS COMPÓSITOS

Julian Gamboa

Engenheiro Eletrônico

Trabalho realizado no Laboratório de Materiais Poliméricos (LAPOL) do Departamento de Materiais da Escola de Engenharia da UFRGS dentro do Programa de Pós-Graduação em Engenharia de Minas, Metalúrgica e de Materiais – PPGE3M, como parte dos requisitos para a obtenção do título de Mestre em Engenharia.

Área de concentração: Ciência e Tecnologia dos Materiais

Porto Alegre

2018

Esta Dissertação foi julgada adequada para obtenção do título de Mestre em Engenharia, área de concentração Ciência e Tecnologia dos Materiais, e aprovada em sua forma final pelo Orientador e pela Banca Examinadora do Curso de Pós-Graduação.

Orientador: Prof. Dr. Sandro Campos Amico (UFRGS)

Banca Examinadora:

Profa. Dra. Letícia Rocha Machado
UFRGS/Centro Interdisciplinar de Novas Tecnologias na Educação, Núcleo de Tecnologia Digital Aplicada à Educação.

Prof. Dr. Marcelo Leandro Eichler
UFRGS/Escola de Engenharia, Instituto de Química.

Prof. Dr. Tales de Vargas Lisbôa
UFRGS/Escola de Engenharia, Departamento de Engenharia Mecânica.

Prof. Dr. Carlos Pérez Bergmann

Coordenador do PPGE3M

Este trabalho é dedicado ao maravilhoso povo do Brasil,
mas principalmente para minha esposa "princesinha"
que representa o maravilhoso povo nordestino.

AGRADECIMENTOS

Os agradecimentos principais são direcionados ao Prof. Dr. Sandro Amico.

E os agradecimentos secundários são para:

- **Minha família da Colômbia/Venezuela**: a qual sempre acreditou na minha capacidade de refazer minha vida durante este duro exílio político.
- **Minha família do Brasil**: a qual sempre levarei no coração, esteja onde estiver.
- **Grupo LAPOL**: onde vi uma exemplar juventude que acredita e luta por um Brasil melhor, mesmo fazendo ciência e engenharia às vezes com mínimos recursos. Especial agradecimento para o Eduardo Menezes por ter me dado um apoio técnico e moral próprio de um amigo.
- **Pós-graduação PPGE3M**: onde vi o processo seletivo, ingresso no mestrado, mais transparente e democrático num curso conceituado como CAPES 7.
- **Alunos de iniciação cientifica do Projeto Mech-Gcomp:** pelos seus comentários e motivação no momento de codificar.
- **Minha esposa:** por estar sempre ao meu lado.
- **Professor Julio:** por ter me ensinado que a matemática é uma ótima ferramenta para o engenheiro.
- **CAPES**: por ter me financiado por meio da Bolsa de mestrado.

"É digno de consideração o homem que procura a verdade com dedicação,
e expõe com sinceridade o resultado das suas investigações.
(Kung Fu Tse)

RESUMO

É uma clara tendência tecnológica propor e testar novos materiais como substitutos daqueles usados tradicionalmente pela humanidade na construção de objetos e estruturas de uso cotidiano, mas as limitações destes materiais constituem um verdadeiro desafio de engenharia. Com o intuito de evoluir e melhorar o ensino dos critérios e envelopes de falha mecânica, foi desenvolvido esta ferramenta útil nas disciplinas de "mecânica de materiais compósitos". As principais ferramentas informáticas usadas foram o framework Django,como ferramenta principal para arquitetar o software, e as linguagens de programação python, html, java para Android, javascript, jquery, e MATLAB com o intuito de acompanhar os lineamentos computacionais deste projeto. Foram implementados os critérios de falha de uso comum no ambiente acadêmico e industrial: máxima tensão, máxima deformação, Tsai-Hill, Azzi-Tsai, Tsai-Wu, Hoffman, Hashin, Cristensen, Puck e Larc03. A implementação foi testada usando números aleatórios e várias proposições matemáticas deduzidas e formuladas durante a revisão bibliográfica. O estudo aprofundado da geométrica analítica, relacionada com estes envelopes de falha, permitiu construir algoritmos cujas saídas superam a qualidade gráfica de alguns softwares tradicionais os quais se limitam a implementar algoritmos tradicionais de computação gráfica gerando envelopes com distorções não desejadas. Foram construídas interfaces gráficas considerando princípios pedagógicos convenientes para o ensino da teoria de mecânica das lâminas onde o contraste entre os diversos critérios e envelopes de falha, mediante a técnica de representação simultânea num único gráfico, facilita as aulas explicativas e comparativas. Este módulo é acessível pela internet mediante controles de acesso personalizados e permite que o usuário reproduza localmente os cálculos de critério de falha e envelopes mediante os códigos MATLAB disponibilizados. A versão Android se vincula com este módulo Django mediante o sistema de espelhado que permite refazer os cálculos no servidor WEB.

Abstract

It is a clear technological trend to propose and test new materials as substitutes of those traditionally used by mankind in the construction of objects and structures but the limitations of these materials are a real challenge of engineering. In order to evolve and improve the teaching of criteria and envelopes of mechanical failure, was developed this useful tool to the disciplines of "mechanics of composite materials". The main computer tools used were the framework Django, as a main tool for architecting software, and programming languages python, html, java for Android, javascript, jquery, and MATLAB with the intention of follow the computational guidelines of this project. Common use criteria, in academic and industrial environments, were implemented: maximum stress, Maximum deformation, Tsai-Hill, Azzi-Tsai, Tsai-Wu, Hoffman, Hashin, Cristensen, Puck and Larc03. The implementation was tested using random numbers and several propositions mathematics deduced and formulated during the bibliographic review. The in-depth study of the analytical geometry, related to these failure envelopes, allowed to construct algorithms whose outputs exceed the graphic quality of some traditional software. which are limited to implementing traditional algorithms of computer graphics generating envelopes with unwanted distortions. Graphical interfaces were constructed considering pedagogical principles suitable for the teaching of blade mechanics theory where the contrast between the various failure criteria and envelopes, through the simultaneous representation in a single graph, facilitates explanatory and comparative classes. This module is accessible through the Internet through personalized access controls and allows that the user reproduce the fault criterion calculations and envelopes locally using the available MATLAB codes. The Android version links to this Django module through the mirror system that allows to redo the calculations in the WEB server.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SÍMBOLOS

AJAX	Asynchronous Javascript and XML
API	Application Programming Interface
APK	Android PacKage
CLT	Teoria clássica de lâminados
CPU	Central Processing Unit
CSS	Cascading Style Sheets
CSV	Comma-separated values
FTP	File Transfer Protocol
HTML	HyperText Markup Language
IDE	Integrated Development Environment
IF	Indíce de Falha
JSON	JavaScript Object Notation
MATLAB	MATrix LABoratory
MIPI	Módulo Inerte de Programação Inicial
PHP	Hypertext Preprocessor
RAM	Random Access Memory
SDK	Software Development Kit
URL	Uniform Resource Locator
E_i	Módulo de Elasticidade na direção i
G_{ij}	Módulo de cisalhamento no plano i-j
ϵ_i	Deformação na direção i
$(\epsilon_i)^X_{ult}$	Deformação na direção normal i, X=C (Compressão), T (Tração)
σ_i	Tensão normal na direção i
$(\sigma_i)^X_{ult}$	Resistência na direção normal i, X=C (Compressão), T (Tração)
τ_{ij}	Tensão de cisalhamento no plano i-j
ν_{ij}	Coeficiente de poisson no plano i-j
γ_{ij}	Deformação angular no plano i-j

Sumário

1 INTRODUÇÃO

No século XXI, muitas aplicações tecnológicas aproveitam as novas propriedades dos materiais compósitos, as quais permitem superar limites de desenho de forma segura e bem calculada, sendo possível assim desenvolver novas tecnologias para os setores industriais, militares, e de saúde. Esta ampla variedade de utilidades dada aos materiais compósitos motiva o estudo constante das suas propriedades físicas e químicas, sendo recente o uso de aplicativos computacionais de simulação para predizer seu comportamento mecânico.

Historicamente, os materiais compósitos estiveram presentes nas construções civis, e os primeiros estudos mecânicos estão associados com a análise de esforços em vigas e pontes (ARFKEN; WEBER, 2005). Estudar as propriedades mecânicas de um material pode-se tornar uma tarefa complexa, especialmente no caso em que diversos esforços e fases estejam presentes, sendo possível simplificar esta análise aplicando condições de simetria, usando a teoria clássica de lâminados (CLT) e critérios de falha mecânica. A importância das teorias e critérios de falha de lâminas e dos materiais compósitos lâminados obriga que as futuras gerações de engenheiros, especialmente das ciências dos matérias e da mecânica, aprendam sobre tais princípios, sendo possível incorporar no processo de aprendizagem ferramentas didáticas modernas como o software aqui detalhado.

Determinar as condições quando um material compósito falha mecanicamente, ou seja, se fratura, é importante para atingir um desenho seguro. Se o material estudado estiver formado por lâminas empilhadas tais condições devem ser primeiramente estudadas para cada lâmina isoladamente para posteriormente tentar predizer as condições de falha do compósito lâminado. Na atualidade, os critérios de falha encontra-se em constante evolução sem que seja possível desprezar as teorias clássicas. Os critérios de Máxima Tensão, Máxima Deformação, Tsai-Hill, Tsai-Wu, Hoffmann, Christensen, Azzi-Tsai, Hashin, Puck e LARC03 foram usados neste trabalho.

Desde 2010 o grupo de pesquisa LAPOL da UFRGS está construindo um software chamado "Mehg-G" que permite levar tais conceitos da mecânica dos materiais compósitos aos alunos da graduação com as inúmeras vantagens que o acesso à internet pode oferecer. O software Mech-Gcomp está sendo desenhado com uma filosofia mais focada no ensino diferenciando-lhe notoriamente dos softwares comerciais similares. Uma vantagem do Mech-Gcomp que lhe coloca acima do software comercial, é o fato de este está em constante crescimento e mesmo assim está totalmente disponível na Internet sem que o usuário seja impedido de acessar o sistema por motivos de atualização.

Neste trabalho foi criado, testado e preparado um módulo inerte de programação inicial (MIPI), nos moldes do framework Django, de critérios e envelopes de falha com

interfaces de usuário desenhadas especialmente para que possam ser aproveitadas pelos alunos das disciplinas de mecânicas de compósitos, e de materiais compósitos da graduação e da pós-graduação em engenharia dos materiais.

O uso do MIPI permitiu apreciar diversos desafios de integração e codificação conforme os moldes Django, não sendo aperfeiçoado este módulo dado que o mesmo não seria destinado para seu uso definitivo nem colocado em produção. Mas a utilidade deste MIPI resulta clara para este trabalho dado que é preciso comparar, de forma empírica, tanto o modelo tradicional de desenvolvimento já usado do projeto inicial MECH-G com o modelo de desenvolvimento da codificação Java/Android feita neste mestrado, comparação que permite direcionar esforços de codificação das funcionalidades educativas.

Com o intuito de manter a filosofia de software livre presente na atual metodologia de desenvolvimento do Mehg-G, foram usadas ferramentas de programação não comerciais o que diminuiu significativamente o custo da etapa de codificação e garantiu a fácil incorporação das muitas ferramentas de cálculo numérico e álgebra computacional difundidas como software acadêmico de uso livre. Também foi criado um aplicativo para o sistema operacional Android (LECHETA, 2015) onde foram codificados em Java os mesmos algoritmos implementados em Python no módulo Django, com o intuito de facilitar um método alternativo que não precisa da Internet para poder funcionar.

O intenso uso dos critérios de falha, mesmo daqueles de simples formulação matemática como os de máximo esforço ou deformação, no desenho industrial com materiais compósitos (LUBIN, 2013) obriga que os mesmos sejam estudados, reformulados e programados com o intuito de obter deles um ótimo aproveitamento acadêmico seja no ensino ou na pesquisa (KATTAN, 2010). Por exemplo, sabe-se que os critérios de falha de máxima tensão, máxima deformação, Tsai-Hill e Tsai-Wu são de uso industrial frequente e por isso constituem a base mínima de qualquer curso de mecânica de materiais compósitos. Os critérios de falha similares, por exemplo Azzi-Tsai, Hoffman e Christessen, constituem refinamentos que devem ser analisados em cursos avançados de pós-graduação e no mínimo programado em ferramentas numéricas consolidadas, como o MATLAB, com o intuito de facilitar o crescimento do pensamento mecânico de um engenheiro de materiais.

É difícil obter no mercado informático uma ferramenta para o estudo dos critérios e envelopes de falha que seja de total acesso pela internet, de fácil uso para usuários inexperientes, e que integre os algoritmos com um banco de dados atualizado das propriedades dos principais materiais. Geralmente, o software comercial possui limitações de licença e não é de uso intuitivo, e possui limitações para ser instalado num sistema operacional em especifico (AUTODESK, 2004-2016).

1.1 Objetivos

Objetivo geral

Desenvolver um sistema informático constituído por dois aplicativos. Um aplicativo escrito em linguagem python, html, javaScript, jquery, e MATLAB conforme os moldes do framework Django e outro aplicativo escrito em Java para o sistema operacional Android. Este sistema servirá para estudar os critérios e envelopes de falha de lâminas no ensino de materiais compósitos.

Objetivos Específicos

- Desenhar os algoritmos de construção dos envelopes de falha, usando geometria analítica e lógica matemática.
- Aplicar vários dos princípios de desenvolvimento, já usados na criação do software Mehg-G, na construção do MIPI proposto com o intuito de considerar e escolher as diversas alternativas de codificação sejam estas do tipo programação WEB ou programação Java/Android.
- Aproveitar o sistema de passe de argumentos, por URL, do framework Django para integrar os aplicativos desenvolvidos.
- Criar interfaces de usuário simples e didáticas para serem usadas em disciplinas de "mecânica de materiais compósitos".
- Descrever o desenho computacional com o intuito de garantir a reprodutibilidade dos algoritmos codificados.

2 FUNDAMENTOS TEÓRICOS

2.1 Análise macromecânica das lâminas

O comportamento mecânico da lâmina depende, principalmente, das propriedades elásticas do material que pode ser quimicamente puro, como no caso dos elementos metálicos, ou compósitos como no caso dos reforços de fibra/matriz usados para diversos fins. Esta primeira diferença físico-química permite simplificar os cálculos ao conhecer o número de constantes elásticas independente que devem ser usadas para um estudo mecânico, no regime elástico, da lâmina. O comportamento elástico das lâminas pode ser formulado usando a "lei de Hooke" na forma matricial relacionando deformações e tensões, que permite aproveitar todas as vantagens matemáticas da álgebra matricial (Eq. 2.1) , especialmente no que diz respeito às propriedades de simetrias.

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_{23} \\ \tau_{31} \\ \tau_{12} \end{bmatrix} = C \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \gamma_{23} \\ \gamma_{31} \\ \gamma_{12} \end{bmatrix} \tag{2.1}$$

Os subíndices 1,2,3 das deformações (ϵ, γ) e das cargas aplicadas (σ, τ) referem-se às direções de aplicação em coordenadas ortogonais locais, ou seja, definidas conforme as orientações da fibra reforçadora sendo esta quem define a direção principal (subíndice 1) no caso de lâminas do tipo fibra/matriz. Na Figura 1 pode-se visualizar os conceitos de esforços normais e cisalhante para uma lâmina, definidos em coordenadas locais.

No caso dos materiais ortotrópicos, a matriz inversa desta matriz de rigidez C, ou seja a matriz de flexibilidade S, permite visualizar facilmente os módulos de elasticidade e de cisalhamento ao tê-los como elementos matriciais:

$$S = \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{12}}{E_1} & -\frac{\nu_{13}}{E_1} & 0 & 0 & 0 \\ -\frac{\nu_{21}}{E_2} & \frac{1}{E_2} & -\frac{\nu_{23}}{E_2} & 0 & 0 & 0 \\ -\frac{\nu_{31}}{E_3} & -\frac{\nu_{32}}{E_3} & \frac{1}{E_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{23}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{31}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{12}} \end{bmatrix} \tag{2.2}$$

Na prática, os esforços de interesse numa lâmina são aqueles da forma planar

pura, ou seja sem componentes normais ao plano da lâmina, o que permite reduzir o número de constantes elásticas independentes e aproveitar as formulações baseadas nas constantes de engenharia $E_x, E_y, G_{XY}, \nu_{XY}$ as quais no caso de uma lâmina não angular (sem rotação angular planar) são escritas como $E_1, E_2, G_{12}, \nu_{12}$. Estas constantes têm uma especial utilidade para caracterizar lâminas pelo fato de estarem associadas com tensões e deformações lineares aplicadas de forma planar de fácil reprodução experimental.

No estado plano de tensões, ou seja, sem tensões normais ao plano que contêm a lâmina, a matriz S converte-se numa matriz reduzida de menor dimensão algébrica, e que pode ser calculada computacionalmente com mais facilidade por ter apenas 9 elementos. Por ser a matriz de rigidez C a inversa de S podemos analisá-la no estado simplificado de tensões planares e criar uma nova matriz chamada de matriz de rigidez reduzida Q com a notação:

$$Q = S^{-1} = \begin{bmatrix} Q_{11} & Q_{12} & 0 \\ Q_{21} & Q_{22} & 0 \\ 0 & 0 & Q_{66} \end{bmatrix} \tag{2.3}$$

onde:

$$Q_{11} = \frac{E_1}{1 - \nu_{12} \cdot \nu_{21}}; Q_{12} = \nu_{12} \cdot \frac{E_2}{1 - \nu_{21} \cdot \nu_{12}}; Q_{22} = \frac{E_2}{1 - \nu_{21} \cdot \nu_{12}}; Q_{66} = G_{12}; \tag{2.4}$$

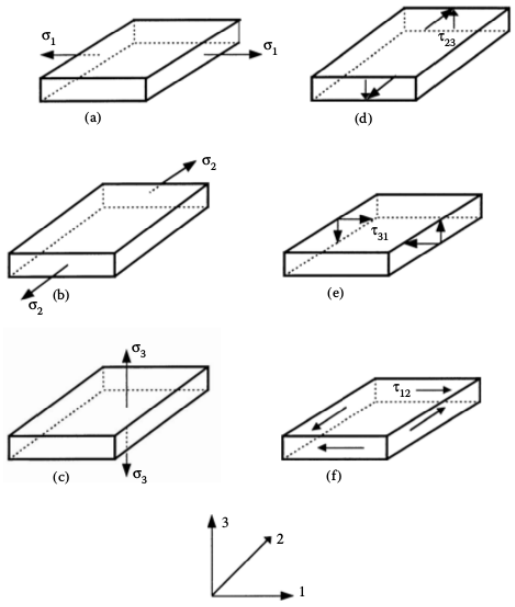

Figura 1: Esforços aplicados numa lâmina para encontrar as constantes de engenharia. (KATTAN, 2010)

2.2 Lâminas angulares

Quando uma lâmina for rotacionada um ângulo θ, com respeito ao eixo X positivo no plano X-Y, a formulação da lei de Hooke matricial (Eq. 2.1) pode se manter, sendo apenas preciso incorporar uma matriz de rotação $T(\theta)$ (Eq. 2.5) para levar em conta a distribuição dos esforços.

$$T = \begin{bmatrix} cos^2(\theta) & sin^2(\theta) & 2 \cdot cos(\theta) \cdot sin(\theta) \\ sin^2(\theta) & cos^2(\theta) & -2 \cdot cos(\theta) \cdot sin(\theta) \\ -cos(\theta) \cdot sin(\theta) & cos(\theta) \cdot sin(\theta) & cos^2(\theta) - sin^2(\theta) \end{bmatrix} \tag{2.5}$$

Assim deve se diferenciar entre as coordenadas globais, indicadas com as letras do subíndice X,Y,Z e as coordenadas locais, indicadas com os subíndices 1,2,3. Sendo possível relacionar as deformações e tensões aplicados usando a (Eq. 2.7) onde a matriz de Reuter (R) está definida como:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \tag{2.6}$$

Para finalmente calcular:

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} = T^{-1} \cdot Q \cdot R \cdot T \cdot R^{-1} \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix} \tag{2.7}$$

Esta matriz de rotação não está relacionada com a matriz de rotação de Euler e portanto não representa uma matriz de rotação dos eixos coordenados, mas é uma matriz unitária (ARFKEN; WEBER, 2005) o que garante a transformação preservada de um espaço de esforços para outro.

2.3 Critérios envelopes de Falha

Os critérios de falha são aquelas funções matemáticas que relacionam o índice de falha (IF), ou seja, o grau de aproximação à condição de fratura com os esforços aplicados. No caso de uma lâmina, esta função têm como variáveis os esforços nas direções principais em coordenadas locais, assim como as propriedades mecânicas do material. Os envelopes de falha são a representação gráfica dos critérios de falha que permitem determinar, no plano de esforços, a combinação de esforços que podem ser suportados por uma lâmina de forma segura.

Pode-se considerar que os critérios de falha mais simples são aqueles cujo modelo matemático considera apenas normalizações e uniões lógicas dos esforços aplicados, como no caso dos "critérios de máxima tensão" e "critérios de máxima deformação", sendo estes os primeiros critérios aqui estudados. E como critérios de maior complexidade, tanto computacional como matemática aqueles que tentam modelar vários fenômenos físicos de fratura como no caso dos critérios de Larc03 e Puck os quais são estudados na etapa final deste mestrado.

2.3.1 Máxima Tensão

Relaciona diretamente os dados experimentais com os esforços aplicados na lâmina, em coordenadas locais, mediante as desigualdades:

$$-(\sigma_1)^c_{ult} \leq \sigma_1 \leq (\sigma_1)^T_{ult} \tag{2.8}$$
$$-(\sigma_2)^c_{ult} \leq \sigma_2 \leq (\sigma_2)^T_{ult} \tag{2.9}$$
$$-(\tau_{12})_{ult} \leq \tau_{12} \leq (\tau_{12})_{ult} \tag{2.10}$$

Do ponto de vista numérico, este envelope de falha relaciona o índice de falha (IF) com os valores extremos, por meio da definição de vários tipos de IF:

$$\begin{aligned} IF\ tração\ direção\ 1 &= \frac{|\ (\sigma_1)\ |}{(\sigma_1)^T_{ult}} \\ IF\ tração\ direção\ 2 &= \frac{|\ (\sigma_2)\ |}{(\sigma_2)^T_{ult}} \\ IF\ compressão\ direção\ 1 &= \frac{|\ (\sigma_1)\ |}{(\sigma_1)^C_{ult}} \\ IF\ compressão\ direção\ 2 &= \frac{|\ (\sigma_2)\ |}{(\sigma_2)^C_{ult}} \\ IF\ cisalhamento &= \frac{|\ (\tau_{12})\ |}{(\tau_{12})_{ult}} \end{aligned} \tag{2.11}$$

No caso de lâminas angulares, estas desigualdades devem ser rotacionadas para construir um sistema de desigualdades um pouco mais complexo da forma:

$$\begin{bmatrix} -(\sigma_1)^c_{ult} \\ -(\sigma_2)^c_{ult} \\ -(\tau_{12})_{ult} \end{bmatrix} < \begin{bmatrix} cos^2(\theta) & sin^2(\theta) & 2 \cdot cos(\theta) \cdot sin(\theta) \\ sin^2(\theta) & cos^2(\theta) & -2 \cdot cos(\theta) \cdot sin(\theta) \\ -cos(\theta) \cdot sin(\theta) & cos(\theta) \cdot sin(\theta) & cos^2(\theta) - sin^2(\theta) \end{bmatrix} \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} \tag{2.12}$$

Em união lógica com o sistema:

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} \begin{bmatrix} cos^2(\theta) & sin^2(\theta) & 2 \cdot cos(\theta) \cdot sin(\theta) \\ sin^2(\theta) & cos^2(\theta) & -2 \cdot cos(\theta) \cdot sin(\theta) \\ -cos(\theta) \cdot sin(\theta) & cos(\theta) \cdot sin(\theta) & cos^2(\theta) - sin^2(\theta) \end{bmatrix} < \begin{bmatrix} (\sigma_1)^T_{ult} \\ (\sigma_2)^T_{ult} \\ (\tau_{12})_{ult} \end{bmatrix} \tag{2.13}$$

Por ser um sistema com muitas desigualdades, as considerações geométricas para construir o envelope de falha não são poucas. De fato autores como Kaw (KAW, 2005) propõem o método de tabelamento para obter os envelopes de falha deste sistema de 6 desigualdades, sendo possível construir rapidamente estas tabelas com ajuda do atual poder computacional, mas o método analítico garante um melhor entendimento da forma final do envelope deste sistema rotacionado. Autores renomeados consultados (KAW, 2005; BARBERO, 2010; CHRISTENSEN, 2012; KATTAN, 2010; LUBIN, 2013) não indicam um método geométrico para construir este tipo de envelope.

A formulação geométrica deste envelope de falha começa com o conceito de semiplano de esforços gerado por cada desigualdade, e com o conceito de faixa definida com a união lógica de duas desigualdades, sendo mais prático trabalhar com faixas ao invés de semiplanos por ser mais visível do ponto de vista geométrico. 17

Um caso simples, destas faixas (Fig. 2a) onde vê-se a faixa vertical definida pela desigualdade: $-(\sigma_1)^C_{ult} < \sigma_x < (\sigma_1)^T_{ult}$, e ve-se o quadrado que representa o envelope de falha de máxima tensão de um material com as propriedades: $(\sigma_1)^T_{ult} = 1500(Pa)$, $(\sigma_1)^C_{ult} = 1500(Pa)$, $(\sigma_2)^T_{ult} = 40(Pa)$ e $(\sigma_2)^C_{ult} = 246(Pa)$. Similarmente a desigualdade $-(\sigma_2)^C_{ult} < \sigma_y < (\sigma_2)^T_{ult}$ define uma faixa horizontal representada (Fig. 2b) neste caso com as propriedades de: $(\sigma_2)^T_{ult} = 1500(Pa)$, $(\sigma_2)^C_{ult} = 1500(Pa)$, $(\sigma_1)^T_{ult} = 40(Pa)$ e $(\sigma_1)^C_{ult} = 246(Pa)$.

Todas as 6 desigualdades (Eq. 2.12 e Eq. 2.13) ao ser convertidas em igualdades, representam um conjunto de linhas retas que limitam a região do plano de esforços onde o índice de falha é menor que 1, sendo possível definir os pontos de intercessão destas retas mediante a eliminação gaussiana. Uma forma de simplificar esta eliminação gaussiana e de visualizar com mais facilidade o problema analítico, consiste em definir faixas de área finita mediante a obtenção dos pontos de intercessão definidas na união lógica das desigualdades que contém o σ_x (ou σ_y) com aquelas que contém o esforço não contido no plano, ou seja τ_{xy}, com o intuito de obter duas faixas que facilmente definem (Fig. 3) uma única área comum. A vantagem de usar estas faixas inclinadas (Fig. 3) é que permite obter uma linha poligonal fechada de área mínima que representa o envelope de falha de máxima tensão e que contém apenas aqueles pontos que satisfazem 6 desigualdades.

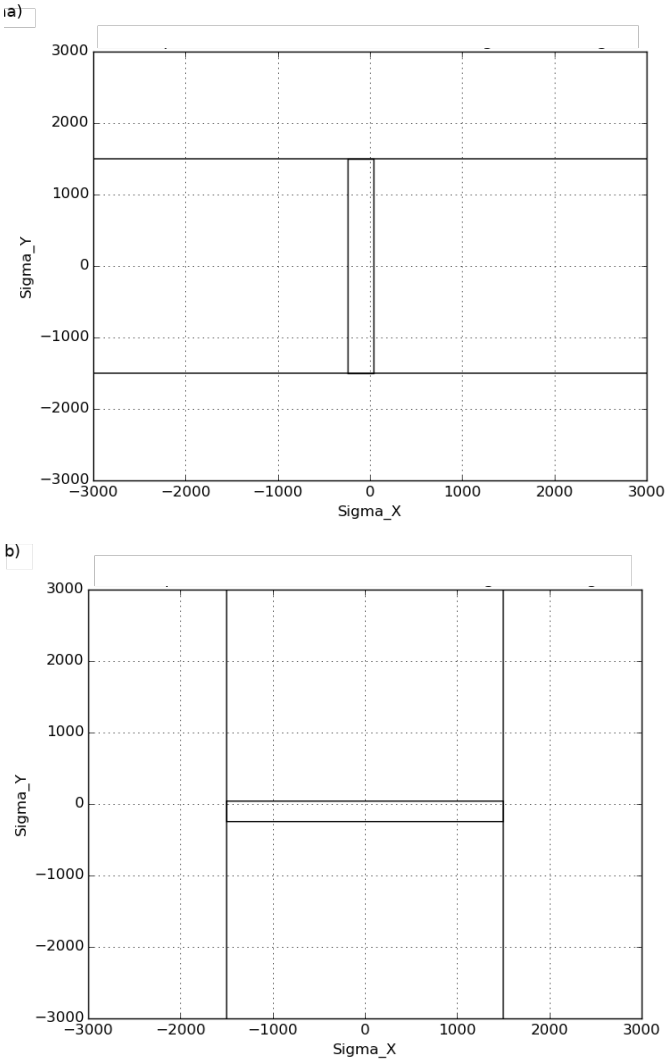

Figura 2: (a) Faixa vertical associada com um envelope de máxima tensão. (b) Faixa horizontal associada com um envelope de máxima tensão.Unidades em Pascal (Pa). (Fonte: Autor)

2.3.2 Máxima Deformação

De maneira similar ao critério anterior, relacionando os dados experimentais das máximas deformações toleradas pela lâmina com as deformações por ela sofridas no regime elástico, são definidos os intervalos de deformação segura onde, por este critério, não existe risco de fratura:

$$-(\epsilon_1)^c_{ult} \leq \epsilon_1 \leq (\epsilon_1)^T_{ult} \tag{2.14}$$

$$-(\epsilon_2)^c_{ult} \leq \epsilon_2 \leq (\epsilon_2)^T_{ult} \tag{2.15}$$

$$-(\gamma_{12})^c_{ult} \leq \gamma_{12} \leq (\gamma_{12})^T_{ult} \tag{2.16}$$

A formulação deste critério permite usar o mesmo tratamento matemático que o

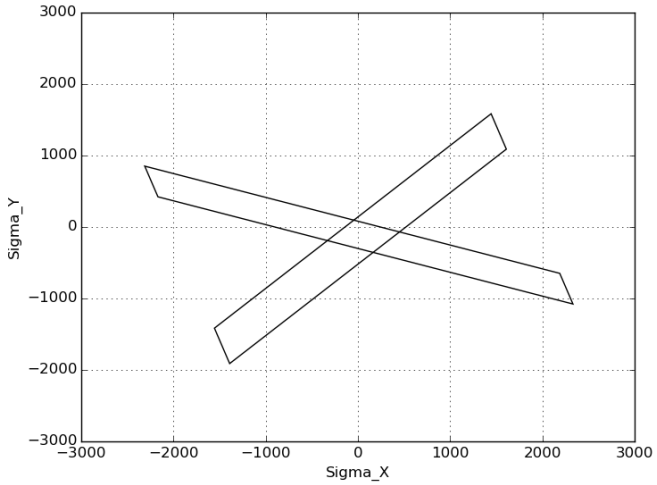

Figura 3: Faixas inclinadas associadas com um envelope de máxima tensão. Unidades em Pascal (Pa). (Fonte: Autor)

critério de máxima tensão e definir os IF de forma similar. Do ponto de vista numérico, este envelope de falha relaciona o índice de falha (IF) com os valores extremos, por meio da definição de vários tipos de IF:

$$IF\ tração\ direção\ 1 = \frac{|\ (\epsilon_1)\ |}{(\epsilon_1)^T_{ult}} \tag{2.17}$$

$$IF\ tração\ direção\ 2 = \frac{|\ (\epsilon_2)\ |}{(\epsilon_2)^T_{ult}} \tag{2.18}$$

$$IF\ compressão\ direção\ 1 = \frac{|\ (\epsilon_1)\ |}{(\epsilon_1)^C_{ult}} \tag{2.19}$$

$$IF\ compressão\ direção\ 2 = \frac{|\ (\epsilon_2)\ |}{(\epsilon_2)^C_{ult}} \tag{2.20}$$

$$IF\ cisalhamento = \frac{|\ (\gamma_{12})\ |}{(\gamma_{12})_{ult}} \tag{2.21}$$

Pode-se prever que o mesmo algoritmo usado para obter o envelope de falha de máxima tensão pode ser adaptado, com modificações mínimas, para este caso, rotacionando as deformações globais para ser transformadas em deformações locais:

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \gamma_{12} \end{bmatrix} = (R \cdot T \cdot R^{-1}) \cdot \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix} \tag{2.22}$$

e construindo o sistema de desigualdades, tendo como variáveis as deformações locais ao invés dos esforços, sendo preciso apenas transformar os pontos obtidos como vértices deste envelope em representações no plano de esforço usando a Lei de Hooke matricial. O novo

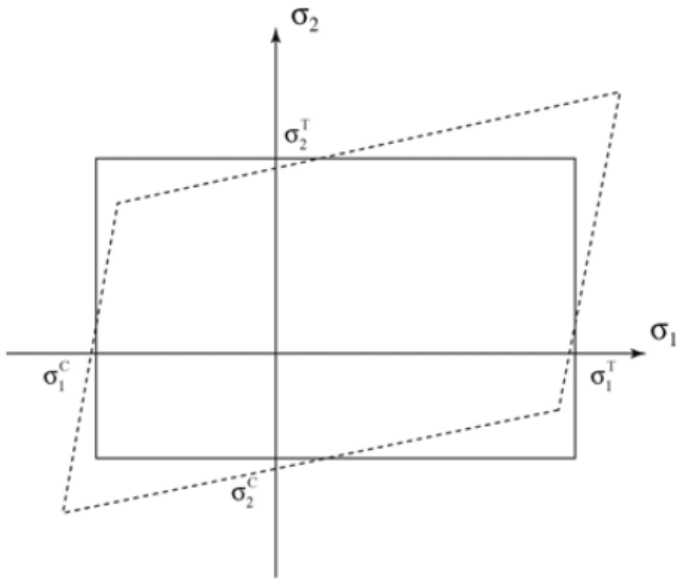

Figura 4: Envelope de falha de máxima deformação (linha tracejada) e de máxima tensão (linha continua) de uma lâmina não rotacionada (KATTAN, 2010).

algoritmo, para este envelope de falha, deve calcular a matriz $\overline{Q}$ conforme a rotação e as propriedades de cada lâmina.

A análise geométrica deste critério de falha é idêntico ao critério de máxima tensão, já que em ambos aparecem semi-planos definidos pelas desigualdades do critério (Eq. 2.16), aparecem faixas e o envelope se define geometricamente como a área comum da interseção lógica destas faixas. Esta interseção lógica poderia ser feita de forma numérica, comparando-se ponto por ponto contido na faixa, ou de forma analítica obtendo, apenas aqueles vértices que a limitam simplificando a tarefa de análise de muitos pontos para simplesmente obter os vértices que limitam a área comum procurada.

Uma diferença, que não afeta o algoritmo de construção deste envelope, é que no plano de esforços (σ_x, σ_y), em cada envelope de falha deste critério deve se apreciar o módulo de Poisson, ν_{12}. No caso de uma lâmina não rotacionada, o coeficiente angular das arestas menos verticais corresponde ao coeficiente de Poisson ν_{12} e as mais inclinadas ao seu inverso $1/\nu_{12}$. Esta inclinação não aparece no envelope de máxima tensão (Fig. 4) visto que as variáveis das desigualdades de esforços máximos não estão acopladas, o que acontece neste critério de deformação, onde verifica-se o acoplamento das deformações por meio da relação: $\nu_{12} = -\frac{\epsilon_2}{\epsilon_1}$. Se o coeficiente de Poisson for zero, este critério de máxima deformação gera envelopes de falha com a mesma forma que os envelopes de máxima tensão. Outra interpretação do $\nu_{12} = 0$ é o fato que desacopla, algebricamente, as desigualdades que definem este critério de falha.

2.4 Critérios e envelopes quadráticos

As desigualdades que modelam os critérios de máxima tensão e deformação são de grande utilidade no estudo de materiais frágeis (cerâmicas, fibras não orgânicas, etc.), mas não acompanham o comportamento tolerante dos materiais dúcteis (maioria dos metais) onde é preciso superar o comportamento elástico antes de provocar a fratura do material. Esta transição entre comportamento elástico e não elástico obriga que seja levada em conta a atuação conjunta das forças que agem sobre o material para definir um critério de falha. O critério de Von-Mises (MISES, 1913) representa uma primeira tentativa de levar em conta a atuação conjunta, por pares, dos esforços que agem sobre um material onde ve-se este agrupamento de esforços nas bases das potências as quais são somadas para definir o índice de falha. Esta formulação polinomial dos esforços combinados em pares permite definir a iteração entre eles, tendo como caraterísticas matemáticas de utilidade:

(a) **Termos cruzados**: Permite a rápida detecção da geometria do envelope de falha mediante a observação de termos cruzados onde um tipo de esforço multiplica outro.

(b) **Tipos de esforços com a mesma potência**: Permite simplificar o estudo geométrico dos envelopes de falha aplicando transformações entre um sistema de coordenadas cartesianas e outro com alguma simetria mais conveniente, como o sistema de coordenadas polares muito usado no estudo das curvas cônicas.

2.4.1 Tsai-Hill

Este critério de falha derivado da antiga teoria (SCHWARTZ et al., 1968), permite levar em conta as interações entre os componentes normais de tensão, aplicados em diferentes direções ortogonais, usando a formulação polinomial:

$$A \cdot (\sigma_1^2) + B \cdot (\sigma_2^2) + C \cdot (\sigma_1 \cdot \sigma_2) + D \cdot (\tau_{12}^2) = IF \tag{2.23}$$

Onde os coeficientes A,B são função de algum limite de elasticidade do material σ_{ult}. Nesta formulação polinomial (Eq. 2.23), pode-se definir o τ_{12} como parâmetro constante e assim obter uma curva cônica planar onde o termo cruzado não nulo $C * (\sigma_1 * \sigma_2)$ define a rotação (maior de 0° e menor de 90° graus) de uma elipse sempre centrada na origem do sistema de coordenadas locais. Quando o IF (Eq. 2.23) for menor ou igual a 1 estamos dentro do regime elástico e quando for maior que 1 estamos fora do regime elástico, o que pode gerar escoamento ou fratura do material. No critério de Tsai-Hill, os coeficientes podem ser definidos como apresentados na Tabela 1.

Tabela 1: Coeficientes da Eq. 2.23 polinomial associada ao critério de Tsai-Hill.

Coeficiente Tsai-Hill	Condição mecânica
$A = (\frac{1}{((\sigma_1)^T_{ult}})^2$	$((\sigma_1)^T_{ult})^2$ se obtém com $\sigma_2 = \tau_{12} = 0$
$B = (\frac{1}{((\sigma_2)^T_{ult}})^2$	$((\sigma_2)^T_{ult})^2$ se obtém com $\sigma_1 = \tau_{12} = 0$
$D = (\frac{1}{((\tau_{12})_{ult}})^2$	$((\tau_{12})_{ult})^2$ se obtém com $\sigma_1 = \sigma_2 = 0$

O valor do coeficiente C, neste critério está definido como: $C = -(\frac{1}{((\sigma_1)_{ult}})^2$. O envelope de falha deste critério, representado no plano de esforços locais σ_1 e σ_2, é de fácil construção pois o polinômio (Eq. 2.23) pode se escrever conforme a fórmula geral de uma elipse rotacionada: $a \cdot (\sigma_1^2) + 2 \cdot b \cdot (\sigma_1 \cdot \sigma_2) + c \cdot (\sigma_2^2) + g = 0$ onde o ângulo de inclinação da elipse é:

$$\theta_{elipse} = \frac{\pi}{2} + \frac{cot^-1(\frac{a-c}{2 \cdot b})}{2} \tag{2.24}$$

O termo cruzado $2 \cdot b \cdot (\sigma_1 \cdot \sigma_2)$ garante que o fator b, na Eq. 2.24, seja não nulo e portanto a elipse que representa este envelope de falha estará sempre inclinada quando os coeficientes a e c forem diferentes. São usados apenas os valores extremos de tração $(\sigma_1)^T_{ult}$ e $(\sigma_2)^T_{ult}$, o que limita a capacidade de predição deste critério, mas no texto original foi formulado considerando a simetria entre os valores extremos de tração e compressão, ou seja, $(\sigma_1)^T_{ult} = (\sigma_1)^C_{ult}$ e $(\sigma_2)^T_{ult} = (\sigma_2)^C_{ult}$

2.4.2 Azzi-Tsai

Com o intuito de tonar o Tsai-Hill um critério mais realista foi proposto o critério de Azzi-Tsai (AZZI; TSAI, 1965) considerando a resistência à compressão. A incorporação desta resistência na formulação deste critério obriga que seja feito um estudo matemático diferente. Os coeficientes da curva cônica (Eq. 2.23), que este critério define, dependem agora do esforço local aplicado tal como se explica na Tabela 2 onde pode-se ver que o coeficiente muda conforme o sinal de cada esforço aplicado.

Tabela 2: Coeficientes do critério de Azzi-Tsai.

Coeficiente Azzi-Tsai	Condição mecânica
$A = (\frac{1}{((\sigma_1)^T_{ult}})^2$	Se $\sigma_1 > 0$
$A = (\frac{1}{((\sigma_1)^C_{ult}})^2$	Se $\sigma_1 < 0$
$C = (\frac{1}{((\sigma_2)^T_{ult}})^2$	Se $\sigma_2 > 0$
$C = (\frac{1}{((\sigma_2)^C_{ult}})^2$	Se $\sigma_2 < 0$
$B = (\frac{1}{((\sigma_1)^T_{ult}})^2$	Se $\sigma_2 > 0$
$B = (\frac{1}{((\sigma_1)^C_{ult}})^2$	Se $\sigma_2 < 0$

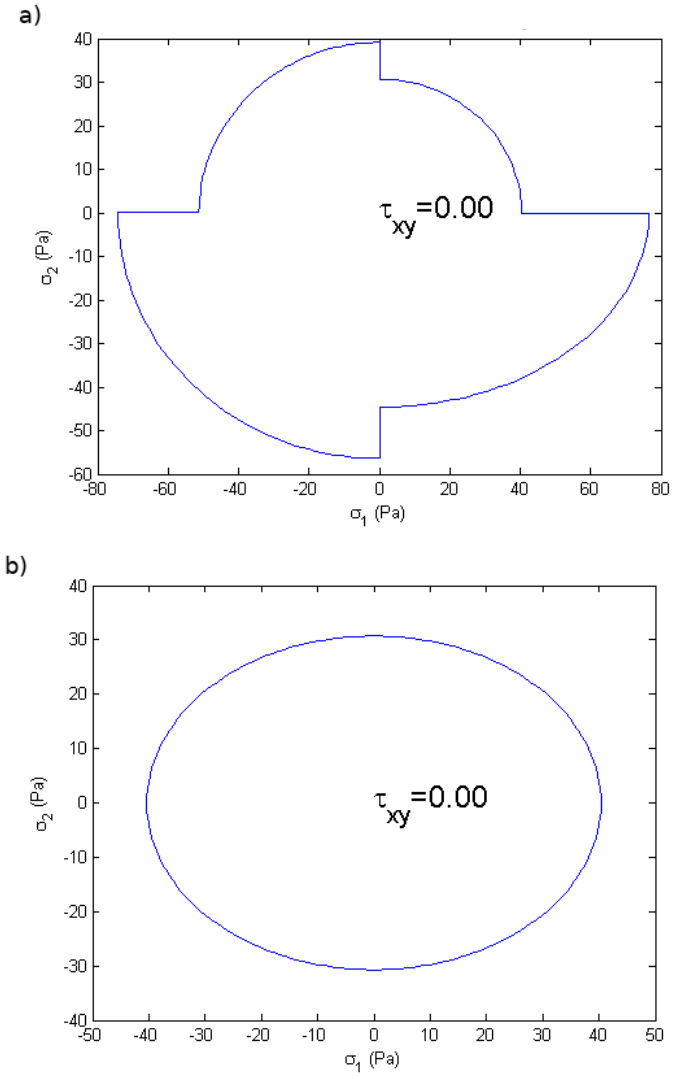

Figura 5: (a) Envelope de falha de Azzi-Tsai numa lâmina de resistência $(\sigma_1)^T_{ult} = 10Pa$, $(\sigma_1)^C_{ult} = -15Pa$,$(\sigma_2)^T_{ult} = 10Pa$,$(\sigma_2)^C_{ult} = -15Pa$. (b) Envelope de falha de Azzi-Tsai numa lâmina de resistência $(\sigma_1)^T_{ult} = 10Pa$, $(\sigma_1)^C_{ult} = -10Pa$,$(\sigma_2)^T_{ult} = 10Pa$,$(\sigma_2)^C_{ult} = -10Pa$. (Fonte: Autor)

O envelope de falha deste critério é a união lógica de várias curvas cônicas que podem ser ou não elipses que geralmente constituem uma construção final pouco simétrica e com significativas descontinuidades. Para ilustrar estas descontinuidades (Fig. 5a) pode-se considerar uma lâmina com valores de resistências similares, próxima a condição de isotropia, para assim garantir na representação do plano de esforços locais a existência de elipses. . Sendo possível construir envelopes Azzi-tsai menos descontínuos à medida que o material se comporte de forma isotrópica ou pelo menos transversalmente isotrópica no plano local 1-2.

Os valores extremos deste envelope de Falha, podem ser definidas como aqueles valores mas à direita ou à esquerda, ou mais na parte acima ou abaixo da figura do envelope. Já o corte com os eixos resulta ambíguo em algumas condições (Fig. 5)

2.4.3 Tsai-Wu

Outra evolução do critério de Tsai-Hill foi o chamado critério de Tsai-Wu, que incorporou um parâmetro experimental biaxial σ_b. A formulação deste critério (TSAI; WU, 1971) preserva a forma inicial do critério original adicionando termos em potência unitária de σ_1 e σ_2. Os coeficientes desta curva cônica $(a \cdot (\sigma_1^2) + 2 \cdot b \cdot (\sigma_1 \cdot \sigma_2) + c \cdot (\sigma_2^2) + 2 \cdot d \cdot (\sigma_1) + 2 \cdot f \cdot (\sigma_1) + g = 0\ )$ pode-se ver na formulação deste critério:

$$\begin{aligned} &(\frac{1}{((\sigma_1)^T_{ult}) \cdot ((\sigma_1)^C_{ult})}) \cdot (\sigma_1^2) + (\frac{1}{((\sigma_2)^T_{ult}) \cdot ((\sigma_2)^C_{ult})}) \cdot (\sigma_2^2) + \\ &(\frac{1}{(\sigma_1)^T_{ult}} - \frac{1}{(\sigma_1)^C_{ult}}) \cdot (\sigma_1) + (\frac{1}{(\sigma_2)^T_{ult}} - \frac{1}{(\sigma_2)^C_{ult}}) \cdot (\sigma_2) + (\frac{\tau_{12}}{((\tau_{12})_{ult})})^2 + \\ &2 \cdot \sigma_1 \cdot \sigma_2 (\frac{1}{(\sigma_b)^2}) \cdot [1 - \sigma_b \cdot (\frac{1}{(\sigma_1)^T_{ult}} - \frac{1}{(\sigma_1)^C_{ult}} + \frac{1}{(\sigma_2)^T_{ult}} \\ &- \frac{1}{(\sigma_2)^C_{ult}}) - (\sigma_b)^2 \cdot (\frac{1}{(\sigma_1)^T_{ult} \cdot (\sigma_1)^C_{ult}} + \frac{1}{(\sigma_2)^T_{ult} \cdot (\sigma_2)^T_{ult}})] = IF \end{aligned} \tag{2.25}$$

Desta formulação pode-se obter os coeficientes usados no estudo geometrico (ARFKEN; WEBER, 2005), no plano (σ_1,σ_2):

$$\begin{aligned} b = &(\frac{1}{(\sigma_b)^2}) \cdot [1 - \sigma_b \cdot (\frac{1}{(\sigma_1)^T_{ult}} - \frac{1}{(\sigma_1)^C_{ult}} + \frac{1}{(\sigma_2)^T_{ult}} \\ &- \frac{1}{(\sigma_2)^C_{ult}}) - (\sigma_b)^2 \cdot (\frac{1}{(\sigma_1)^T_{ult} \cdot (\sigma_1)^C_{ult}} + \frac{1}{(\sigma_2)^T_{ult} \cdot (\sigma_2)^T_{ult}})] \end{aligned} \tag{2.26}$$

$$c = (\frac{1}{((\sigma_2)^T_{ult}) \cdot ((\sigma_2)^C_{ult})}) \tag{2.27}$$

$$2 \cdot d = (\frac{1}{(\sigma_1)^T_{ult}} - \frac{1}{(\sigma_1)^C_{ult}}) \tag{2.28}$$

$$\tag{2.29}$$

Os critérios de Tsai-Hill, Azzi-Tsai e Tsai-Wu não consideram os fenômenos físicos associados ao processo de fratura do material, o que faz com que estes critérios sejam alvo de intensas criticas por parte da comunidade de engenheiros mecânicos. Aparentemente, este supera os critérios de máxima tensão e máxima deformação ao considerar o acoplamento de um esforço com outro nos termos cruzados, mas o fato de ter uma representação única do IF impossibilita a apreciação de forma direta do tipo de esforço que contribui de forma significativa para à falha.

2.4.4 Hoffman

Pode-se considerar como uma simplificação (HOFFMAN, 1967) do critério de Tsai-Wu, sem o parâmetro experimental biaxial. Este é formulado como:

$$\begin{aligned}(\frac{1}{((\sigma_1)^T_{ult})\cdot((\sigma_1)^C_{ult})})\cdot(\sigma_1^2)+(\frac{1}{((\sigma_2)^T_{ult})\cdot((\sigma_2)^C_{ult})})\cdot(\sigma_2^2)+\\ (\frac{1}{(\sigma_1)^T_{ult}}-\frac{1}{(\sigma_1)^C_{ult}})\cdot(\sigma_1)+(\frac{1}{(\sigma_2)^T_{ult}}-\frac{1}{(\sigma_2)^C_{ult}})\cdot(\sigma_2)+(\frac{\tau_{12}}{((\tau_{12})_{ult})})^2-\\ \sigma_1\cdot\sigma_2\cdot(\frac{1}{((\sigma_1)^T_{ult})\cdot((\sigma_1)^C_{ult})})=IF\end{aligned} \tag{2.30}$$

Neste IF encontra-se garantida a curva do tipo elipse quando o descriminante for positivo:

$$\begin{aligned}a\cdot c-b^2>0\Rightarrow(\frac{1}{((\sigma_1)^T_{ult})\cdot((\sigma_1)^C_{ult})})\cdot(\frac{1}{((\sigma_2)^T_{ult})\cdot((\sigma_2)^C_{ult})})>\\ (\frac{1}{((\sigma_1)^T_{ult})\cdot((\sigma_1)^C_{ult})})^2\Rightarrow\frac{(\sigma_1)^T_{ult}\cdot(\sigma_1)^C_{ult}}{((\sigma_2)^T_{ult})\cdot((\sigma_2)^C_{ult})}>0\end{aligned} \tag{2.31}$$

o que indica que existem materiais cujas resistências podem gerar um descriminante negativo e como consequência "destruir" o envelope de falha. Quando for preciso representar os critérios de Tsai no plano global (σ_x, σ_y) é preciso explicitar as coordenadas locais em termos das globais na formulação do IF no critério correspondente.

2.4.5 Hashin

Com o intuito de considerar os mecanismos de falha, tanto na matriz como na fibra de forma separada distinguindo os efeitos da compressão e da tração, foi proposto por Hashin (HASHIN, 1980) um critério que se formula conforme os fenômenos de falha.

(a) Modo de falha na fibra:

Se $\sigma_1 > 0$, o índice de falha se define como:

$$IF=(\frac{\sigma_1}{(\sigma_1)^T_{ult}})^2+(\frac{\tau_{12}}{(\tau_{12})_{ult}})^2 \tag{2.32}$$

Se $\sigma_1 \leq 0$, o índice de falha se define como:

$$IF=(\frac{\mid\sigma_1\mid}{(\sigma_1)^C_{ult}}) \tag{2.33}$$

(b) Modo de falha na matriz: se $\sigma_2 \geq 0$

$$IF = (\frac{\sigma_2}{(\sigma_2)^T_{ult}})^2 + (\frac{\tau_{12}}{(\tau_{12})_{ult}})^2 \tag{2.34}$$

Se $\sigma_2 \leq 0$ o índice de falha se define como:

$$IF = (\frac{\sigma_1}{2 \cdot (\tau_{23})_{ult}})^2 + [(\frac{(\sigma_2)^C_{ult}}{2 \cdot (\tau_{23})_{ult}})^2 - 1] \cdot (\frac{\sigma_2}{(\sigma_2)^C_{ult}})^2 + (\frac{\tau_{12}}{(\tau_{12})_{ult}})^2 \tag{2.35}$$

O envelope de falha do critério de Hashim é o resultado da união das curvas que se obtém com as diversas formulações do IF (Eq. 2.32, 2.4.5, 2.34 2.35).

2.4.6 Christensen

Outro critério de falha que considera o fenômeno de fratura é o critério de Christessen (CHRISTENSEN, 2012), formulado conforme os modos de falha, numa condição de esforços no plano:

(a) Modo de falha na fibra, onde o índice de falha se define como:

$$IF = (\frac{1}{(\sigma_1)^T_{ult}} - \frac{1}{(\sigma_1)^C_{ult}}) \cdot (\sigma_1) + (\frac{1}{((\sigma_1)^T_{ult}) \cdot ((\sigma_2)^C_{ult})}) \cdot (\sigma_2^2) \tag{2.36}$$

(b) Modo de falha na fibra, para $\sigma_2 \leq 0$, onde o índice de falha se define como:

$$IF = (\frac{1}{(\sigma_2)^T_{ult}} - \frac{1}{(\sigma_2)^C_{ult}}) \cdot (\sigma_2) + (\frac{1}{((\sigma_2)^T_{ult}) \cdot ((\sigma_2)^C_{ult})}) \cdot (\sigma_2^2) + (\frac{\tau_{12}}{(\tau_{12})_{ult}})^2 \tag{2.37}$$

2.5 Critérios baseados no fenômeno de falha e parâmetros experimentais das fases

Os modernos critérios de Puck e LarC03 foram desenvolvidos especialmente para os matérias compósitos lâminados levando em conta vários parâmetros experimentais. As considerações especificas sobre cada modo de falha permite iniciar uma discussão sobre a formulação matemática, como alguns autores ((PANOSSO, 2012) ,(GOUVEA et al., 2006)) que consideram o critério de Larc03 e o de Puck como uma coleção de critérios individuais. Porém, considera-se aqui que a formulação matemática não homogênea destes é apenas uma forma de escrita que se deve interpretar como uma família de funções que surge da necessidades de explicar um único fenômeno físico chamado de falha mecânica.

2.5.1 Puck

Os critérios de falha de máxima tensão, máxima deformação, e até mesmo os critérios quadráticos de Tsai-Hill e Tsai-Wu não consideram nem o mecanismo de falha nem o comportamento físico da fratura, o que lhes limita como ferramenta eficiente para o estudo das propriedades mecânicas dos materiais compósitos. O critério de Puck, formulado por Puck e Schurmnn (PUCK; SCHÜRMANN, 1998) considera a falha como um fenômeno físico modelado matematicamente que permite classifica-lo como mais completo (MANTIČ, 2013) que os critérios já descritos neste capítulo. Este critério possui como limitante o fato que pode ser aplicado apenas para o estudo de compósitos com reforço de fibra unidirecional, e com fratura do tipo frágil. Neste critério de falha se define o índice de falha em duas partes, uma para as fraturas da fibra e outra para a falha da matriz.

Neste critério, a falha na matriz, estudada do ponto de vista do critério de Mohr-Coulomb, permite formular três modos de falha dentro da matriz:

(a) Modo A: Quando se têm um esforço $\sigma_n \geq 0$, no plano de falha, pode-se usar a relação de esforços normalizados como uma boa aproximação mecânica:

$$(\frac{\sigma_n}{R_{\perp}^{(+)A}})^2 + (\frac{\tau_{nt}}{R_{\perp\perp}^{A}})^2 + (\frac{\tau_{n1}}{R_{\perp\parallel}^{A}})^2 = 1 \tag{2.38}$$

Um cuidado que se deve ter no estudo deste plano de falha se refere aos diversos valores de resistências na fratura no plano de ação (R^A), os quais devem ser considerados para um único plano por aplicação da hipótese já mencionada. Uma consideração matemática útil, que permite contornar a dificuldade experimental de obter $R_{\perp\perp}^{A}$, consiste em aproximar este valor com o limite de compressão Y^c. Quando o esforço τ_{n1} atua sozinho ele gera uma falha no seu próprio plano de ação, o que permite aplicar a substituição $R_{\perp\parallel}^{A} = S^L$ e o limite de resistência a fratura $R_{\perp}^{(+)A}$ iguala-se ao limite de resistência a tração Y^T.

Um esforço $\sigma_n \leq 0$ impede a falha por cisalhamento, o que mecanicamente pode-se entender como uma tensão que as forças de cisalhamento (τ_{n1},τ_{nt}) devem superar antes de gerar uma falha, que se descreve de forma similar ao critério de Mohr-Coulomb:

$$(\frac{\tau_{nt}}{R_{\parallel\parallel}^{A} - (p_{\parallel\parallel}^{(-)}) \cdot (\sigma_n)})^2 + (\frac{\tau_{n1}}{R_{\perp\parallel}^{A} - (p_{\perp\parallel}^{(-)}) \cdot (\sigma_n)})^2 = 1 \tag{2.39}$$

Neste critério, e com o intuito de obter uma aproximação de qualidade aos resultados experimentais, são re-escritas as aproximações mecânicas (Eq. 2.38) e (Eq. 2.39) da forma:

$$c_2 \cdot (\frac{\sigma_2}{R_{\perp}^{(+)A}})^2 + (\frac{\tau_{21}}{R_{\perp\perp}^{A}})^2 + c_1 \cdot (\frac{\sigma_2}{R_{\perp}^{(+)A}}) = 1 \tag{2.40}$$

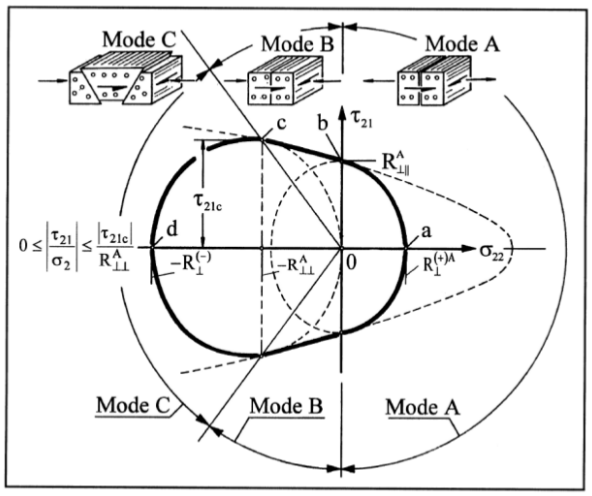

Figura 6: Modos A,B,C de falha da matriz associados ao envelope de Mohr no plano σ_{22}, τ_{21}, (PUCK; SCHÜRMANN, 1998).

A Equação 2.41 representa o IF usado neste modo de falha, de $\sigma_n \geq 0$, que pode-se escrever de forma extensa como:

$$\sqrt{(\frac{\tau_{21}}{R^{A}_{\perp\parallel}})^2 + (1 - \frac{p^{(+)}_{\perp\parallel} R^{(+)A}_{\perp}}{R^{A}_{\perp\parallel}})^2 \cdot (\frac{\sigma_2}{R^{(+)A}_{\perp}})^2} + \frac{p^{(+)}_{\perp\parallel}}{R^{A}_{\perp\parallel}} \cdot \sigma_2 = IF_{modoA} \tag{2.41}$$

Neste modo de falha, o plano de falha possui um $\theta_{fp} = 0$ tal como se observa na Figura 6.

(b) **Modo B:** Neste modo, o plano de falha mantém $\theta_{fp} = 0$ do modo A, cuja formulação tem validade quando $\sigma_2 < 0$ e $0 \leq |\frac{\sigma_2}{\tau_{21}}| \leq \frac{R^A_{\perp\perp}}{|\tau_{21c}|}$, e se expressa como:

$$\frac{1}{S_{12}} \cdot (\sqrt{(\tau_{21})^2 + (p^{(-)}_{\perp\parallel} \cdot \sigma_2)^2} + p^{(-)}_{\perp\parallel} \cdot \sigma_2) = IF_{modoB} \tag{2.42}$$

(c) **Modo C:** A inclinação do plano de falha neste modo é maior que zero e o máximo 54°, tal como se pode observar na Figura 7 onde $\sigma_2 \leq 0$. A formulação do IF associado com este modo de falha é:

$$[(\frac{\tau_{21}}{2 \cdot (1 + p^{(-)}_{\perp\perp} \cdot S_{21})})^2 + (\frac{\sigma_2}{Y_c})^2] * \frac{Y_c}{-\sigma_2} = IF_{modoC} \tag{2.43}$$

É tem validade quando $0 \leq |\frac{\tau_{21}}{\sigma_2}| \leq \frac{|\tau_{21c}|}{R^A_{\perp\perp}}$. Neste três modos de falha da matriz, "p" é um parâmetro experimental de inclinação.

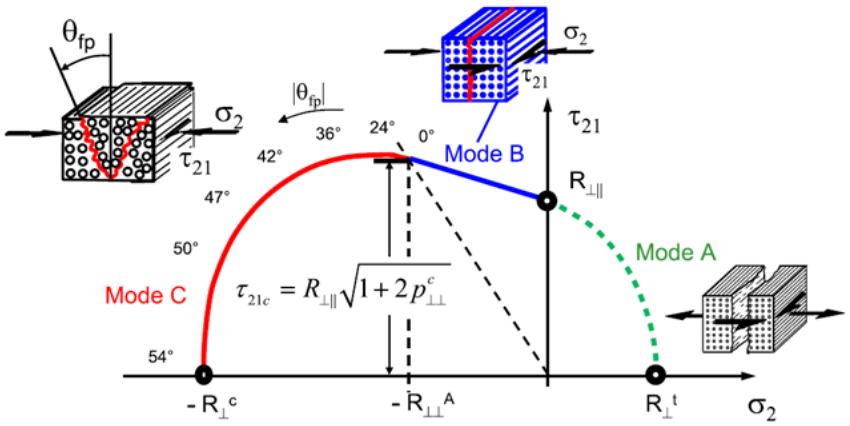

Figura 7: Modos A,B,C de falha da matriz associados ao envelope de Mohr no plano σ_2,τ_{21}. (PUCK; SCHÜRMANN, 1998)

2.5.2 LARC03

O critério de Larc03 baseou-se nas observações dos fenômenos de falha na fibra e na matriz (DAVILA; CAMANHO; ROSE, 2005), mas se diferença do critério de Puck porque não usa parâmetros não físicos e permite prever com precisão a falha da fibra.

Sob compressão na matriz ($\sigma_2 < 0$), o IF (**LaRC03#1**) associado com falhas na matriz, se define como:

$$IF = (\frac{\tau^T_{eff}}{(\tau_{23})_{ult}})^2 + (\frac{\tau^L_{eff}}{(\tau_{12})_{ult}})^2 \tag{2.44}$$

Onde τ_{eff}^{T} e τ_{eff}^{L} se definem por:

$$\tau_{eff}^{T} = -\sigma_2 \cdot cos(\alpha_0) \cdot (sin(\alpha) - (\eta)^T \cdot cos(\alpha)) \tag{2.45}$$

$$\tau_{eff}^{L} = cos(\alpha_0) \cdot (\|\tau_{12}\| + (\eta)^L \cdot \sigma_2 \cdot cos(\alpha)) \tag{2.46}$$

para,

$$\eta^T = \frac{-1}{tan(2 \cdot \alpha_0)} \tag{2.47}$$

$$\eta^L = \frac{(\tau_{12})_{ult} \cdot cos(2 \cdot \alpha_0)}{(\sigma_2^C)_{ult} \cdot cos^2(\alpha_0)} \tag{2.48}$$

O α_0 tem um valor aproximado de 53°. No caso de tração na matriz ($\sigma_2 > 0$), o IF (**LaRC030#2**), associado com falhas na matriz, se define como:

$$IF = (1-g) \cdot \frac{\sigma_2}{Y_{is}^T} + g \cdot (\frac{\sigma_2}{Y_{is}^T})^2 + (\frac{\tau_{12}}{S_{is}^L})^2 \tag{2.49}$$

Nesta definição do IF há embutidos vários dados experimentais e específicos para cada material que podem ser apreciados nas expressões:

$$g = \frac{\Lambda_{22}^0}{\Lambda_{44}^0} \cdot (\frac{Y_{is}^T}{S_{is}^L})^2 \tag{2.50}$$

As resistências que dependem da espessura t da lâmina, podem ser definidas conforme:

(a) Para lâminas grossas:

$$Y_{is}^T = 1,12 \cdot \sqrt{2} \cdot (\sigma_2^T)_{ult} \tag{2.51}$$

$$S_{is}^L = \sqrt{2} \cdot (\tau_{12})_{ult} \tag{2.52}$$

(b) Para lâminas delgadas:

$$Y_{is}^T = \sqrt{\frac{8 \cdot G_{Ic}(L)}{\pi \cdot t \cdot (\Lambda_{22})^0}} \tag{2.53}$$

$$S_{is}^L = \sqrt{\frac{8 \cdot G_{IIc}(L)}{\pi \cdot t \cdot (\Lambda_{44})^0}} \tag{2.54}$$

Finalmente, os valores de $(\Lambda_{22})^0$ e $(\Lambda_{44})^0$ são:

$$(\Lambda_{22})^0 = 2 \cdot (\frac{1}{E_2} - \frac{(\nu_{21})^2}{E_1}) \tag{2.55}$$

$$(\Lambda_{44})^0 = \frac{1}{G_{12}} \tag{2.56}$$

No caso **LaRC030#3**, associado com falhas nas fibras, onde a fibra é submetida à esforços de tração, tem-se o IF com a simples expressão:

$$IF = \frac{\epsilon_{11}}{\epsilon_{11}^T} \tag{2.57}$$

No caso **LaRC030#4**, onde a fibra é submetida a esforços de compressão, tem-se:

$$IF = \frac{\|\tau_{12}^m\| + (\eta^L) \cdot \sigma_{22}^m}{S_{is}^L} \tag{2.58}$$

O IF **LaRC030#5** usado nos casos de compressão das fibras com matriz tracionada é:

$$IF = (1-g) \cdot \frac{\sigma_{22}^m}{Y_{is}^T} + g \cdot (\frac{\sigma_{22}^m}{Y_{is}^T})^2 + (\frac{\tau_{12}}{S_{is}^L})^2 \tag{2.59}$$

O IF (**LaRC030#6**) usado nos casos de falha na matriz sob compressão biaxial é:

$$IF = (\frac{\tau_{eff}^m T}{S^T})^2 + (\frac{\tau_{eff}^m L}{S_{is}^L})^2 \tag{2.60}$$

Toda esta variedade de critérios de falhas evidencia a constante procura por modelos que representem matematicamente o comportamento dos materiais quando submetidos a esforços mecânicos, seja este um modelo simples como no caso dos critérios extremos (máxima tensão, máxima deformação) ou mais complexo como no casos dos critérios de Puck e Larc. Observa-se que apesar da longa data daqueles critérios mais simples os mesmos permanecem vigentes e aplicáveis para qualquer tipo de material quando sejam conhecidas os limites teóricos de aplicação.

Por exemplo no caso do critério de máxima tensão, o mesmo parece ser mais útil em materiais reforçados que em materiais metálicos e isotrópicos dado que no caso do compósito com reforço alinhado a elevada resistência da fibra impede que o envelope de falha se deforme. Na medida que uma lâmina isotrópica, com valores de resistência ao cisalhamento (τ_{12}) similares aos valores de resistência à tração (σ_1), for rotacionada aprecia-se uma deformação no envelope de falha passando de ter 4 vértices para ter 6 vértices o que não é conveniente dado que cada vértice representa uma singularidade indesejada dentre o

conjunto de pontos que conformam este envelope. Os vértices, ou pontos não suaves, são apenas aceitos quando modelam com exatidão um evento no processo de fratura como um delâminado, fratura dúctil, etc., mas sua aparição não justificável representa uma incomoda deformação da curva fechada que o envelope constitui. Deformação esta que poucas vezes aparece quando for rotacionada (um ângulo θ) uma lâmina reforçada porque a elevada resistência ao cisalhamento da fibra mantém a faixa criada pela desigualdade $-\tau_{12} \leq -cos(\theta) \cdot sin(\theta) \cdot \tau_x + cos(\theta) \cdot sin(\theta) \cdot \tau_y + (cos(\theta)^2 - sin(\theta)^2) \cdot \tau_{xy} \leq \tau_{12}$ o suficientemente afastada das outras faixas associadas ao envelope.

Os critérios quadráticos como Tsai-Hill, Tsai-Wu, Hoffman, e Azi-Tsai por estar associados com as curvas cônicas permite construir envelopes mais suaves sem vértices, exceto no caso do envelope de Azi-Tsai que foi uma pretendida melhora do antigo envelope de Tsai-Hill pouco aceito pela comunidade científica mundial justamente pelos vértices e descontinuidades que criam incerteza e pouco ajudam na modelagem do processo de fratura.

2.6 Fundamentos de informática

A implementação destes dez (10) envelopes de falha precisou de um domínio de informática significativo para superar obstáculos técnicos próprios de um projeto em contínua reformulação e feito por empreendedores autodidatas.

Os elementos mais exteriores que o usuário percebe dos módulos construídos, tanto no framework Django como no sistema operacional Android (KARCH, 2010), neste mestrado, são as interfaces que se definem como aquelas entidades que permitem um fluxo controlado de dados entre o sistema de processamento e o usuário (TANENBAUM, 2003). Nos desenvolvimentos WEB, ou seja naqueles sistemas de informática cujas entradas e saídas usam os protocolos de Internet (LAMBERT et al., 2005), estas interfaces são feitas na linguagem HTML que pode ser interpretada por qualquer navegador de internet (GATLIN, 2009) que converte o código escrito nesta linguagem em elementos gráficos para facilitar a interação do usuário.

O HTML tem evoluído de uma forma muito ordenada desde o começo da Internet educativa nos anos 90 o que faz com que esta linguagem de programação seja implementada pela maioria dos frameworks WEB atuais como Django (BENNETT, 2008). Muitas são as vantagens do HTML e os pesquisadores da área de informática educativa (TATNALL; OSORIO; VISSCHER, 2004), salientam a capacidade desta linguagem para criar vários caminhos de informação implícitos, o que facilita as atividades didáticas focadas na Internet. O HTML por ser uma linguagem chave na evolução da Internet desempenhou um amplo papel histórico que deve ser levado em conta na análise dos frameworks WEB como o Django, sendo este papel resumido como:

(a) Papel de suporte essencial da computação gráfica na Internet:

A pesar dos esforços individuais que vários fabricantes de software comercias fizeram nos anos 90 (GUREWICH; GUREWICH, 1994) para desenvolver e incluir interfaces gráficas junto com os compiladores (AHO et al., 2011) por eles desenvolvidos, não foi até a difusão massiva do HTML que começou a era de computação gráfica focada na Internet. Foi por esta razão que apareceram bibliotecas, ou seja funções computacionais disponibilizadas para ser usadas facilmente pelos programadores, como FLOT (GOOGLE, 2007–2014), desenhada especialmente para ser inseridas em interfaces HTML e permitir a realização de gráficos com velocidade e simplicidade.

(b) Papel de ferramenta principal na construção de conteúdos transparentes:

O código HTML é transparente, ou seja, assim como o navegador vê, o usuário também pode vê-lo. Assim, as novas gerações de programadores podem adquirir um bom domínio desta linguagem de forma autodidata por meio da aplicação prática de simples tutoriais que explicam, mais que as estruturas sintáticas e léxicas da linguagem (AHO et al., 2011), vários exemplos de engenharia reversa analisando páginas já publicadas na Internet.

(c) Papel de constante integração com novas tecnologias:

Uma das grandes facilidades das interfaces WEB feita com HTML é sua facilidade de aceitar fragmentos de códigos de outras linguagens de programação (DAYLEY, 2013) para auxiliar no controle dos fluxos de informação. Esta facilidade de integração faz da camada visual de um frameWork WEB uma entidade em constante evolução para incorporar novas tecnologias.

Um servidor WEB é um computador com capacidade de atender pedidos de informação (TANENBAUM, 2003) por meio dos protocolos de internet como HTTP, FTP, etc. integrando banco de dados e interpretes das linguagens de programação das interfaces WEB tais como PHP (CONVERSE; PARK, 2003), PERL (SANCHEZ, 2010), CGI (GUELICH; GUNDAVARAM; BIRZNIEKS, 2000), JavaScript (FLANAGAN, 2006), etc.

Um framework WEB é um programa incompleto que pode ser completado para obter um sistema com funções específicas num servidor WEB. O Django é um framework WEB gratuito e usado industrialmente por companhias como Instagram (LINASCHKE, 2011). Pode ser instalado em servidores APACHE (LAURIE; LAURIE, 2003), mas seu elevado consumo de memória RAM obriga que seja configurado com cuidado. Este framework é feito exclusivamente em Python (LUTZ, 2013) e implementa uma arquitetura do tipo Modelo (Model), Template (Template) e Vista (View) MTV o que permite uma programação ordenada e produtiva. Esta arquitetura MTV permite um planejamento detalhado das operações de codificação ao classificar as necessidades de programação

conforme a estrutura do modelo OSI de camadas (TANENBAUM, 2003). Assim por exemplo no desenvolvimento deste módulo foi aproveitada esta intensa separação entre a camada visual ou "View"(implementada em HTML) e a camada de Modelo (implementada em Python) para definir o mecanismo de teste do código final.

Para conhecer a implementação da arquitetura MTV no Django foi preciso consultar o manual oficial (BENNETT, 2008) concluindo que é preciso apenas configurar determinados arquivos Python deste frameWork para ativar uma determinada aplicação com ele desenvolvido, sendo estes arquivos:

settings.py: Este é o arquivo principal de configuração onde foi preciso indicar na variável BASE_DIR um endereço para cada computador onde foi feita a programação, e indicar a pasta comum de trabalho, contida no Dropbox, adicionando o endereço na estrutura Django de dados chamda de "sys.path". Já a variável ”DATABASES” foi configurada para usar o banco de dados MySQL.

urls.py: Este arquivo permite definir quais são os URL que devem ser atendidos pelas funções implementadas no frameWork. O passe de argumentos via URL (DAYLEY, 2008) foi uma potencialidade não usada no projeto macro mas de fácil implementação por meio de simples expressões regulares (AHO et al., 2011).

views.py: Este arquivo contém as funções que processam os argumentos de entrada pela interface de usuário, e também funções de processamento intermediário. No projeto macro Mehg-G, foram colocadas diretamente as funções numéricas neste arquivo o que gera textos extensos de difícil manutenção o que pode se evitar usando um sistema de referencia indireta por meio da funcionalidade de wrap do python.

Além destes arquivos python (definidos com a extensão ".py") (MARTELLI, 2006), o Django provê um conjunto de pastas na estrutura de diretórios onde podem se colocar diversas entidades WEB tais como:

Templates e código JavaScript: Um template Django permite criar um conteúdo WEB dinâmico usando as variáveis transmitidas pelo frameWork e previamente definidas nas funções contidas no arquivo **views.py**. Estes são colocados na pasta de templates do módulo onde é possível criar tantas pastas quanto necessárias para organizar os templates e os códigos JavaScript auxiliares, sendo preciso apenas indicar o endereço para sua efetiva localização por parte das funções de saída HTTP.

Arquivos estáticos: Os códigos HTML mais simples geralmente precisam de arquivos fixos como imagens ou arquivos de texto simples, sendo possível colocar neste diretório estático qualquer um destes arquivos sem que seja preciso conceder maiores permissões para uma visualização no template. No módulo desenvolvido neste mestrado deixou-se de usar arquivos estáticos com o intuito de evitar potenciais brechas de segurança, sendo usados apenas aqueles arquivos de imagens já usados no projeto macro, tais como logo,

ícones, etc.

Tecnicamente não existe aplicação WEB sem banco de dados, sendo possível defini-lo como um sistema de informática (SUEHRING, 2011) com capacidade de responder inteligentemente a qualquer consulta que lhe seja formulada numa linguagem específica e clara. Neste mestrado, foi usado no módulo Django, o banco de dados MySQL por ser gratuito, eficiente e intensamente documentado sendo possível configurá-lo para atender numerosas consultas simultâneas conforme as potencialidades do servidor WEB que lhe suporte. No módulo Android foi usado o banco de dados SQLite (ALLEN; OWENS, 2011) sendo este algo similar ao MySQL por ter a mesma linguagem relacional de consulta SQL (DATE, 2015).

O python é uma linguagem de programação simples, interpretado, gratuito que pode ser instalado nos sistemas operacionais Microsoft Windows e Linux, ao redor do qual foram desenvolvidas inúmeras bibliotecas igualmente livres e gratuitas. Por exemplo, a biblioteca **numpy** possui funções matemáticas de uso comum na engenharia (ARFKEN; WEBER, 2005), e a biblioteca **matplotlib** possui ferramentas para construir gráficos de forma rápida e simples. Outra vantagem do python é o fato de poder ser usado tanto dentro de um framework, para codificar funções complexas, como num código simples de poucas linhas (na linha de comando na modalidade de shell), o que permite prototipar funções com a técnica de construção de crescimento constante.

O JavaScript (FLANAGAN, 2006) é uma linguagem de programação usada intensa mente na programação das interfaces WEB que deve ser executado no navegador web do usuário. Uma versão evoluída do JavaScript é o JQUERY que representa uma biblioteca de funções escritas em JavaScript que foi publicada com a intenção principal de facilitar o uso do JavaScript e evitar que a diversidade de navegadores WEB criasse uma grave distorção nesta linguagem. Um dos principais usos do JQUERY é facilitar o tratamento dos eventos reportados pela interface gráfica, sendo preciso definir um evento como uma alteração feita pelo usuário na sua interface com o intuito de receber ou transmitir alguma informação.

O JSON (BASSETT, 2015) é um formato para comunicar dados entre diversas aplicações WEB que se estrutura como um dicionário (PATEL, 2015), ou seja, para cada entrada ou palavra definida está associada uma sequência de dados cujo significado depende da entrada. Uma forma de integrar facilmente o JSON no python é mediante a biblioteca chamada de "json" e o framework Django possui a classe **django.http** para poder gerar saídas no protocolo de internet HTTP.

O MATLAB (HUNT et al., 2006) é um software acadêmico e comercial muito usado na engenharia pela simplicidade que oferece para ingressar matrizes, analisar equações diferenciais simples e gerar gráficos num único ambiente integrado onde são combinadas ferramentas de programação e de visualização de dados. A boa aceitação desta ferramenta

por parte da comunidade mundial de engenheiros de diversas áreas garante a vigência da linguagem MATLAB pelos próximos anos (KATTAN, 2010) fazendo com que este software seja atrativo para futuros profissionais da engenharia de materiais.

Um ambiente de desenvolvimento integrado, ou IDE pelas suas siglas em inglês, é um software que permite desenvolver uma aplicação de forma ordenada e rápida (DEITEL; DEITEL, 2015), integrando um conjunto de ferramentas tipicas do desenvolvimento de software como os depuradores de código (debugger), emuladores, etc. (OKUYAMA; MILETTO; NICOLAO, 2014). O uso de uma IDE adequada permite acelerar o ritmo de produção (MASSARI, 2016) porque a mesma permite detectar e corrigir eventuais erros com maior facilidade. Uma IDE pode possuir muitas e diversas ferramentas, mas no caso da IDE usada neste projeto do tipo Android Studio (YENER; DUNDAR, 2016) é preciso explicar que a mesma possui um elemento chamado gradle que permite construir os arquivos de distribuição, chamados de APK, de forma rápida e segura usando as devidas chaves de segurança que a plataforma da Google Play exige.

Um sistema operacional é um conjunto de softwares que garantem o funcionamento do computador (TANENBAUM, 2010) e permite que o mesmo possa prestar determinados serviços em condições de eficiência e segurança informática aceitáveis. O sistema operacional Android é usado na maioria dos dispositivos moveis de telefonia celular, como telefones celulares inteligentes, comercializados no Brasil (CANALTECH, 2014) e pode-se definir este como um sistema operacional otimizado para ser usado em dispositivos eletrônicos com minima memoria RAM e processadores de mínimo poder computacional que possui uma máquina virtual java (FRIESEN, 2013) que pode ser usada para executar códigos feitos nesta linguagem de programação.

O java como linguagem de programação é classificado (DEITEL; DEITEL, 2015) como uma linguagem de programação imperativa com elementos de programação funcional, muito similar a antiga linguagem C que utiliza o paradigma de programação orientada a Objetos (SERSON, 2000), de forma intensa. A linguagem de programação java focado nos sistemas operacionais android respeita o estándar definido internacionalmente (GUIHOT, 2012) perante a IEEE Standars Assosiation, mas geralmente é usado uma estrutura que cada IDE de trabalho define, por isso vários autores ((FRIESEN, 2013),(GUIHOT, 2012)) usam a expressão "java android" para salientar um estilo de programação em java que arquiteta os objetos de uma forma conveniente para o sistema operacional Android.

O paradigma de programação orientada a objetos (POO) permite criar um código mais estruturado que pode ser lido e mantido com maior facilidade que os códigos planos não Orientados a Objetos e no caso de java é impossível codificar sem considerar esta filosofia de desenho o que tem criado comentários técnicos que classificam, de forma explícita, esta linguagem de programação como muito prolixa (DOYLE; STRETCH, 1987). Um exemplo de linguagens de programação que usam este paradigma são: java, javascript,

python, entre muitas outras.

Na POO é preciso definir e codificar objetos os quais são elementos de código que tentam representar um objeto de uso comum na realidade. Cada um destes objetos é definido por uma classe que pode se entender como um segmento do código onde se especifica tanto os elementos de dados do objeto como os métodos, também chamados de funções, que este possa implementar. Este paradigma de programação permite a reutilização de classes (definições prévias de objetos) por meio de mecanismos de expansão que muda de uma linguagem de programação para outra. No caso de java é possível aproveitar as classes já definidas, assim como definir polimorfismo e sobrecargas.

Outro conceito essencial neste trabalho é o relativo às expressões regulares as quais são expressões simbólicas que ao ser interpretadas permitem criar determinadas sequências de dados (AHO et al., 2011), sendo usadas para dar determinada capacidade de interpretação linguística num framework. Geralmente, os frameworks WEB ganham a capacidade interpretar URLs por meio de uma configuração indicada em expressões regulares, e no caso de Django o uso destas expressões permite criar um filtro de argumentos de entrada muito eficiente.

2.7 Fundamentos Pedagógicos

A nível nacional, no grupo de ensino de física da Universidade Federal de São Carlos na década de 70 foram dados os primeiros passos no uso do computador no processo de ensino (FILHO, 2013). Desde então, a incorporação das novas tecnologias informáticas é um fator comum, e até obrigatório, na maioria dos cursos de ciências e engenharias. Analisando este fato histórico é importante comentar que a iniciativa de incorporar novas tecnologias no processo educativo universitário parece obedecer, do ponto de vista pedagógico, a fatores como:

(a) Grau de domínio dos conceitos básicos:

Os teóricos do ”desenho de software” educativo salientam que é inútil projetar qualquer software se não houver clareza sobre os conceitos que serão transmitidos (TATNALL; OSORIO; VISSCHER, 2004) e estes constituem um conjunto completo de idéias mutuamente representativas. Este postulado educativo permite entender que os softwares educativos são gerados por grupos de pesquisa e de ensino que dominem vários materiais didáticos básicos (apostilas, mapas de conhecimentos, discurso didático, etc.) reproduzindo o fluxo de informação na forma computacional.

(b) Visão detalhada da totalidade do processo educativo:

Além de um significativo domínio conceitual é preciso entender o processo educativo para poder aumentar a utilidade do software em desenvolvimento. Uma visão integrada do

conjunto de etapas de cada processo pedagógico permite harmonizar as potencialidades das ferramentas computacionais com as necessidades didáticas. Por isso, a qualidade didática de um software educativo depende das considerações pedagógicas que sejam incorporadas no momento de projetar o fluxo de dados entre a ferramenta e o aluno.

(c) Capacidade computacional dos participantes do processo educativo: Entre as considerações de informática educativa pode-se considerar a capacidade computacional de todos os agentes envolvidos (FREIRE; NOGUEIRA, 1989), e apesar da tendencia tecnológica de gerar equipamentos mais poderosos e baratos, é importante levar em conta as eventuais desigualdades que caraterizam o sistema educacional nacional. É sabido que aquelas iniciativas de incorporar novas tecnologias sofrem de grande inércia quando o público alvo não possui um domínio técnico básico que lhe garanta uma fácil operação com as interfaces computacionais. Por isso, um software educativo geralmente conquista rapidamente um público quando é projetado para estudantes do nível superior.

(d) Facilidade de treinamento: A facilidade de treinamento prevista para qualquer software educativo é um fator determinante para garantir o máximo aproveitamento de cada uma das funcionalidades programadas. Historicamente, o software totalmente aberto facilita a criação dos manuais de treinamento, na forma de textos simples ou tutoriais, sendo quase obrigatório que o software educativo seja totalmente aberto para outros grupos e tenham a capacidade de aprofundar sua compreensão e uso computacional. A falta de treinamento prejudica a difusão de um software educativo porque sem uma descrição clara do mecanismo básico de uso, gera-se uma desmotivação nos usuários. No caso da UFRGS, com uma das mais antigas faculdades de computação do Brasil, e com cursos superiores focados na informática educativa, pode-se apreciar a consideração destes fatores pedagógicos nos trabalho de conclusão de curso, dissertação de mestrado ou tese de doutorado.

Uma das considerações pedagógicas que devem ser levadas em conta no momento do desenho das interfaces WEB é a teoria da autonomia e da inclusão digital, que ajudam de maneira especial na eficiência didática do módulo desenvolvido neste mestrado. A inclusão digital educativa (BONILLA; PRETTO, 2011), definida como o grau de participação dos potenciais usuários do software num processo educativo qualquer focado neste, é garantida por um software educativo de total acesso pela internet. Sendo maior esta inclusão quando o público alvo é do nível superior técnico ou universitário. Neste caso, a inclusão digital educativa ajuda no desenvolvimento da pedagogia da autonomia (FREIRE, 2000) por parte do usuário que pode explorar de forma autônoma as interfaces WEB desenvolvidas.

3 FERRAMENTAS UTILIZADAS

Por ser um trabalho intensamente focado no desenvolvimento de software, com um elevado componente numérico, as ferramentas usadas são essencialmente softwares livres e um computador pessoal do tipo portátil com os sistemas operacionais Windows e Linux Ubuntu.

Para implementar os critérios e envelopes de falha é preciso programar como funções computacionais aquelas funções e construções matemáticas que os representam. Como cada representação matemática gera um fluxo de dados do tipo numérico, é preciso preservar a informação nela contida. Para garantir esta meta, é preciso analisar cada uma das ferramentas computacionais do ponto de vista numérico e desprezar aquelas que potencialmente destroem dados do tipo numérico flotante.

No caso do módulo Django, aqui desenvolvido, foi aproveitada a experiência prévia do grupo de desenvolvimento selecionando as ferramentas já usadas no projeto Mehg-G pois são de ótima qualidade numérica. E no caso do módulo Android, parte deste sistema informático, as ferramentas de trabalho foram escolhidas conforme a experiência do autor em desenvolvimento de aplicativos escritos em Java para dispositivos moveis com o sistema operacional Android.

O sistema operacional do computador pessoal usado no desenvolvimento não foi algo determinante dado que as ferramentas necesárias para elaborar o módulo Django estão disponiveis gratuitamente tanto para o sistema operacional Microsoft Windows (SOUZA, 2016) como Linux(NEGUS, 2015). Disponibilidade que se repete com as ferramentas necessarias para elaborar o módulo Android dado que a IDE Android Studio e ferramentas derivadas estão disponiveis gratuitamente tanto para o sistema operacional Linux como Windows. A constante evolução do projeto macro obriga que seja considerada a versão do sistema operacional que suporte os pacotes e versões usados para garantir futuros aprimoramentos do módulo aqui desenvolvido.

O grau e qualidade da documentação de uma ferramenta computacional determina em parte sua aceitação pela comunidade acadêmica ou industrial. Neste trabalho foram usados softwares com documentação não nula, como o MySQL que possui uma excelente documentação oficial em contraste com a biblioteca FLOT (GOOGLE, 2007–2014) fracamente documentada. Outra das bibliotecas mininamente documentada foi a Chartist (KUNZ, 2017) usada para construir os gráficos testes com os quais foram espelhados os envelopes de falha gerados pelo módulo Android.

A lista de ferramentas computacionais usadas neste desenvolvimento encontra-se nas Tabelas 3 e 4, separadas conforme o módulo na qual foi usada, listadas na sua versão

Linux principalmente por razões econômicas. Algumas ferramentas foram usadas tanto no desenvolvimento de um módulo como no outro dado seu carácter de ampla utilidade como no caso dos editores de texto, mas estas ferramentas não são listadas dada sua simplicidade.

A escolha das ferramentas foi feita desde o começo deste mestrado, dedicando um tempo especial à análise das vantagens e desvantagens de cada uma porque a eficiência da programação dependia das especificações e potencialidades de cada ferramenta assim como sua integração operativa. Foi preciso estudar o manual de cada ferramenta evitando assim incompatibilidades por obsolescência.

Tabela 3: Softwares usados, no módulo Django, e principais caraterísticas técnicas aproveitadas.

Software	**Característica técnica aproveitada**
Python versão 2.7	Permite uma programação numérica rápida
Django versão 1.5	Framework bem estruturado
Flot versão 0.8.3	Representação gráfica de qualidade
Jquery versão 1.1 (2010)	Fácil programação das funções da interface web
Ajax versão 2010	Maior controle do HTML
MySQL versão 5.5.22	Eficiência na administração do banco de dados

Tabela 4: Softwares usados, no módulo Android, e principais caraterísticas técnicas aproveitadas.

Software	**Característica técnica aproveitada**
Android Studio 3.3	Android Studio 3.3
SQLite versão 3.11.0	Eficiência na administração do banco de dados
Android Emulator	Permite instalar rápidamente um apk no emulador
Gradle versão 3.3	Rápida compilação e atualização do código no emulador

A ferramentas informáticas foram usadas de forma combinada com o intuito de obter o máximo proveito, por exemplo a biblioteca Flot foi usada com python para poder visualizar diretamente o comportamento numérico da biblioteca numpy.

Na construção do módulo Android, uma ferramente de grande utilidade foi o gradle 3.3.0 que permite poupar tempo de atualização nos testes de cada nova versão do código.

Outra ferramenta de significativa utilidade é a consola de aplicativos da Google Play, onde pode-se obter relatórios de desempenho do APK testado pela companhia Google em diferentes versões de Android e diferentes modelos de telefones celulares. Esta mesma consola permite definir usuários testadores e receber destes avisos de eventuais erros do APK os quais são essenciais dado que existe uma ampla variedade de dispositivos mobiles sobre os quais pode ser usado o APK produzido. Cabe salientar que está é a única ferramenta comercial usada dado que é uma tendência mundial distribuir os APK usando

Tabela 5: Biblioteca Python e Django usadas.

Libraria	**versão.**
beautifulsoup4	4.4.1
cycler	0.9.0
Django	1.9.1
django-allauth	0.24.1
django-appconf	1.0.1
django-bootstrap3	6.2.2
django-compressor	1.6
django-registration-redux	1.2
matplotlib	1.5.1
mysqlclient	1.3.7
numpy	1.10.4
oauthlib	1.0.3
pyparsing	2.1.0
python-dateutil	2.4.2
python-openid	2.2.5
pytz	2015.7
rcssmin	1.0.6
requests	2.9.1
requests-oauthlib	0.6.0
rjsmin	1.0.12
six	1.10.0
wheel	0.24.0
xlwt	1.0.0

a plataforma Google Play ao invés de facilitar diretamente o APK para o usuário, sendo uma plataforma econômica.

O constante crescimento do software Mehg-G, e a filosofia de desenvolvimento competitivo dos novos módulos, obriga que cada programador participante do projeto desenvolva um ambiente único de trabalho que apenas pode ser sincronizado quando se tenha certeza que os códigos gerados não oferecem instabilidade operacional para o projeto. E justamente para acompanhar esta política de desenvolvimento cooperativo, foram usados pastas compartilhadas no software Dropbox com a topologia de: (i) ambiente Django instalado numa pasta local no computador do participante, (ii) pasta comum de desenvolvimento temporal no Dropbox e direcionadas para cada ambiente Django.

As bibliotecas de python e Django usadas no desenvolvimento deste módulo estão listadas na Tabela 5. O Matlab usado foi na versão R2010b (7.11), que apesar de ser um pouco antiga garante que o código gerado pelo módulo possa ser usados em computadores de pequeno porte. A biblioteca de python chamada de matplotlib (TOSI, 2009) foi usada na etapa de testes por permitir uma rápida codificação no sistema operacional Linux sem que fosse preciso ativar uma IDE do tipo MATLAB.

4 DESENHO DOS ALGORITMOS E INTERFACES GRÁFICAS

O fato de existir softwares comerciais, como o software Helius da AutoDesk (AUTODESK, 2004-2016), com capacidade de construir envelopes de falha, motivou os esforços de programação para lhes iguala em velocidade e qualidade visual.

Para atingir uma velocidade, de execução comparável ao software comercial, foram considerados os algoritmos mais simples (CORMEN, 2009) sabendo que esta simplicidade garante uma boa velocidade de execução do código, e foi reduzido o trânsito de argumentos entre funções por meio de consultas diretas no banco de dados.

4.1 Desenho dos algoritmos

O desenho dos algoritmos foi divido em duas etapas: (i) implementação numérica do critério de falha, (ii) construção geométrica do envelope de falha.

4.1.1 Implementação numérica dos critérios de falha

A codificação, tanto em python, Java/Android como no javascript, começou com a escrita de fragmentos cada um representando uma função computacional simples com argumentos explícitos o que permitiu rápidos testes onde cada função representava um critério de falha. Esta codificação inicial permitiu compreender o comportamento numérico dos critérios e foi feita com ajuda do time de desenvolvimento do Mech-Gcomp.

Estas funções tem o ângulo θ como um argumento comum de entrada e representa a rotação da lâmina. Foi possível criar mais funções, usando como ponto de partida estas da etapa inicial, simplificando as funções numéricas implementadas aproveitando a elevada qualidade do banco de dados, o que permitiu uma reformulação da estrutura de argumentos destas para poupar memoria RAM, que seria útil caso seja preciso atender dezenas de usuários. Esta reformulação consistiu apenas em limitar o número de argumentos recebidos pela função, e o invés de receber as propriedades da lâmina estudada (ϵ_{T1}, ϵ_{C1}, ϵ_{T2}, ϵ_{C2}, γ_{12}, ν_{12}, E_1, E_2, G_{12} σ_{T1}, σ_{C1}, σ_{T2}, σ_{C2}, τ_{12}), recebe o identificador (id) da lâmina no banco de dados ou equivalente, sendo feita a consulta das propriedades dentro de cada função.

Agrupa-se os critérios de falha conforme: (i) **Grau do polinômio:** os critérios de máxima tensão e máxima deformação tem como formulação uma combinação, em união lógica, de polinômios de grau 1. Os demais critérios estão formulados com polinômios de

segundo grau. (ii) **Tipo de argumentos de entrada:** são considerados como critérios de falha de "argumentos mínimos" aqueles que recebem como argumentos de entrada uma informação simples como tipo de material, estado de esforços (ou deformações) em coordenadas globais, e ângulo de orientação da lâmina. Nesta categoria tem-se os critérios de máxima tensão, máxima deformação, Tsai-Hill, Azzi-Tsai e Hoffman. Os demais critérios recebem mais argumentos, por tentar conciliar modelos matemáticos com evidência experimental o que obriga que tanto os "argumentos mínimos" como os "argumentos experimentais" sejam considerados de maneira conjunta.

É importante salientar que a programação em python resultou mais fácil que a programação em MATLAB pois a primeira dispõe de muitas bibliotecas para implementar e manipular estruturas de dados de uma forma rápida, simples e eficiente (STEWART, 2014). De fato, o MATLAB foi usado apenas como ferramenta de verificação dos resultados obtidos com python e os códigos nele desenvolvidos, na sua versão final e apenas para fins didáticos, foram disponibilizados na interface gráfica de usuário para permitir uma fácil reprodução dos resultados gráficos gerados por este módulo Django.

4.1.2 Implementação geométrica do critério de falha

A implementação geométrica dos critérios de falha, ou seja a programação dos envelopes, foi simplificada utilizando os princípios de geometria analítica. Os primeiros envelopes de falha programados foram os de "máxima tensão" e de "máxima deformação", sendo possível adaptar o algoritmo do primeiro tipo de envelope levando em conta apenas uma transformação elástica de esforços em deformações. Os envelopes não lineares mais simples, ou seja sem argumentos experimentais de entrada, como o Azzi-Tsai, Hoffman e Tsai-Hill foram programados, e finalmente foram programados aqueles envelopes de falha polinomiais que usam dados experimentais.

Desenho do algoritmo geométrico para o envelope de máxima tensão

Apesar de possuir uma formulação matemática simples, como a união lógica de normalizações, o tratamento analítico deste critério foi divido em duas etapas: (i) estudo analítico do critério de falha considerando apenas a condição de igualdade , (ii) classificação geométrica dos pontos obtidos no estudo anterior por meio da teoria de grafos (ARFKEN; WEBER, 2005).

Para obter os pontos que representam os potenciais vértices foram resolvidos os sistemas lineares. Usando os valores, no caso do envelope de "máxima tensão" de $\alpha = 2$ e $\beta = 1$:

$$\begin{bmatrix} c \cdot c & s \cdot s \\ s \cdot s & c \cdot c \end{bmatrix} \cdot \mathbf{x} = \begin{bmatrix} (\sigma_1)^C_{ult} - \alpha \cdot s \cdot c \cdot \tau_{xy} \\ (\sigma_2)^C_{ult} + \alpha \cdot s \cdot c \cdot \tau_{xy} \end{bmatrix} \tag{4.1}$$

$$\begin{bmatrix} c \cdot c & s \cdot s \\ s \cdot s & c \cdot c \end{bmatrix} \cdot \mathbf{x} = \begin{bmatrix} (\sigma_1)^T_{ult} - \alpha \cdot s \cdot c \cdot \tau_{xy} \\ (\sigma_2)^T_{ult} + \alpha \cdot s \cdot c \cdot \tau_{xy} \end{bmatrix} \tag{4.2}$$

$$\begin{bmatrix} c \cdot c & s \cdot s \\ s \cdot s & c \cdot c \end{bmatrix} \cdot \mathbf{x} = \begin{bmatrix} (\sigma_1)^T_{ult} - \alpha \cdot s \cdot c \cdot \tau_{xy} \\ (\sigma_2)^C_{ult} + \alpha \cdot s \cdot c \cdot \tau_{xy} \end{bmatrix} \tag{4.3}$$

$$\begin{bmatrix} c \cdot c & s \cdot s \\ s \cdot s & c \cdot c \end{bmatrix} \cdot \mathbf{x} = \begin{bmatrix} (\sigma_1)^C_{ult} - \alpha \cdot s \cdot c \cdot \tau_{xy} \\ (\sigma_2)^T_{ult} + \alpha \cdot s \cdot c \cdot \tau_{xy} \end{bmatrix} \tag{4.4}$$

$$\begin{bmatrix} c \cdot c & s \cdot s \\ -\beta \cdot s \cdot c & \beta * s \cdot c \end{bmatrix} * \mathbf{x} = \begin{bmatrix} (\sigma_1)^C_{ult} - \alpha \cdot s \cdot c \cdot \tau_{xy} \\ (\tau_{12})_{ult} - (c \cdot c - s \cdot s) * \tau_{xy} \end{bmatrix} \tag{4.5}$$

$$\begin{bmatrix} c \cdot c & s \cdot s \\ -\beta \cdot s \cdot c & \beta \cdot s \cdot c \end{bmatrix} * \mathbf{x} = \begin{bmatrix} (\sigma_1)^T_{ult} - \alpha \cdot s \cdot c \cdot \tau_{xy} \\ (\tau_{12})_{ult} - (c \cdot c - s \cdot s) \cdot \tau_{xy} \end{bmatrix} \tag{4.6}$$

$$\begin{bmatrix} c \cdot c & s \cdot s \\ -\beta \cdot s \cdot c & \beta \cdot s \cdot c \end{bmatrix} \cdot \mathbf{x} = \begin{bmatrix} (\sigma_1)^T_{ult} - \alpha \cdot s \cdot c \cdot \tau_{xy} \\ -(\tau_{12})_{ult} - (c \cdot c - s \cdot s) \cdot \tau_{xy} \end{bmatrix} \tag{4.7}$$

$$\begin{bmatrix} c \cdot c & s \cdot s \\ -\beta \cdot s \cdot c & \beta \cdot s \cdot c \end{bmatrix} \cdot \mathbf{x} = \begin{bmatrix} (\sigma_1)^C_{ult} - \alpha \cdot s \cdot c \cdot \tau_{xy} \\ (\tau_{12})_{ult} - (c \cdot c - s \cdot s) \cdot \tau_{xy} \end{bmatrix} \tag{4.8}$$

$$\begin{bmatrix} c \cdot c & s \cdot s \\ -\beta \cdot s \cdot c & \beta \cdot s \cdot c \end{bmatrix} \cdot \mathbf{x} = \begin{bmatrix} (\sigma_2)^C_{ult} - \alpha \cdot s \cdot c \cdot \tau_{xy} \\ (\tau_{12})_{ult} - (c \cdot c - s \cdot s) \cdot \tau_{xy} \end{bmatrix} \tag{4.9}$$

$$\begin{bmatrix} c \cdot c & s \cdot s \\ -\beta \cdot s \cdot c & \beta \cdot s \cdot c \end{bmatrix} \cdot \mathbf{x} = \begin{bmatrix} (\sigma_2)^T_{ult} - \alpha \cdot s \cdot c \cdot \tau_{xy} \\ (\tau_{12})_{ult} - (c \cdot c - s \cdot s) \cdot \tau_{xy} \end{bmatrix} \tag{4.10}$$

$$\begin{bmatrix} c \cdot c & s \cdot s \\ -\beta \cdot s \cdot c & \beta \cdot s \cdot c \end{bmatrix} \cdot \mathbf{x} = \begin{bmatrix} (\sigma_2)^T_{ult} - \alpha \cdot s \cdot c \cdot \tau_{xy} \\ -(\tau_{12})_{ult} - (c \cdot c - s \cdot s) \cdot \tau_{xy} \end{bmatrix} \tag{4.11}$$

$$\begin{bmatrix} c \cdot c & s \cdot s \\ -\beta \cdot s \cdot c & \beta \cdot s \cdot c \end{bmatrix} \cdot \mathbf{x} = \begin{bmatrix} (\sigma_2)^C_{ult} - \alpha \cdot s \cdot c \cdot \tau_{xy} \\ (\tau_{12})_{ult} - (c \cdot c - s \cdot s) \cdot \tau_{xy} \end{bmatrix} \tag{4.12}$$

Onde c representa a função $cos(\theta)$ e s a função $sin(\theta)$, e o vetor de esforços é:

$$\mathbf{x} = \begin{bmatrix} (\sigma_x) \\ (\sigma_y) \end{bmatrix} \tag{4.13}$$

No caso de rotação com um ângulo múltiplo de $(\pi/2)$ pode-se ver que alguns determinantes se anulam (4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12) o que inviabiliza o algoritmo por ocorrência de erro numérico. Com o intuito de contornar esta situação, foi

criado outro algoritmo para estes ângulos, o qual não resolve os sistemas mas procura os vértices diretamente já que a matriz de rotação T, por ser unitária nunca gera um determinante nulo (ARFKEN; WEBER, 2005).

Tanto a rotina usada inicialmente, como a rotina deste último tipo de rotação são chamadas seletivamente, conforme o ângulo de rotação da lâmina, e não trabalham cooperativamente. Esta condição de chamada seletiva é aplicada para os envelopes de falha do tipo extremo, pois nos ângulos múltiplos de $(\pi/2)$ o cálculo se simplifica porque os elementos da matriz T se tornam binários o que permite poupar tempo computacional.

Nos ângulos que não são múltiplos de $(\pi/2)$, é preciso resolver todos os sistema matriciais (Equações 4.1- 4.12), o que consome significativo poder computacional. Cada sistema matricial permite obter um possível ponto do envelope, sendo preciso determinar quais pontos podem ser usados para construir a curva poligonal fechada que constitui o envelope de falha. Para atingir este objetivo é preciso desenhar um algoritmo levando em conta:

(a) Considerar os dois quadriláteros das faixas de cada par de desigualdades:

Já que este critério de falha define 3 faixas infinitas no plano de esforços (σ_x, σ_y), é possível formar no máximo 3 quadriláteros que podem ter ou não uma área comum. O parâmetro nulo τ_{xy} garante que exista uma área comum (exemplo na Fig. 8 (a)) e para valores elevados deste é impossível obter esta área comum (Fig. 8 (b)) o que se interpreta como a geração de um IF acima de 1.

(b) Determinar o número de pontos do envelope de falha: Estudando os 12 pontos, obtidos por eliminação gaussiana, é possível predizer quantos pontos terá o envelope aplicando as regras a seguir. Se os pontos solução dos sistemas com "τ" (Equações 4.5 -4.12) estão fora da área comum definida pelo quadrilátero cujos vértices são solução aos sistemas "sem τ" (Equações 4.1- 4.4), então o envelope de falha estará definido apenas por estes últimos 4 vértices. Caso contrário será preciso aprofundar na analise geométrica.

Quando algum dos pontos solução dos sistemas com "τ" estão na área construída com os vértices "sem τ" , o envelope tem um mínimo de 3 ou máximo de 6 vértices, sendo preciso testar cada um deles para determinar se pertencem ou não ao envelope. Este teste consiste em calcular o índice de falha (IF) associado ao ponto e descartá-lo quando supera o valor de 1 em qualquer uma das normalizações que define este critério de falha.

(c) Determinar os vértices de um quadrilátero que tenha um ponto interior ou tangente ao outro:

A combinação de 3 faixas infinitas pode gerar uma área comum delimitada por uma linha poligonal fechada de no mínimo 3 e no máximo 6 vértices, se algum dos vértices de um quadrilátero estiver contido (Fig. 9), no outro declarando-se este como parte do envelope de falha e inicia-se um mecanismo de procura dos vértices restantes. Para isso do

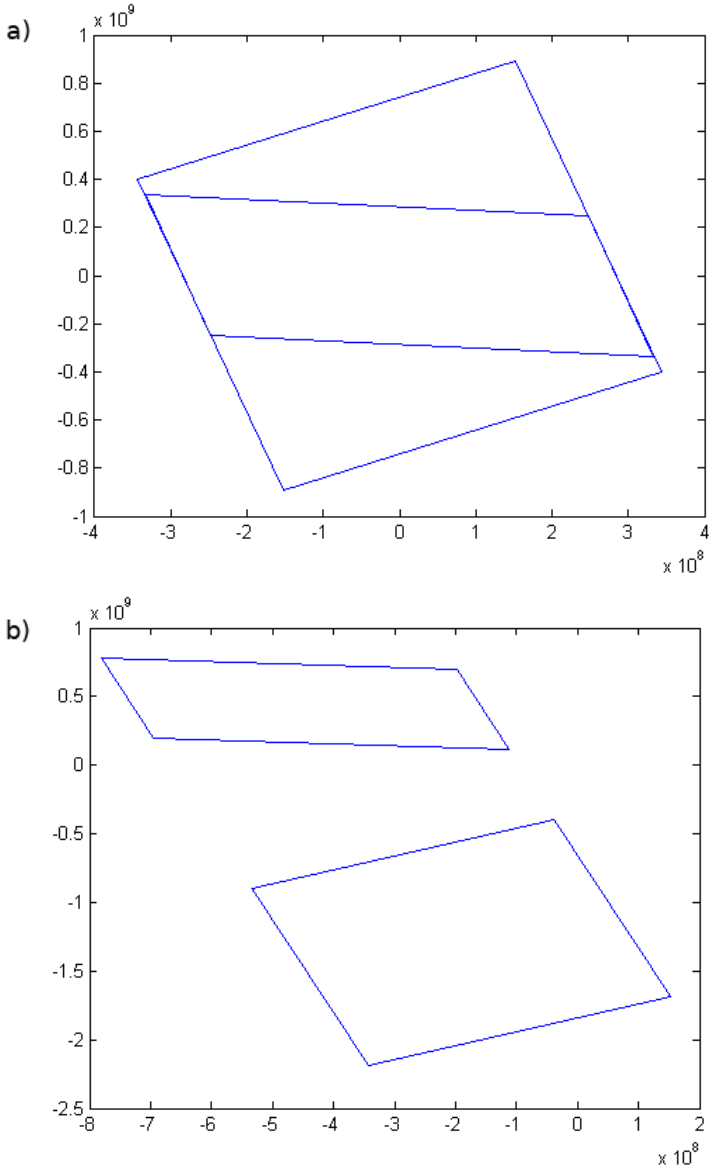

Figura 8: (a) Área comum entre as 3 faixas. Envelope de falha de 4 vértices. (b) Área comum não existente, impossibilidade de construção de envelope de falha. (Fonte: Autor)

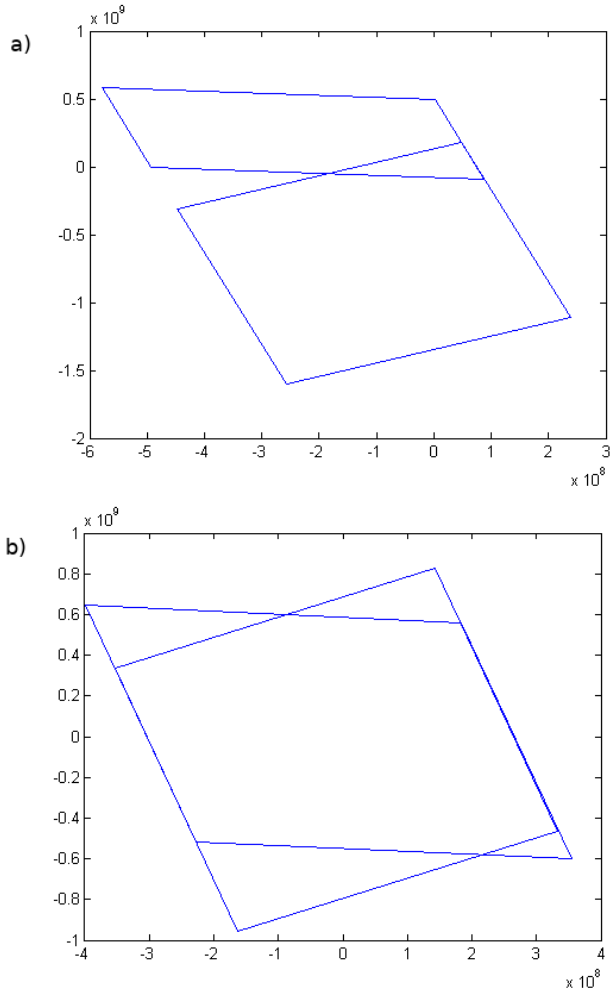

Figura 9: a) Envelope de falha de forma triangular (apenas um vértice do quadrilátero contido no outro). b) Envelope de falha com 6 vértices (2 vértices de um quadrilátero estão contidos no outro). (Fonte: Autor)

vértice anterior, segue-se a linha limite da faixa que o definiu até o momento em que esteja saindo de alguma das faixas. Este procedimento se repete com a outra linha definitória deste vértice inicial para assim determinar os novos vértices do envelope de falha. Quando se tem apenas um vértice contido, este é um envelope de falha do tipo triangular (Fig. 9a), o que permite concluir rapidamente o algoritmo.

Quando se tem mais de um vértice contido no outro quadrilátero (Fig. 9b), é preciso estudar todos os vértices contidos para determinar os fragmentos de linha que compõem o envelope de falha, ou seja aquele que parte de um vértice interno e finaliza num ponto de saída de faixa. Finalmente, são unidos todos estes fragmentos de linha para construir o envelope de falha.

Outro algoritmo que permite determinar quais são os vértices deste envelope de falha e a forma como devem conectar-se, consiste em avaliar o IF de cada um dos 12 pontos (Equações 4.1- 4.12) descartando aqueles que não sejam unitários. Agrupando

apenas aqueles pontos, possíveis vértices, com IF 1 é possível aplicar critérios de conexão da forma:

(a) Procurar o vizinho com ângulo maior: Considerando que uma aresta deste envelope de falha não pode "entrar" na área de IF menor de 1 e que duas arestas do envelope formam um ângulo cujo valor representa o máximo dentre os possíveis ângulos que podem ser construídos ao formar triadas com todos os vértices (ARFKEN; WEBER, 2005). Neste estudo angular é preciso usar uma estrutura de dados (CORMEN, 2009) para registrar o ângulo que cada combinação de três vértices possa gerar.

(b) Conetar na sequência ordenada:

São ordenados os vértices do envelope registrando primeiramente aqueles que estejam mais a esquerda. Conecta-se os vértices buscando o elemento comum nos extremos das triadas de ângulos "máximos". Por tanto escolhe-se aquela triada que esteja mais a esquerda e posteriormente conecta-se. Como resultado deste método obtêm-se dois caminhos chamados de "caminho superior" e "caminho inferior".

Desenho do algoritmo geométrico para o envelope de máxima deformação

Tecnicamente foi o mesmo algoritmo usado para o envelope anterior, funcionando no espaço das deformações locais, pois as desigualdades estão definidas em termos delas:

$$\begin{aligned} IF = (IF\ tração\ direção\ 1) \cap (IF\ tração\ direção\ 2) \\ \cap (IF\ compressão\ direção\ 1) \cap (IF\ compressão\ direção\ 2) \cap (IF\ cisalhamento) \end{aligned} \tag{4.14}$$

Em ambos algoritmos, "máxima tensão" e "máxima deformação", obtêm-se as propriedades da lâmina estudada, diretamente do banco de dados ou fonte de informação similar, a saída é gerada no formato especifico da biblioteca gráfica usada (Chartist (KUNZ, 2017), FLOT (GOOGLE, 2007–2014), GrapView (GEHRING, 2017)) para representar uma linha poligonal fechada se for possível. Cabe salientar que o teste de interseção dos quadriláteros criados (Fig. 8) é usado para determinar se o envelope de falha existe ou não, e no caso de não existir, estes algoritmos geram uma saída do tipo vetor nulo.

O cálculo e construção do envelope é feita no espaço de esforços (ϵ_x, ϵ_y) para finalmente transformar ao espaço de esforços globais (σ_x, σ_y) cada um dos vértices obtidos por meio da matriz de rigidez $\overline{Q}$ rotacionada.

Desenho do algoritmo geométrico para o envelope de Tsai Hill

Este algoritmo tem como entrada o número identificador **id** que identifica a lâmina no banco de dados, o ângulo de rotação da lâmina, o esforço cisalhante τ_{xy}, o estado de esforços, e o número de elementos que serão colocados no vetor de saída caso seja possível gerar um envelope de falha. Este fator se calcula em função dos valores extremos usados

(σ_{ult}), assim por exemplo no caso um material isotrópico ou transversalmente isotrópico, no plano 2-3 , caso no qual $(\sigma_2)^T_{ult} = (\sigma_3)^T_{ult}$ pode-se usar 50 como número de elementos de saída, ou seja como dimensão da estrutura de dados que serão representados no vetor de saída.

Neste algoritmo, ao invés de ser usada a formulação do IF em coordenadas locais, foi usada a formulação em coordenadas globais:

$$\begin{aligned}
IF_{tsai_hill} = {} & cos(\theta)^4\sigma_x^2 t_1^2 - cos(\theta)^4\sigma_x\sigma_y t_1^2 + cos(\theta)^4\sigma_y^2 t_2^2 \\
& + cos(\theta)^4 t_{12}^2\tau_{xy}^2 + 6cos(\theta)^3 sin(\theta)\sigma_x t_1^2\tau_{xy} - 2cos(\theta)^3 sin(\theta)\sigma_x t_{12}^2\tau_{xy} \\
& - 2cos(\theta)^3 sin(\theta)\sigma_y t_1^2\tau_{xy} + 2cos(\theta)^3 sin(\theta)\sigma_y t_{12}^2\tau_{xy} - 4cos(\theta)^3 sin(\theta)\sigma_y t_2^2\tau_{xy} \\
& - cos(\theta)^2 sin(\theta)^2\sigma_x^2 t_1^2 + cos(\theta)^2 sin(\theta)^2\sigma_x^2 t_{12}^2 + 2cos(\theta)^2 sin(\theta)^2\sigma_x\sigma_y t_1^2 \\
& - 2cos(\theta)^2 sin(\theta)^2\sigma_x\sigma_y t_{12}^2 + 2cos(\theta)^2 sin(\theta)^2\sigma_x\sigma_y t_2^2 - cos(\theta)^2 sin(\theta)^2\sigma_y^2 t_1^2 \\
& + cos(\theta)^2 sin(\theta)^2\sigma_y^2 t_{12}^2 + 8cos(\theta)^2 sin(\theta)^2 t_1^2\tau_{xy}^2 - 2cos(\theta)^2 sin(\theta)^2 t_{12}^2\tau_{xy}^2 \\
& + 4cos(\theta)^2 sin(\theta)^2 t_2^2\tau_{xy}^2 - 2cos(\theta) sin(\theta)^3\sigma_x t_1^2\tau_{xy} + 2cos(\theta) sin(\theta)^3\sigma_x t_{12}^2\tau_{xy} \\
& - 4cos(\theta) sin(\theta)^3\sigma_x t_2^2\tau_{xy} + 6cos(\theta) sin(\theta)^3\sigma_y t_1^2\tau_{xy} - 2cos(\theta) sin(\theta)^3\sigma_y t_{12}^2\tau_{xy} \\
& + sin(\theta)^4\sigma_x^2 t_2^2 - sin(\theta)^4\sigma_x\sigma_y t_1^2 \\
& + sin(\theta)^4\sigma_y^2 t_1^2 + sin(\theta)^4 t_{12}^2\tau_{xy}^2
\end{aligned} \tag{4.15}$$

Onde as constantes t1, t2, t12 se definem em função das resistências do material como: $t1 = \frac{1}{(\sigma_1)^T_{ult}}$, $t2 = \frac{1}{(\sigma_2)^T_{ult}}$, $t12 = \frac{1}{(\tau_{12})_{ult}}$. Usando a fórmula linear quadrática desta curva cônica:

$$a \cdot \sigma_x^2 + 2 \cdot b \cdot \sigma_x \cdot \sigma_y + c \cdot \sigma_y^2 + 2 \cdot d \cdot \sigma_x + 2 \cdot f \cdot \sigma_y + g = 0 \tag{4.16}$$

Onde θ representa o ângulo de rotação da lâmina. Pode-se definir os coeficientes em coordenadas globais usando τ_{xy}, como parâmetro da curva cônica (4.16):

$$a = cos(\theta)^4 \cdot t_1^2 + sin(\theta)^4 \cdot t_2^2 - cos(\theta)^2 \cdot sin(\theta)^2 \cdot t_1^2 + cos(\theta)^2 \cdot sin(\theta)^2 \cdot t_{12}^2 \tag{4.17}$$

$$\begin{aligned}
2b = {} & -cos(\theta)^4 \cdot t_1^2 + 2 \cdot cos(\theta)^2 \cdot sin(\theta)^2 \cdot t_1^2 - 2 \cdot cos(\theta)^2 \cdot sin(\theta)^2 \cdot t_{12}^2 \\
& - sin(\theta)^4 \cdot t_1^2 + 2 \cdot cos(\theta)^2 \cdot sin(\theta)^2 \cdot t_2^2
\end{aligned} \tag{4.18}$$

$$c = sin(\theta)^4 \cdot t_1^2 + cos(\theta)^4 \cdot t_2^2 - cos(\theta)^2 \cdot sin(\theta)^2 \cdot t_1^2 + cos(\theta)^2 \cdot sin(\theta)^2 \cdot t_{12}^2 \tag{4.19}$$

$$\begin{aligned} 2d = 6 \cdot cos(\theta)^3 \cdot sin(\theta) \cdot t_1^2 \cdot \tau_{xy} - 2 \cdot cos(\theta)^3 \cdot sin(\theta) \cdot t_{12}^2 \cdot \tau_{xy} \\ -2 \cdot cos(\theta) \cdot sin(\theta)^3 \cdot t_1^2 \cdot \tau_{xy} + 2 \cdot cos(\theta) \cdot sin(\theta)^3 \cdot t_{12}^2 \cdot \tau_{xy} \\ -4 \cdot cos(\theta) \cdot sin(\theta)^3 \cdot t_2^2 \cdot \tau_{xy} \end{aligned} \tag{4.20}$$

$$\begin{aligned} 2f = 2 \cdot cos(\theta)^3 \cdot sin(\theta) \cdot t_1^2 2 \cdot \tau_{xy} - 4 \cdot cos(\theta)^3 \cdot sin(\theta) \cdot t_2^2 \cdot \tau_{xy} \\ -2 \cdot cos(\theta)^3 \cdot sin(\theta) \cdot t_1^2 \cdot \tau_{xy} + 6 \cdot cos(\theta) \cdot sin(\theta)^3 \cdot t_1^2 \cdot \tau_{xy} \\ -2 \cdot cos(\theta) \cdot sin(\theta)^3 \cdot t_{12}^2 \cdot \tau_{xy} \end{aligned} \tag{4.21}$$

$$\begin{aligned} g = -1 + 8cos(\theta)^2 sin(\theta)^2 t_1^2 \tau_{xy}^2 - 2cos(\theta)^2 sin(\theta)^2 t_{12}^2 \tau_{xy}^2 + 4cos(\theta)^2 sin(\theta)^2 t_2^2 \tau_{xy}^2 \\ +cos(\theta)^4 t_{12}^2 \tau_{xy}^2 + sin(\theta)^4 t_{12}^2 \tau_{xy}^2 \end{aligned} \tag{4.22}$$

O comprimento dos eixos principais desta elipse se definem como:

$$eixo_a = \sqrt{\frac{2 \cdot (a \cdot f^2 + c \cdot d^2 + g \cdot b^2 - 2 \cdot b \cdot d \cdot f^2 - a \cdot c \cdot g^2)}{(b^2 - a \cdot c) \cdot (\sqrt{(a-c)^2 + 4 \cdot b^2} - (a+c))}} \tag{4.23}$$

e

$$eixo_b = \sqrt{\frac{2 \cdot (a \cdot f^2 + c \cdot d^2 + g \cdot b^2 - 2 \cdot b \cdot d \cdot f^2 - a \cdot c \cdot g^2)}{(b^2 - a \cdot c) \cdot (-\sqrt{(a-c)^2 + 4 \cdot b^2} - (a+c))}} \tag{4.24}$$

Para obter esta curva fechada é preciso satisfazer o discriminante da forma:

$$b^2 - 4 \cdot a \cdot c > 0 \tag{4.25}$$

O que pode se expressa em termos de propriedades mecânicas como:

$$(\frac{1}{(\sigma_1)_{ult}^T})^2 \cdot (\frac{1}{(\sigma_2)_{ult}^T})^2 - \cdot (\frac{1}{(\sigma_1)_{ult}^T})^2 \cdot (\frac{1}{2})^2 > 0 \Rightarrow 4 \cdot ((\sigma_1)_{ult}^T)^2 > ((\sigma_2)_{ult}^T)^2 \tag{4.26}$$

No caso de materiais isotrópicos esta condição, é satisfeita, mas no caso de outros materiais é preciso fazer um teste de discriminante antes de tentar construir um envelope de falha para eliminar as curvas do tipo abertas.

Já que a saída é representada no plano de esforços (σ_x,σ_y) é usada a formulação da elipse (ARFKEN; WEBER, 2005) diretamente em coordenadas globais. No caso isotrópico

poupamos o calculo do ângulo de inclinação da elipse θ_{elipse} , mas caso contrario é calculado e será usado para rotacionar os pontos da elipse mediante uma rotação de Euler usando:

$$\begin{bmatrix} cos(\theta_{elipse}) & sin(\theta_{elipse}) \\ -sin(\theta_{elipse}) & cos(\theta_{elipse}) \end{bmatrix} \tag{4.27}$$

Já os pontos da elipse são obtidos pela formula polar:

$$\left[r = \frac{raioMaior \cdot raioMenor}{\sqrt{(raioMenor \cdot cos(\theta_{varredura}))^2 + (raioMaior \cdot sin(\theta_{varredura}))^2}} \right] \tag{4.28}$$

Onde os raios elípticos maior e menor da elipse são calculados conforme (Eq. 4.23, 4.24) e os valores de $\theta_{varredura}$ estão entre 0 e 2π.

Como a posição da origem depende do parâmetro τ_{xy}, pode ser calculado como:

$$origem_x = \frac{c \cdot d - b \cdot f}{b^2 - a \cdot c} \tag{4.29}$$

e

$$origem_y = \frac{a \cdot f - b \cdot d}{b^2 - a \cdot c} \tag{4.30}$$

O único ciclo que este algoritmo possui é do tipo "for" para varrer a formula polar (Eq. 4.28) dentro do qual além do cálculo do valor do único raio é feita a rotação de Euler e se for preciso o deslocamento da origem. Esta última operação esta condicionada com o intuito de evitar operações aritméticas desnecessárias.

Desenho do algoritmo geométrico para o envelope de Azzi Tsai

Este envelope é similar ao algoritmo anterior exceto por uma etapa posterior de seleção dos valores em função do quadrante do plano de esforços locais em estudo. Para cada um dos 4 casos, o algoritmo chama uma função, conforme a curva cônica que deve construir, passando como argumentos os valores de resistência que devem ser usados. Destas curvas são escolhidos apenas os valores que satisfazem a condição do quadrante, com prévia transformação a coordenadas locais. A condição de quadrante se resume na Tabela 6.

Tabela 6: Relação entre as propriedades usadas na formulação do critério e o quadrante.

Quadrante	**Propriedades usadas**
$\sigma_1 < 0$ e $\sigma_2 < 0$	$propriedade_1 = (\sigma_1)^C_{ult}$, $propriedade_2 = (\sigma_2)^C_{ult}$, $propriedade_3 = (\sigma_1)^C_{ult}$
$\sigma_1 < 0$ e $\sigma_2 > 0$	$propriedade_1 = (\sigma_1)^C_{ult}$, $propriedade_2 = (\sigma_2)^T_{ult}$, $propriedade_3 = (\sigma_1)^T_{ult}$
$\sigma_1 > 0$ e $\sigma_2 < 0$	$propriedade_1 = (\sigma_1)^T_{ult}$, $propriedade_2 = (\sigma_2)^C_{ult}$, $propriedade_3 = (\sigma_1)^C_{ult}$
$\sigma_1 > 0$ e $\sigma_2 > 0$	$propriedade_1 = (\sigma_1)^T_{ult}$, $propriedade_2 = (\sigma_2)^T_{ult}$, $propriedade_3 = (\sigma_1)^T_{ult}$

São usadas estas propriedades na formula cônica, como em:

$$IF = (\frac{\sigma_1}{propriedade_1})^2 + (\frac{\sigma_2}{propriedade_2})^2 - \frac{\sigma_1 \cdot \sigma_2}{(propriedade_3)^2} + (\frac{\tau_{12}}{(\tau_{12})_{ult}})^2 \tag{4.31}$$

É importante indicar que o sentido usado na varredura do algoritmo que gera estas quatro elipses determina o sentido de construção do envelope de Azzi-Tsai. Foi escolhida uma varredura no sentido horário $(0, 2 \cdot \pi)$ sendo construído o envelope na ordem dos quadrantes: $(\sigma_1 > 0, \sigma_2 > 0)$, $(\sigma_1 < 0, \sigma_2 > 0)$, $(\sigma_1 < 0, \sigma_2 < 0)$, e $(\sigma_1 > 0, \sigma_2 < 0)$. Então transforma-se as coordenadas locais em globais em cada ponto.

Nesta caso também é necessário verificar, usando o descriminante (Eq. 4.25), se estamos perante uma curva cônica do tipo elipse porque no caso contrario será necessário emitir como saída numérica um vetor nulo dado que não é possível considerar uma linha não elíptica como envelope de falha Azzi Tsai.

Desenho do algoritmo geométrico para o envelope de Tsai-Wu

Foi o primeiro algoritmo desenhado com entrada de valores experimentais, sendo preciso ingressar o parâmetro biaxial $\sigma_{biaxial}$. Este algoritmo é bastante similar ao algoritmo de Tsai-Hil, sendo preciso considerar os coeficientes da equação cônica em coordenadas de esforços globais (σ_x, σ_y) na forma:

$$a = 2 \cdot H \cdot sin^2(\theta) \cdot cos^2(\theta) + (\frac{(2 * cos(\theta) * sin(\theta))^2}{((\tau_{12})_{ult})^2}) + \frac{sin(\theta)^4}{(\sigma_1^T)_{ult} \cdot (\sigma_1^C)_{ult}} + \frac{cos(\theta)^4}{(\sigma_2^T)_{ult} \cdot (\sigma_2^C)_{ult}} \tag{4.32}$$

$$\begin{aligned} b = H \cdot cos(\theta)^4 + H \cdot sin(\theta)^4 + (\frac{1}{((\tau_{12})_{ult})^2}) \cdot (-4 \cdot cos(\theta)^2 \cdot sin(\theta)^2) \\ + \frac{cos(\theta)^2 \cdot sin(\theta)^2}{(\sigma_1^T)_{ult} \cdot (\sigma_1^C)_{ult}} \\ + \frac{cos(\theta)^2 \cdot sin(\theta)^2}{(\sigma_2^T)_{ult} \cdot (\sigma_2^C)_{ult}} \end{aligned} \tag{4.33}$$

$$c = 2 \cdot H \cdot sin^2(\theta) \cdot cos^2(\theta) + (\frac{(2 \cdot cos(\theta) \cdot sin(\theta))^2}{((\tau_{12})_{ult})^2}) + \frac{cos(\theta)^4}{(\sigma_1^T)_{ult} \cdot (\sigma_1^C)_{ult}} + \frac{sin(\theta)^4}{(\sigma_2^T)_{ult} \cdot (\sigma_2^C)_{ult}} \tag{4.34}$$

$$\begin{aligned} d = 4 \cdot H \cdot \tau_{XY} \cdot (cos^3(\theta) \cdot sin(\theta) - sin^3(\theta) \cdot cos(\theta)) \\ +(\frac{4 \cdot \tau_{xy} \cdot cos(\theta) \cdot sin^3(\theta) - 4 \cdot \tau_{xy} \cdot sin(\theta) \cdot cos^3(\theta)}{((\tau_{12})_{ult})^2}) \cdot (\tau_{xy}^2) \\ +\frac{4 \cdot sin(\theta)^3 \cdot cos(\theta) \cdot \tau_{xy}}{(\sigma_1^T)_{ult} \cdot (\sigma_1^C)_{ult}} \\ +\frac{-4 \cdot cos(\theta)^3 \cdot sin(\theta) \cdot \tau_{xy}}{(\sigma_2^T)_{ult} \cdot (\sigma_2^C)_{ult}} \\ +(\frac{1}{(\sigma_1^T)_{ult}} - \frac{1}{(\sigma_1^C)_{ult}}) \cdot sin(\theta)^2 \\ +(\frac{1}{(\sigma_2^T)_{ult}} - \frac{1}{(\sigma_2^C)_{ult}}) \cdot cos(\theta)^2 \end{aligned} \tag{4.35}$$

$$\begin{aligned} f = 4 \cdot H \cdot \tau_{XY} * (cos^3(\theta) \cdot sin(\theta) - sin^3(\theta) \cdot cos(\theta)) \\ +(\frac{4 \cdot \tau_{xy} \cdot cos(\theta) \cdot sin^3(\theta) - 4 \cdot \tau_{xy} \cdot sin(\theta) \cdot cos^3(\theta)}{((\tau_{12})_{ult})^2}) \cdot (\tau_{xy}^2) \\ +\frac{4 \cdot cos(\theta)^3 * sin(\theta) \cdot \tau_{xy}}{(\sigma_1^T)_{ult} \cdot (\sigma_1^C)_{ult}} \\ +\frac{-4 \cdot sin(\theta)^3 \cdot cos(\theta) \cdot \tau_{xy}}{(\sigma_2^T)_{ult} \cdot (\sigma_2^C)_{ult}} \\ +(\frac{1}{(\sigma_1^T)_{ult}} - \frac{1}{(\sigma_1^C)_{ult}}) \cdot cos(\theta)^2 \\ +(\frac{1}{(\sigma_2^T)_{ult}} - \frac{1}{(\sigma_2^C)_{ult}}) \cdot sin(\theta)^2 \end{aligned} \tag{4.36}$$

$$\begin{aligned} g = -1 - (\tau_{xy}^2) \cdot (8 \cdot H \cdot cos^2(\theta) \cdot sin^2(\theta)) \\ +(\frac{cos^4(\theta) + sin^4(\theta) - 2 \cdot cos^2(\theta) \cdot sin^2(\theta)}{((\tau_{12})_{ult})^2}) \cdot (\tau_{xy}^2) \\ +\frac{4 \cdot \tau_{xy} \cdot cos(\theta)^2 \cdot sin(\theta)^2}{(\sigma_1^T)_{ult} \cdot (\sigma_1^C)_{ult}} \\ +\frac{4 \cdot \tau_{xy} \cdot cos(\theta)^2 \cdot sin(\theta)^2}{(\sigma_2^T)_{ult} \cdot (\sigma_2^C)_{ult}} \\ +(\frac{1}{(\sigma_1^T)_{ult}} - \frac{1}{(\sigma_1^C)_{ult}}) \cdot (2 \cdot cos(\theta) \cdot sin(\theta) \cdot \tau_{xy}) \\ -(\frac{1}{(\sigma_2^T)_{ult}} - \frac{1}{(\sigma_2^C)_{ult}}) \cdot (2 \cdot cos(\theta) \cdot sin(\theta) \cdot \tau_{xy}) \end{aligned} \tag{4.37}$$

Onde o valor H deve-se entender como:

$$\begin{aligned} H = (\frac{1}{2 \cdot (\sigma_{biaxial})^2}) \cdot (1 - \sigma_{biaxial} \cdot (\frac{1}{(\sigma_1^T)_{ult}} - \frac{1}{(\sigma_1^C)_{ult}} + \frac{1}{(\sigma_2^T)_{ult}} - \frac{1}{(\sigma_2^C)_{ult}}) \\ -(\sigma_{biaxial})^2 \cdot (\frac{1}{((\sigma_1)^T)_{ult}((\sigma_1)^C)_{ult}} + \frac{1}{((\sigma_2)^T)_{ult}((\sigma_2)^C)_{ult}})) \end{aligned} \tag{4.38}$$

Neste algoritmo é preciso calcular o descriminante para determinar o tipo de curva cônica que os valores das resistências do material determina.

4.1.2.1 ·

Desenho do algoritmo geométrico para os envelopes de Hoffman

A similitude entre o critério de Tsai-Wu e de Hoffman permite desenhar o algoritmo deste critério da mesma forma que no caso anterior, usando um valor H (Eq. 4.38) diferente:

$$H = -\frac{1}{2 \cdot (\sigma_1^T)_{ult} \cdot (\sigma_1^C)_{ult}} \tag{4.39}$$

Desenho do algoritmo geométrico para os envelopes de Hashin e Christensen

Estes dois critérios, ao igual que os critérios extremos, define-se como a união lógica de duas funções portanto o envelope de falha é a interseção lógica das áreas definidas por cada função.

É preciso obter os pontos de interseção das curvas representadas por cada função, como no caso dos algoritmos geométricos dos critérios extremos, ou avaliar ponto por ponto de cada curva sendo parte do envelope aqueles que satisfazem a união lógica que define o critério, sendo este último método, o mais conveniente dado que os pontos de interseção das curvas pode representar um cálculo de raiz cuja solução analítica não seja possível formular (ARFKEN; WEBER, 2005) o que aumentaria a complexidade do algoritmo ao ter que incorporar (BARROSO et al., 1987) uma etapa de cálculo numérico de raízes.

Ao expressar cada função destes critérios em termos das coordenadas locais observa-se que as mesmas definem a forma de uma curva cônica sendo possível codificar um segmento que calcule os coeficientes cônicos e outro que permita classifica-la como curva aberta ou curva fechada. Uma vez codificada cada curva, são avaliados os IF apenas daquelas curvas fechadas usando as técnicas de varredura em coordenadas polares construindo o conjunto de pontos que constituem o envelope apenas para aqueles com um IF de valor próximo à 1.

Desenho do algoritmo geométrico para os envelopes de Puck e Larc03

Para implementar os critérios de falha mais modernos, os quais na sua maioria estão desenhados para analisar cada modo de falha, é preciso desenhar algoritmos que possam diferenciar entre estes modos, ou seja é preciso definir uma etapa de controle.

Dado que estes critérios possuem raízes quadradas na sua formulação é difícil explicitar uma equação cônica como nos casos anteriores, sendo preciso usar o método de Graham-Scan (CORMEN, 2009) sobre um conjunto de pontos obtidos mediante o cálculo do IF de uma grade radial de pontos, no plano (σ_x, σ_y), construída com valores conveniente conforme as propriedades mecânicas de cada material.

Este método aproximado foi evitado para os critérios anteriores dado que a qualidade do envelope depende dos valores do espaçamento angular com que se define a grade, e o consumo de memoria RAM é claramente maior.

4.2 Desenho das interfaces

Dado que o sistema informático foi projetado para funcionar tanto na WEB como no sistema operacional Android, foi preciso desenhar interfaces totalmente orientadas ao ambiente WEB de trabalho e interfaces orientadas ao sistema operacional Android.

Interfaces WEB

A proposta inicial simplificada do Mech-Gcomp se afasta do tradicional esquema de "linha de comando" e "arquivos planos" para alimentar os dados no software educativo (GATLIN, 2009), e usa os elementos de entrada mais simples do HTML, adaptados ao framework Django por meio da criação de templates, na arquitetura MVT. Este framework possui uma estrutura de diretórios suficientemente ordenada para permitir o desenvolvimento constante de novos templates sem conflitar com os já programados, sempre que seja configurado corretamente o arquivo **urls.py** do Django. Este arquivo permite ocultar rascunhos de template não aprovados para produção, prática de programação comum (BENNETT, 2008) inclusive nas grandes companhias de software.

Uma parte significativa do código javascript (extensão ".js") do projeto macro foi substituído pelo mecanismo de passe de argumentos que o mesmo Django disponibiliza poupando assim um código que é usado apenas uma vez pelo usuário e que pode representar uma pesada carga para o servidor WEB que suporta o projeto, pois cada entrega de um arquivo ".js" por parte do servidor gera um uso dos recursos de memoria e da largura de banda do canal de transmissão. Por exemplo as funções que geram os pontos dos envelopes de falha recebem minimamente, como argumentos, o conteúdo da expressão regular processada pelo Django na chamada URL do módulo "EnvelopesDidaticos", codificada na sua forma mais simples, como:

```
url(r'^criterios/de/falha/
(?P<angulo>-?[0-9]\d*(\.\d+)?)/
(?P<lâminaURL>\d{2})/
(?P<criterio>\d{2})/
(?P<sigX>-?[0-9]\d*(\.\d+)?)/
(?P<sigY>-?[0-9]\d*(\.\d+)?)/
(?P<tauXY>-?[0-9]\d*(\.\d+)?)/
(?P<limpar>\d{1})/$',
```

```
views.criterios_de_falha_passing_url, name='criterios_de_falha'),
```

Estes argumentos são: (i) **"angulo"**: como um valor tipo flotante, (ii) **"laminaURL"**: sendo este um valor inteiro que indica o campo identificador no banco de dados MySQL, (iii) **"criterio"**: valor inteiro para identificar o critério de falha usado, (iv) Estado de esforços iniciais (**"sigX, sigY, tauXY"**), e (v) **"limpar"**: argumento para que seja limpado o espaço de representação dos envelopes de falha. E, conforme a função geométrica (vide Tabela 7), foram acrescentados estes argumentos para implementar corretamente cada envelope de falha, isto mediante o acréscimo das expressões regulares no arquivo urls.py.

Outra consideração do desenho das interfaces foi a simplicidade de programação impondo que as mesmas não sejam responsivas para redimensionamento, ou seja a interface não se adapta à tela do usuário o que pode parecer uma divergência com as novas tendencias tecnológicas (LIBBY; GUPTA; TALESRA, 2016) mas garante a simplicidade. Quando é preciso trabalhar com parâmetros adicionais (como no caso do envelope de Tsai Wu), o template chama uma função do tipo prompt do javascript para invocar, via AJAX, um URL especifico que coloca este parâmetro como variável global no arquivo python que implementa o envelope em especifico.

Tabela 7: Argumentos adicionais URL recebidos pelas funções geradoras de pontos dos envelopes de falha.

Função geometrica	**Argumentos adicionais**
gra_IF_Tsi_Wu	Parâmetro biaxial
gra_IF_Christessen	Parâmetros da fibra
gra_IF_Puck	Parâmetros da fibra
gra_IF_Larc03	Parâmetros da fibra

Com o intuito de aproveitar ao máximo a capacidade de framework para gerar saídas em JSON, foram configuradas expressões regulares "URLs JSON" para visualizar desde um navegador WEB as saídas de dados daquelas funções que geram um fluxo de informação para as interfaces. Todas as funções associadas às chamadas URL das interfaces estão reguladas pelo controle de usuário o que evita visualizações furtivas. Estas saídas JSON foram de grande utilidade no teste do funcionamento dos critérios e envelopes de falha.

Foi aproveitado um único template para todos os envelopes de falha. Este conceito de "template universal", permitiu representar de forma simultânea diversos envelopes de falha, uma função didática que o software comercial Helius não possui. Foi criado um histórico onde eram indicados com clareza, a modo de legenda, os tipos de envelope representados num único gráfico. Nesta versão inicial foi limitado a 6 o número de envelopes simultâneos que podem ser construídos. Nestas interfaces se apresentam os url para que o aluno possa transferir o arquivo MATLAB correspondente com os envelopes construídos durante a

sessão, assim como um conjunto de dados numéricos para o software GNUPLOT(JANERT, 2016).

Uma das funcionalidades do Django, não aproveitadas até agora no software Mech-Gcomp, é o repositório de arquivos HTML. Neste módulo foi, criada uma pasta para colocar textos explicativos do tipo aula que podem facilmente ser invocadas a partir do navegador para assim aproveitar o poder do HTML e poupar tempo porque os urls inseridos nestes textos didáticos conduzem de forma imediata para as funções de envelopes e critérios de falha, por meio do passe de argumentos via URL.

Este desenho, tendo como premissa o uso racional dos recursos e o uso constante de um modelo básico para representar os dados gerados pela camada numérica permitiu concluir rapidamente a etapa de codificação das interfaces sem que fosse preciso rever os algoritmos já desenhados, ou seja, as interfaces neste sistema atuam como simples elementos de apresentação e entrada de dados, seja esta entrada totalmente manual ou via URL.

As interfaces foram testadas e aprimoradas por meio da técnica de passe simples de argumentos de entradas, não foram feitos testes especiais de varredura de possíveis valores de entrada, pelo elevado grau de confiabilidade que o Django possui dentro da comunidade informática.

Outro tipo de interface WEB deste sistema informático são os espelhos os quais não estão integrados à framework nenhum e representam um conjunto de códigos em HTML, Bootstrap e javascript que executam de forma rápida e simples o mesmo cálculo numérico que foi codificado em python. Este espelhos representam a codificação dos algoritmos de forma menos estruturada e são tanto interfaces como códigos numéricos o que degrada sua qualidade como elemento didático dado que representa uma prática de programação indesejável do ponto de vista industrial(DAYLEY, 2008).

A função de cada espelho consiste em reproduzir num navegador WEB o mesmo envelope que foi calculado no aplicativo Android sem que o usuário tenha que introduzir manualmente os dados necessários, com este fim esta interface WEB recebe os parâmetros via URL no mesmo formato que foi usado para passar argumentos nas interfaces WEB anteriores. Esta repetição faz transparente para o usuário docente a construção de links no material didático.

Do ponto de vista computacional estes espelhos consomem um mínimo de poder de computo do servidor WEB dado que o javascript que executa o cálculo numérico é entregue ao usuário e executado no computador dele, o que parece ser uma vantagem no caso em que este sistema informático seja usado por uma turma numerosa.

Por ser apenas um espelho não foi usado um banco de dados sendo apenas usado como fonte de informação um arquivo com os dados mecânicos de cada lâmina em formato

JSON

Interfaces Android

Estas interfaces devem ser mais limitadas que no caso anterior dado que o sistema operacional Android possui severas limitações visuais quando comparado com os sistema operacionais Linux Ubuntu, além das limitações próprias de dispositivos mobiles com memoria RAM de apenas 1 Gigabyte e velocidade do relógio de processador de 500 MHz.

Com o intuito de não criar um desenho complexo foi usada como ferramentas de desenho e construção desta interface aquelas ferramentas integradas na IDE Android Studio 3.0 sendo usados aqueles elementos básicos listados na Tabela 8.

Tabela 8: Elementos visuais usados na interface Android.

Elemento Visual	**Comentários Técnicos**
TextView	Permite escrever uma mensagem de texto
ListView	Apresenta uma lista de elementos
SeekBar	Barra deslizante simples
EditText	Permite ingressar um texto qualquer
Button	Função de botão
ImageView	Expõe imagens na interface
RelativeLayout	Parâmetro biaxial
GraphView	Gráfico da biblioteca GraphView
NavigationView	Apresenta menu lateral de navegação
AppBarLayout	Apresenta barra superior

Pode-se afirmar que estes elementos visuais são nativos do Android, exceto o GraphView que foi usado por ser um elemento leve que não apresentou falhas durantes as numerosas provas desta biblioteca (GEHRING, 2017).

O ordenamento espacial entre estes elementos visuais foi projetado para garantir certo comportamento responsivo. Não foi usada programação de Fragments (MONTEIRO, 2014) para evitar codificações extensas e de pouca utilidade nesta primeira versão Android deste sistema informático.

Foi usado um número limitado de cores e figuras com o intuito de não usar muita memoria RAM assim como evitar que o tamanho da versão final do APK supera-se o valor inicialmente projetado de 5 MB.

5 TESTE DOS ALGORITMOS

Um teste de software exige que seja definido com significativo cuidado os ambientes em que serão executados os códigos, os quais foram projetados antes de se iniciar a etapa de codificação para que o produto da programação em python e MATLAB seja de fácil adaptação para estes ambientes. Assim, para cada critério e envelope de falha foi preparado um arquivo do tipo *.py, para ser colocado no seu respetivo ambiente de teste e facilitar a observação direta dos tempos de execução e as saídas numéricas.

5.1 Ambientes e testes iniciais para o módulo Django

Com o intuito de evitar interferências de qualquer tipo no teste do algoritmo foi seguida uma metodologia na preparação e execução das provas de funcionalidade, representada pelas etapas de:

(a) Isolamento do código e prova de funcionalidade:

O arquivo com o código python que implementa cada critério de falha foi colocado numa pasta única para poder ser executado diretamente a partir do terminal do sistema operacional linux Ubuntu. Assim, foi criada uma pasta para cada critério de falha codificado e outra para cada envelope de falha codificado. O fato de separar cada função computacional permite verificar a funcionalidade e apreciar qualitativamente os tempos de execução. Nesta etapa, foi apreciado que todas as funções computacionais admitiam os estados de esforços e argumentos experimentais como entrada e geravam uma saída de forma rápida, sem consumo excessivo de CPU ou de memória RAM.

(b) Provas isoladas de funcionalidade com sequência de dados aleatória:

A etapa anterior permite detectar eventuais erros de sintaxe e de codificação, mas não permite visualizar eventuais erros de lógica, sendo preciso usar a técnica de construção de mapas de cores para ver a qualidade lógica das funções construídas. O mapa de cores foi definido com apenas duas cores: a cor preta que representa um IF menor que um e a cor amarela que representa um IF maior de um. Este mapa depende dos valores de entrada, ou seja dos estados de esforços, e das propriedades da lâmina, motivo pelo qual foi gerada uma sequência aleatória de 1000 estados de esforços destrutivos e não destrutivos com ajuda da função **random** da biblioteca numérica de python **numpy**. Neste mapa de cores é possível definir como fronteira aquela região do plano que combina pontos amarelos e pretos de forma muito próxima, e nesta região deve existir o envelope de falha.

Foram construídos, usando apenas os códigos python de cada critério de falha, os mapas de cores para ângulos notáveis de 0°, 30°, 45°, 60°, e 90° para cada material

registrado no banco de dados, o que permitiu detectar erros mínimos próprios da atividade de digitação, e também apreciar uma fronteira de cores com formas muito similares aos envelopes de falha estudados.

(c) Provas de funcionalidade no framework com sequência de dados aleatória:

A etapa final de testes, destes algoritmos e códigos para o módulo Django, consistiu em adaptar os arquivos *.py para serem integrados ao Django, e posteriormente efetuar um teste de funcionalidade. Foi construído um módulo especial chamado de **"Área do programador"** com o intuito de facilitar a geração das sequências aleatórias de entrada e a construção dos mapas de cores.

Todas as funções codificadas foram testadas, sendo evidenciadas funcionalidade computacional e velocidades aceitáveis de execução, tanto de forma isolada como acopladas ao Django.

5.2 Ambientes e testes iniciais para o módulo Android

Os testes do códigos preparados em java aproveita a ampla variedade de métodos de teste consolidados ao redor da biblioteca Junit (TAHCHIEV; LEME; MASSOL, 2010) o que permite efetuar testes mais complexos e confiáveis. Nestes testes com Junit é preciso usar a IDE Android Studio onde é possível duplicar os arquivos java para serem integrados no ambiente de teste.

Os elementos visuais da interface Android foi testada com ajuda da biblioteca chamada "expresso" amplamente usada naqueles projetos Android onde os elementos visuais devem atingir uma elevada qualidade (MACHARLA, 2017). Estes testes dos elementos visuais tem com intuito observar o comportamentos dos mesmos durante o uso do módulo Android no emulador e detetar eventuais erros de representação ou ocultamento do elemento visual estudado.

Os testes dos códigos numéricos Android usou comunicação HTTP entre o emulador e um banco de dados MySQL (SUEHRING, 2011) local por meio de um fluxo controlado com um sistema de fabricação própria caraterizado por:

(a) Envio dos dados usando o método GET do HTTP, com ajuda da classe Java Android "AsyncTask", os quais são registrados com ajuda de um código feito em PHP (CONVERSE; PARK, 2003) para inserir no banco de dados MySQL as informações geradas pela rotina numérica java. Após registrar no sistema MySQL outro segmento de PHP, executado desde a linha de comando, gerava os gráficos com GNU plot (JANERT, 2016) para analisar o grau de coerência geométrica entre os resultados esperados e as saídas numéricas obtidas.

(b) Estudo das saídas numéricas dos códigos python, o que permite a revisão

mutua de ambos códigos java e python, comparando as duas sequências numéricas de saídas perante as mesmas entradas. Estas comparações foram feitas com ajuda do sistema de registro MySQL descrito no item anterior usado conjuntamente com as ferramentas de comparação próprias do Linux como os comandos **diff**, **awk**, etc.

5.3 Considerações teóricas do processo de verificação

5.3.1 Caraterísticas geométricas dos envelopes de máxima tensão e de máxima deformação

A geometria destes envelopes pode se resumir com várias proposições matemáticas que a definem.

(a) Proposição de igualdade em planos diferentes: Para uma lâmina única, o envelope de falha de máxima tensão representado no plano (σ_x,σ_y) possui a mesma forma que o envelope de máxima deformação, representado no plano (ϵ_x,ϵ_y), se o ângulo de rotação for um múltiplo de ($\pi/2$).

Usando as constantes α e β, é possível defini-as como $\alpha = 2$ e $\beta = 1$ na matriz de rotação T (Equação 5.1) que transforma esforços globais em locais.

$$T = \begin{bmatrix} cos^2(\theta) & sin^2(\theta) & \alpha \cdot cos(\theta) \cdot sin(\theta) \\ sin^2(\theta) & cos^2(\theta) & -\alpha \cdot cos(\theta) \cdot sin(\theta) \\ -\beta \cdot cos(\theta) \cdot sin(\theta) & \beta \cdot cos(\theta) \cdot sin(\theta) & cos^2(\theta) - sin^2(\theta) \end{bmatrix} \tag{5.1}$$

Utiliza-se uma matriz da forma $R \cdot T \cdot R^{-1}$, com $\alpha_{rtr} = 1$ e $\beta_{rtr} = 2$, a qual transforma deformações globais em locais, definida como:

$$R \cdot T \cdot R^{-1} = \begin{bmatrix} cos^2(\theta) & sin^2(\theta) & \alpha_{rtr} \cdot cos(\theta) \cdot sin(\theta) \\ sin^2(\theta) & cos^2(\theta) & -\alpha_{rtr} \cdot cos(\theta) \cdot sin(\theta) \\ -\beta_{rtr} \cdot cos(\theta) \cdot sin(\theta) & \beta_{rtr} \cdot cos(\theta) \cdot sin(\theta) & cos^2(\theta) - sin^2(\theta) \end{bmatrix} \tag{5.2}$$

Finalmente, vê-se que os únicos termos que simbolicamente diferem entre as matrizes T e a matriz $R \cdot T \cdot R^{-1}$ são os termos matriciais (1,3), (2,3), (3,1), (3,2) ou seja os quatro termos não diagonais, que se tornam iguais apenas quando estes fossem zero pelas raízes da função seno ou coseno nos ângulos múltiplos de $\pi/2$. Cabe salientar que o parâmetro τ_{xy} não altera a forma destes envelopes nesta condição de rotação pois se encontra presente nas desigualdades com um fator de ($cos^2(\theta) - sin^2(\theta)$), ou seja, o τ_{xy} não afeta a construção do envelope de falha neste caso.

(b) Proposição de 4 vértices para envelopes construídos com ângulos múltiplos de $(\pi/2)$**:**

Todo envelope de falha, de uma lâmina única, construído com o critério de máxima tensão ou deformação possui apenas 4 vértices para um material estudado no regime elástico se esta estiver rotacionada num ângulo múltiplo inteiro de $(\pi/2)$.

Para demostrar esta proposição, considera-se:

Axioma do ponto de quietude absoluta: No regime elástico e satisfeita a lei de Hooke, define-se uma condição de não deformação da lâmina quando não for aplicado esforço nenhum, o que define uma região no plano de esforço que deve conter este ponto de quietude absoluta (0,0).

Axioma da linha reta: Cada ponto na área contida pelo envelope de falha deve satisfazer todas as 6 desigualdades (no caso de máxima tensão, Eq. 2.10, e no caso de máxima deformação, Eq. 2.16). Quando estas forem tratadas como igualdades, representam uma linha reta infinita no espaço de esforços, ou de deformações respectivamente, que ao ser projetada entre os diversos planos de esforços também é infinita.

Axioma da faixa definida por desigualdade dupla: As linhas retas associadas com duas desigualdades que possuem as mesmas variáveis, com os mesmos coeficientes, são paralelas e ambas desigualdades são satisfeitas formando uma "faixa" de separação constante e de área infinita no plano.

Axioma das faixas ortogonais: Quando a lâmina estiver rotacionada num ângulo múltiplo de $(\pi/2)$, para um único plano de esforços existirá apenas duas faixas mutuamente ortogonais que tem uma área comum não nula. Nas matrizes de rotação (Eq. 2.5, Eq. 5.2) este ângulo de rotação gera um vetor [0 0 1] que define apenas uma dupla desigualdade e gera uma faixa $-(\tau_{12})_{ult} < \tau_{xy} < (\tau_{12})_{ult}$ que ao ser respeitada garante a existência do envelope de falha.

Fica em evidência que ao ser representados num plano de esforços qualquer, estas faixas mutuamente ortogonais geram um envelope de falha com apenas 4 vértices, mesmo se o envelope for construído inicialmente no plano de deformações porque a transformação entre ambos os planos é unitária (ARFKEN; WEBER, 2005).

Para uma lâmina rotacionada num ângulo diferente de $(\pi/2)$, pode-se aplicar esta proposição quando a faixa definida pela desigualdade do parâmetro τ_{xy} contenha toda a área comum definida pelas faixas de esforços representadas, como definido na proposição à seguir.

(c) Proposição de 4 vértices para envelopes construídos com ângulos não múltiplos de $(\pi/2)$**:** Para uma lâmina única um envelope de falha, representado no plano de esforços σ_x e σ_y , usando o critério de máxima tensão ou deformação, contém apenas

quatro vértices quando nenhuma das igualdades:

$$(-\tau_{12})_{ult} = -\beta \cdot cos(\theta) \cdot sin(\theta) \cdot \sigma_x + \beta \cdot cos(\theta) \cdot sin(\theta) \cdot \sigma_y + (cos(\theta)^2 - sin(\theta)^2) \cdot \tau_{xy}$$

e

$$-\beta \cdot cos(\theta) \cdot sin(\theta) \cdot \sigma_x + \beta \cdot cos(\theta) \cdot sin(\theta) \cdot \sigma_y + (cos(\theta)^2 - sin(\theta)^2) \cdot \tau_{xy} = (\tau_{12})_{ult}$$

cortarem de forma secante a região comum definida pelas desigualdades:

$$-(\sigma_1)^c_{ult} < cos(\theta)^2 \cdot \sigma_x + sin(\theta)^2 \cdot \sigma_y + (\alpha \cdot cos(\theta) \cdot sin(\theta)) \cdot \tau_{xy} < (\sigma_1)^T_{ult}$$

e

$$-(\sigma_2)^c_{ult} < sin(\theta)^2 \cdot \sigma_x + cos(\theta)^2 \cdot \sigma_y - (\alpha \cdot cos(\theta) \cdot sin(\theta)) \cdot \tau_{xy} < (\sigma_2)^T_{ult}$$

Onde os valores de (β,α) podem ser apenas de (1,2) ou (2,1).

Uma forma de demonstrar esta proposição é observando que a igualdade do esforço cortante está dominada apenas pelo valor de $(\tau_{12})_{ult}$ e se este valor aumentar então esta "faixa" aumenta sua largura (separação entre suas retas limitantes) o que evita qualquer corte secante com a área comum que não dependa deste valor extremo $(\tau_{12})_{ult}$.

(d) Proposição de numero máximo de vértices:

Um envelope de falha, do tipo máxima tensão ou máxima deformação, possui no máximo 6 vértices.

Com a ajuda das proposições anteriores sabe-se que é possível construir envelopes de falha de apenas 4 vértices usando uma rotação da lâmina em ângulos que sejam múltiplos de $(\pi/2)$, ou respeitando a condição indicada na **"proposição de 4 vértices para envelopes construídos com ângulos não múltiplos de** $(\pi/2)$**"**. Usando novamente o plano de esforços σ_x e σ_y vê-se que as três faixas infinitas, definidas pelas desigualdades do critério, se interceptam gerando uma área comum com no mínimo 4 vértices, naquelas condições de rotação, e se esta área comum inicial for cortada de forma secante por uma reta é gerada um área comum delimitada por 5 vértices.

As retas geradas pelas igualdades do critério podem cortar está área comum inicial de 4 vértices, sendo possível ter apenas 2 retas paralelas correspondentes àquela desigualdade do critério de falha com o termo $(cos(\theta)^2 - sin(\theta)^2)$. Se uma destas retas adicionais divide por corte secante a área inicial, adiciona-se um vértice, ao envelope de falha, quando uma das áreas divididas contem apenas um vértice. Logo, conclui-se que se estas retas adicionais geram cortes secantes deste tipo, eliminando uma parte daquela área comum que contém apenas um vértice, gera-se um envelope de falha de no máximo 6 vértices.

5.3.2 Considerações geométricas sobre os critérios Tsai-Hill, Azzi-Tsai e Tsai-Wu

Sabendo que a forma do envelope no plano de esforços globais (σ_x,σ_y) para os critérios Tsai-Hill, Azi-Tsai, Tsai-Wu está relacionado com uma curva cônica do tipo elipse, as caraterísticas geométricas de interesse estão definidas a seguir:

(a) Envelope com inclinação invariante perante rotação da lâmina para materiais isotrópicos No caso do material isotrópico, ou transversalmente ortotrópico com $(\sigma_1)^T_{ult} = (\sigma_2)^T_{ult}$, tem-se os coeficientes elípticos "a" e "c" (Eq. 4.16) se igualam o que gera um ângulo único para qualquer valor de θ de rotação angular da lâmina.

(b) Envelope com inclinação variante perante rotação da lâmina para materiais não isotrópicos Quando $(\sigma_1)^T_{ult} \neq (\sigma_2)^T_{ult}$ os coeficientes elípticos (da forma implícita $ax^2 + 2bxy + cy^2 + 2dx + 2fy + g = 0$) "a" e "c" são diferentes gerando como envelope de falha uma elipse onde se pode definir a inclinação reformulando a Equação 2.24:

$$\theta_{elipse} = \frac{\pi}{2} + \frac{cot^-1(numeradorAngular)}{2}$$
$$numeradorAngular = \frac{a-c}{2 \cdot b} \tag{5.3}$$

Que muda quando θ da lâmina varia no intervalo de 0 até $\frac{\pi}{2}$ onde temos os valores extremos:

Para $\theta = 0$

$$\theta_{elipse} = \frac{\pi}{2} + \frac{cot^-1(\frac{t1^2 - t2^2}{2 \cdot t2^2})}{2} \tag{5.4}$$

Para $\theta = \frac{\pi}{2}$

$$\theta_{elipse} = \frac{\pi}{2} + \frac{cot^-1(\frac{t2^2 - t1^2}{2 \cdot (t2^2)})}{2} \tag{5.5}$$

Onde se definem: $t1 = \frac{1}{(\sigma_1)^T_{ult}}$, $t2 = \frac{1}{(\sigma_2)^T_{ult}}$, $t12 = \frac{1}{(\tau_{12})_{ult}}$.

(c) Centrado na origem quando o τ_{xy} for nulo:

Usando a formulação usada para calcular o centro da elipse (Eq. 5.6 e Eq. 5.7)

$$x_0 = \frac{c \cdot d - b \cdot f}{b^2 - a \cdot c} \tag{5.6}$$

$$y_0 = \frac{a \cdot f - b \cdot d}{b^2 - a \cdot c} \tag{5.7}$$

vê-se que a elipse fica centrada na origem quando d e f forem nulos, como nos casos de Tsai-Hill e Azzi-Tsai na condição de $\tau_{xy} = 0$. Já no caso do envelope de Tsai-Wu este centrado na origem acontece apenas quando o $\tau_{xy} = 0$ e o material for isotrópico.

Observa-se que o desenho dos algoritmos não foi cíclico nem foi usada técnica de "tentativa e erro" porque os testes projetados e usados permitiram criar cada algoritmo usando sólidas bases teóricas. Outro aproveitamento dado a estas proposições geométricas, foi para ser usado como conteúdo na preparação de várias aulas didáticas colocadas na pasta Django configurada como repositório de arquivos HTML.

6 RESULTADOS

Este trabalho combinou fundamentos teóricos da mecânica de lâminas, fundamentos de programação numérica tanto em python como em javascript e java, assim como fundamentos pedagógicos da informática educativa. Este caráter multidisciplinar obriga que os resultados sejam apresentados de forma separada com o intuito de aprofundar ordenadamente a análise.

6.1 Resultados de Informática

Os resultados de informática deste mestrado se ordenam cronologicamente como: (i) algoritmos, (ii) códigos MATLAB, python e java, (iii) interfaces de usuário. O módulo python, possui dois subsistemas. Um projetado como objetivo deste trabalho de mestrado e o outro chamado de "módulo do programador" que foi produto da sistematização dos testes sob a óptica de princípios teóricos amadurecidos e direcionados na área de geometria analítica. O módulo Java/Android representa uma versão mais complexa do módulo python preservando os mesmos algoritmos numéricos usados para calcular os critérios e construir os envelopes.

6.1.1 Algoritmos

A primeira etapa de codificação foi projetada com o intuito de transformar o corpo teórico em elementos de informática pertinentes, tais como algoritmos básicos e estrutura de dados. Foram desenhados algoritmos com a máxima simplicidade sem abordar o paradigma de processamento paralelo, nem os detalhes relacionados com a administração de memória, o que permitiu uma rápida codificação simplificada em MATLAB e simultaneamente em python. A versão Java/Android destes algoritmos demandou de um maior esforço dado que não foi usada nenhuma biblioteca para o cálculo matricial, sendo preciso desenvolver e testar rigorosamente funções de inversão de matrizes, cálculo de determinantes, etc.

6.1.2 Códigos em Python

Os algoritmos numéricos e geométricos foram implementados em python e incorporados na camada VIEW do framework Django, ou seja, no arquivo **views.py** do módulo implementado. Estruturalmente um módulo Django está representado por uma única pasta, que pode conter outras pastas o que permite ordenar os códigos. Inicialmente foi criada a pasta **envelopesDidaticos**, que contém os arquivos python a seguir:

envelope_maxima_tensao.py: Diferentemente da implementação em MATLAB do algoritmo, em python foram definidas funções com o intuito de manter o desenho modular. Estas funções tem como saída, por meio do retorno de uma estrutura de dados tipo lista, um conjunto de pontos que serão representandos pela biblioteca FLOT no template correspondente. Esta função é chamada por outra função dentro do arquivo **views.py** e possui uma etapa prévia de escolha conforme o ângulo de rotação da lâmina estudada, já que no caso de se tratar de um ângulo múltiplo de $\frac{\pi}{2}$ o algoritmo poupa as eliminações gaussianas e calcula diretamente os vértices do envelope de falha se existir. Em todo momento, é usado τ_{xy} como parâmetro e as propriedades da lâmina são consultadas dentro da função conforme o identificador (**id**) de cada lâmina dentro do banco de dados usando o modelo definido pelo Django desde o inicio do projeto macro.

envelope_maxima_deformacao.py: Este arquivo é muito similar ao arquivo anterior, incorporando apenas uma etapa final de transformação de deformações (ϵ_x,ϵ_y) em tensões para manter o padrão do projeto e representar em todo momento no plano (σ_x,σ_y). A codificação separada destes envelopes de falha se justifica na necessidade de manutenção e para garantir uma adequada apreciação qualitativa da etapa de testes. A codificação destes dois algoritmos se diferencia dos seus equivalentes MATLAB visto que a eliminação gaussiana é codificada diretamente com ajuda da biblioteca numérica **numpy** que permitiu a programação das operações de multiplicação e inversão de matrizes. Quando não é possível construir o envelope de falha, neste tipo de envelope e no anterior, a saída da função python é uma lista vazia.

envelope_tsai_hill.py: É o mais simples dentre os arquivos python que implementam os envelopes de falha tipo quadráticos dado que o algoritmo representa apenas o código numérico de um gráfico da curva cônica do tipo elipse, e um filtro para determinar os efeitos do valor do parâmetro τ_{xy}.

envelope_azzi_tsai.py: Arquivo de maior tamanho que o "envelope_tsai_hill.py" dado que a etapa de escolha do valor de resistência mecânica representa uma parte significativa deste algoritmo.

envelope_tsai_wu.py: Muito similar ao arquivo "envelope_tsai_hill.py" dado que o ingresso do parâmetro biaxial não altera o estilo de codificação usado.

envelope_hoffman.py: Muito similar ao arquivo "envelope_tsai_wu.py" dado que representa um caso especial deste critério.

envelope_hashin.py: Apesar que este critério considera um modo de falha para fibra e outro para a matriz, foi preparado um arquivo python único com o intuito de não criar uma quantidade exagerada de arquivos.

envelope_cristensen.py: Apesar que este critério se define por união lógica, tal como os critérios de "máxima tensão" e "máxima deformação" o tamanho do código python foi

mais reduzido que estes.

envelope_puck.py: Dado que no momento de ser invocada a função python do envelope Puck esta recebe todos os parâmetros relacionados, sejam essenciais ou não para o cálculo. Foi gerado um código python simples que não precisa de interações, condicionadas, com a camada visual do framework.

envelope_larc03.py: Arquivo de dimensão similar ao "envelope_puck.py" dado que contém um codigo escrito considerando as mesmas restrições arquitetônicas do caso anterior, ou seja permitindo como única interação com a camada visual o passe de argumentos no momento de invocar a função python que o arquivo contém e recebendo um valor de retorno.

Os arquivos que implementam a função computacional do critério de falha também foram colocados de forma separada na pasta de **criteriosDidaticos**, sendo preciso detalhar:

calc_IF_maxima_tensao.py: É menor que o arquivo "envelope_maxima_tensao.py" dado que é preciso apenas codificar como saída o valor do IF.

calc_IF_maxima_deformacao.py: mais extenso que o arquivo anterior porque contém uma codificação similar e uma etapa adicional de transformação de coordenadas.

calc_IF_tsai_hill.py: É um arquivo extenso dado foi considerada a formulação cônica do critério em coordenadas de esforços globais.

calc_IF_azzi_tsai.py: Similar ao caso anterior tendo como elemento adicional um simples seletor da resistência conforme o sinal do esforço local.

calc_IF_tsai_wu.py: Similar ao caso anterior e similar ao arquivo do critério de Hoffman.

calc_IF_hoffman.py: Menor, em numérico de linhas, ao caso anterior dado que não implementou etapa de análise do parâmetro biaxial $\sigma_{biaxial}$.

calc_IF_hashin.py: É o menor dos arquivos python contidos nesta pasta.

calc_IF_cristessen.py: É o segundo menor dos arquivos python.

calc_IF_puck.py: O número de linhas deste arquivo é considerável dado que o critério considera vários parâmetros que são calculados dentro da função python.

calc_IF_larc03.py: um arquivo de extensão considerável dado que o critério de Puck está formulado considerando vários parâmetros experimentais. O fato que a função python aqui implementada recebe todos os parâmetros no momento de ser chamada, evita que seja necessário adicionar código para controlar eventuais interações com a camada visual.

Todos estes arquivos python são lidos pelo subsistema "módulo do programador" e integrados mediante a técnica wrap, o que permite uma rápida reprogramação dos critérios

e envelopes de falha.

6.1.3 Códigos em Java/Android

Outra versão dos algoritmos desenhados encontra-se contidas nos códigos escritos em java com ajuda da IDE Android Studio. Nesta IDE os arquivos *.java podem ser tratados de forma integrada (Fig. 10) dado que a ferramenta Gradle(VARANASI, 2015) facilita a compilação e empacotamento das bibliotecas usadas.

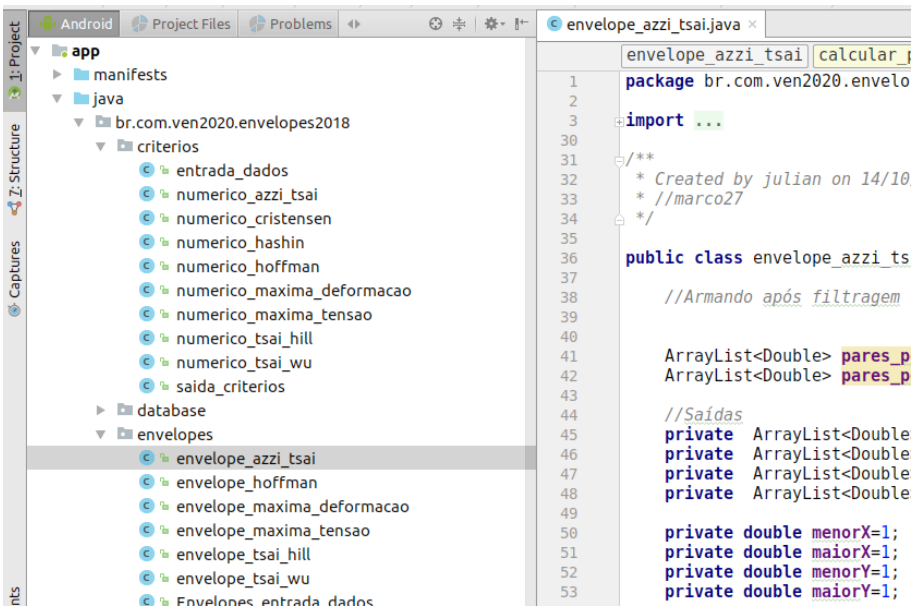

Figura 10: Arquivos Java na IDE Android Studio.(Fonte: Autor)

Os código java que contém apenas os algoritmos numéricos, tanto dos critérios como dos envelopes, se definem como classes com o intuito de poder ser instanciados como objetos naquelas outras classes que arquitetam esta aplicação. Sobre os arquivos que não possui os algoritmos numéricos deve-se analisar:

entrada_dados.java: Implementado como uma extensão da classe "AppCompatActivity", e permite escolher o critério de falha que será usado pelo usuário. Nesta classe é verificada a qualidade dos dados de entrada e, se fossem válidos, é permitido o cálculo do critério de falha para finalmente, usando a função API "startActivity", invocar a classe **saida_criterios** para poder apresentar os índices de falha obtidos. Desde esta classe se instancia aquela classe que implementa o algoritmos do critérios de falha (Tabela 9) conforme o IF que se deseja calcular. No caso daqueles critérios que precisam de parâmetros experimentais foram criados quadros de diálogo os quais são invocados desde esta classe conforme as iterações do usuário.

Envelopes_entrada_dados.java: Como uma extensão da classe "AppCompatActivity" que permite: (i) escolher o envelope de falha (Tabela 10) que será usado pelo usuário, (ii)

Tabela 9: Classes que implementam os critérios de falha.

Classe java	**Comentário**
numerico_maxima_tensao	Sem parâmetro experimental de entrada
numerico_maxima_deformacao	Similar à anterior, com conversão das deformações
numerico_tsai_hill	Sem parâmetro experimental de entrada
numerico_azzi_tsai	Com seletor conforme tipo e esforços
numerico_tsai_wu	Com parâmetro biaxial de entrada $\sigma_{biaxial}$
numerico_hoffman	Similar ao anterior com $\sigma_{biaxial}$ nulo
numerico_hashin	Com seletor do IF conforme o esforço local
numerico_cristessen	União lógica com saída vetorial
numerico_larc03	O construtor recebe todos parâmetros de entrada
numerico_puck	Recebe todos os parâmetros de entrada possíveis

Tabela 10: Classes que implementam os envelopes de cada critérios de falha.

Classe java	**Comentário**
envelope_maxima_tensao	Calcula apenas os vértices do envelope
envelope_maxima_deformacao	Similar à anterior, com conversão geométrica
envelope_tsai_hill	Desenha uma elipse
envelope_azzi_tsai	Seleciona as resistências conforme os esforços
envelope_tsai_wu	Considera o parâmetro biaxial de entrada $\sigma_{biaxial}$
envelope_hoffman	Similar ao anterior com $\sigma_{biaxial}$ nulo
envelope_hashin	Solicita o τ_{23} como entrada
envelope_cristessen	Representa a união lógica de duas funções
envelope_larc03	Com transformação geométrica
envelope_puck	Solicita vários parâmetros

consultar no banco de dados local SQLite sobre aquele ponto que será representado de forma conjunta com o envelope de falha calculado, (iii) construir o URL que direcionado ao espelho onde é possível refazer o cálculo, feito pelo usuário no aplicativo módulo Android, usando um servidor WEB. Este URL é colocado na interface de usuário como um hiperlink HTML para redirecionar ao navegador no momento de ser tocado na tela Android.

Outras classes necessárias para arquitetar o software Android, e que foram codificadas da forma:

MainActivity.java: Declarada no manifesto do aplicativo, arquivo "AndroidManifest.xml", como a primeira classe invocada pelo módulo Android quando for iniciado pelo usuário. Contém vários métodos essenciais para cumprir a função educativa, tais como:

listar_laminas: Usado para apresentar uma lista das lâminas registradas no banco de dados estático SQLITE que contem as propriedades mecânicas de cada material listado. Este método controla um elemento visual do tipo ListView (Tabela 8) recebendo os eventos e ativando uma janela para oferecer várias funcionalidades ao usuário.

janela: Método usado para oferecer ao usuários as três operações básicas disponibilizadas

para cada lâmina: (i) "Propriedades da lâmina" para visualizar as propriedades de cada lâmina, (ii) "Envelopes de Falha" para escolher o tipo de envelope que deseja construir, (iii) "Critérios de Falha" para escolher o critério de falha com o qual calcular o IF.

lembrar_ultimo_grafico: Método usado para colocar na tela inicial do aplicativo Android aquele último gráfico estudado pelo usuário.

DatabaseHelper.java: Estende a classe "SQLiteOpenHelper" e permite usar o banco de dados estático SQLITE, assim denominado pois é usado apenas para obter as propriedades mecânicas das lâminas. Nesta classe o método "prepareDatabase" permite atualizar este banco de dados no caso de uma atualização do aplicativo sem que seja preciso intervenção do usuário. Foi criado o método "obter nomes_laminas" para obter apenas uma estrutura de dados do tipo List<String> com os nomes das lâminas, informação que será usada para preparar o ListView que exibe estes nomes. Já o método "obter_propriedades_laminas" permite obter uma lista similar contendo todas as propriedades de uma lâmina especificada no parâmetro de entrada "nome_lamina". Uma vantagem de implementar este método de consulta por nome da lâmina ao invés de usar algum numero identificador é para facilitar que futuros desenvolvedores possam detectar eventuais erros num banco de dados duplicado.

db_historico_estados_esforcos: Classe que permite registrar os cálculos efetuados pelo usuário assim como recuperar os esforços já usados. No método onCreate definiu-se a estrutura da única tabela do banco de dados, e no caso de atualizar o sistema esta será apagada com o método onUpgrade. Foi usada uma estrutura de dados do tipo ArrayList<String> para recuperar os últimos 10 registros usando o método "obter historico_estados_esforcos". Dois métodos que permitem poupar memoria, o que é necessário quando se projeta um módulo Android, são "obter_estados_esforcos_especifico" e "procurar_lamina_diretamente", o primeiro destes retorna um ArrayList<String> colocado numa posição em especifico da tabela "historico_estados" e o último permite recuperar apenas o nome da lâmina de um registro em específico.

historico: Classe que extende "SQLiteOpenHelper" e implementa um banco de dados com uma única tabela onde se registram tanto os envelopes como os critérios de falha usados. Esta tabela constitui uma cópia parcial da tabela que registra os estados de esforços usados, dado que registra apenas a lâmina e tipo de critério ou envelope usado.

6.1.4 Códigos de integração conforme os moldes do framework Django

As funções python codificadas não podem interagir diretamente com a biblioteca FLOT, ou seja a camada VIEW do Django não gera diretamente o gráfico do envelope de falha. Por este motivo a saída dos algoritmos, programados em python, é passada para uma variável global no contexto do arquivo **views.py** com o intuito de permitir uma fácil

visualização no formato JSON com ajuda de uma função auxiliar que apresenta esta saída quando for consultada desde o navegador WEB. É assim como aparecem os códigos de integração para permitir que as saídas das funções python, tanto dos critérios como dos envelopes, possam ser visualizadas pelo usuário.

O meio criado para trocar informação entre o módulo MIPI implementado, e o usuário foram as interfaces caraterizadas por:

(a) Estar construída na linguagem de programação HTML, e incorporar fragmentos de código de javascript com o intuito de verificar os dados ingressados manualmente, e prover a configuração do sistema gráfico FLOT para representar em cores e formas corretas os envelopes construídos. Foram colocadas todas as funções javascript juntas ao código da interface, pela facilidade de integração que esta linguagem tem com o HTML o que permite fazer mudanças visuais sem alterar o núcleo python do MIPI. Como as funções python foram desenvolvidas com máxima simplicidade, as verificações numéricas dos valores ingressados no módulo foram desenvolvidas em javascript e limitadas para estar presentes apenas nas interface, sem que esteja presente verificação de tipo nas funções python.

(b) Na computação comercial focada na internet, é obrigatório usar algum formato para melhorar a qualidade visual dos conteúdos web, neste caso esta sendo usando o estilo Bootstrap versão 2.3.2 por meio da incorporação das folhas de estilo CSS na interface. A falta destas diminuiria a qualidade gráfica simplificando a interface.

(c) Ingresso Controlado. Aproveitando uma propriedade do framework Django foram desenvolvidas interfaces de ingresso controlado.

(d) Não responsivo para redimensionamento da tela do usuário. Esta não foi uma qualidade desenvolvida no projeto macro e portanto não foi implementada neste módulo. O motivo foi de manter a menor quantidade de código não HTML possível nas interfaces, facilitando assim a construção contínua dos fragmentos de javascript.

Estas propriedades podem ser apreciadas em cada uma das interfaces desenvolvidas. Assim por exemplo a interface inicial do módulo (Fig. 11) foi desenhada para ser usada apenas se o usuário indicar corretamente os seus dados de acesso.

Figura 11: Interface de usuário para construção manual do estado de esforços e cálculo do índice de falha (IF). (Fonte: Autor)

Nela podem se ingressar manualmente as informações necessárias para calcular um índice de falha, tais como:

(a) Material da lâmina: Pode ser escolher entre os 27 tipos de materiais, cadastrados num banco de dados disponível no módulo matriz do Mech-Gcomp, os quais correspondem às propriedades dos principais materiais usadas pelo software Helius e de outras bases de uso livre acadêmico ou comercial. Também é possível usar lâminas com propriedades definidas pelo próprio usuário dado que o projeto macro incorporou a funcionalidade de registro há 2 anos.

(b) Ângulo de rotação da lâmina: Foi inserido um fragmento javascript de verificação para evitar que o usuário inicialize um calculo do IF sem ter indicado o ângulo da lâmina.

(c) Estado mecânico inicial da lâmina: Seja este um estado de tensões ou deformações. Esta interface aceita apenas estados mecânicos inicias puros, ou seja apenas com deformações inicias ou esforços iniciais, sem permitir mistura entre os diversos tipos de condições iniciais.

(d) Escolha do critério de falha: Com ajuda de um elemento de entrada padrão HTML, do tipo botão seletor de escolha única, o usuário pode indicar qual critério de falha usar no cálculo.

Outra interface (Fig. 12) presente no "módulo de programador", especialmente desenhada para um usuário mais profissional, é a chamada "interface de testes massivos" composta de uma função de geração de arquivos de testes e um analisador gráfico do arquivo de esforços, caraterizada por:

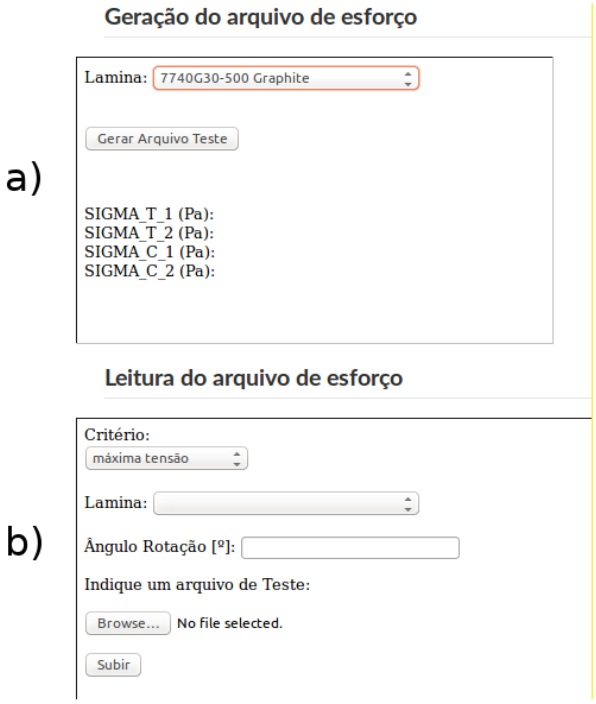

Figura 12: (a) Interface para gerar um arquivo teste com esforços σ_x, σ_y, τ_{xy}. (b) Interface para teste de um arquivo de esforços usando um critério de falha e um material especificado.(Fonte: Autor)

(a) Sistema gerador estados aleatórios de esforços:

No decorrer da etapa de testes dos algoritmos, foi usado um sistema gerador de estados aleatórios de esforços (σ_x, σ_y, τ_{xy}) usando o pacote random da biblioteca numpy (BRESSERT, 2012) de python. Visto que os testes se beneficiaram do caráter branco (ARFKEN; WEBER, 2005) dos estados aleatórios gerados, e por ser uma função de fácil implementação no Django, foi desenvolvido este sistema onde é gerado um arquivo no formato csv (dados separados por vírgula) com 100 estados aleatórios conforme os esforços máximos suportados por um material especifico. Antes de gerar este arquivo, é preciso considerar todos os esforços máximos do material e escolher entre eles os máximos valores absolutos dentre os esforços de tração e compressão em cada direção local da lâmina para indicar ao gerador de estados aleatórios quais são os limites de varredura. Foi definida uma varredura numérica nos 4 quadrantes, do plano (σ_x, σ_y), para tentar cobrir o maior numero de casos possíveis onde os esforços geram um IF menor e maior de 1.

Na Figura (13) vê-se o link url (texto em cor azul conforme a definição da linguagem HTML) que disponibiliza o arquivo de saída e os valores extremos do material considerados na geração aleatória. Nesta figura, vê-se o nome do arquivo "Teste_7740G30500 Graphite paraTodosCriterios.txt" respeitando o formato de nomeação onde o prefixo "Teste_" é seguido do nome do material no banco de dados Mech-Gcomp e do texto "_paraTodosCriterios.txt" por ser um arquivo plano do tipo TXT que pode ser estudado com qualquer um dos

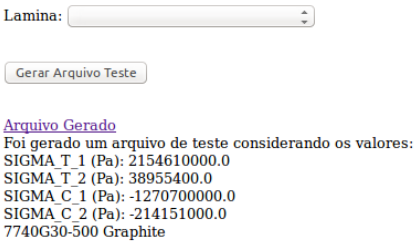

Figura 13: Detalhe da interface geradora do arquivo "Teste 7740G30500 Graphite paraTodosCriterios.txt" de estados aleatorios (σ_x, σ_y, τ_{xy}). (Fonte: Autor)

Figura 14: Interface de carregamento do arquivo de estados de esforços. (Fonte: Autor)

critérios implementados.

(b) Analisador gráfico:

Na parte inferior da "interface de testes aleátorios" (Fig. 12), há um sistema que permite introduzir diferentes estados de esforços para serem analisados com um critério e lâmina específicos. Uma das utilidades desta função é sua representação gráfica conjunta (Fig. 15) onde cada estado (ponto no gráfico) é colorido conforme o valor do IF que atinge no critério de falha e no material simulado. Nesta versão da ferramenta, representa-se como pretos aqueles pontos que não superam o valor de 1 (mecanicamente seguros conforme o critério) e como amarelos aqueles que superam este valor. Assim uma simples e rápida inspeção visual do gráfico colorido permite recomendar um determinado material para suportar os esforços contidos no arquivo de entrada.

Esta interface está composta de duas partes, a primeira (Fig. 14) que permite carregar um arquivo a partir do computador do usuário e indicar o critério de falha que determina o mapa de cores, o material e o ângulo de rotação da lâmina. E uma parte final (Fig. 15) contém o mapa de cores construído com ajuda da biblioteca FLOT.

Este módulo auxiliar, subproduto do projeto inicialmente formulado, constitui uma ótima ferramenta para testar a implementação de futuros critérios de falha no módulo Django deste sistema informático dado que a geração de mapas de cores constitui uma técnica eficiente para detectar eventuais erros de programação.

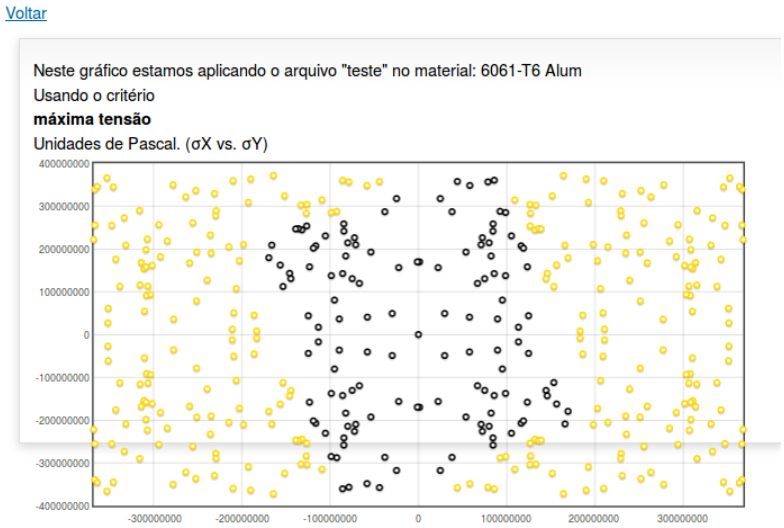

Figura 15: Mapa de cores para o critério de máxima tensão usando como entrada o arquivo de dados aleatórios para o Alumínio 6061-T6 rotacionado 30°. (Fonte: Autor)

Finalmente, as interfaces de apresentação dos envelopes de falha compreendem três versões. Uma versão usada para a representação individual de cada envelope desenhado de forma não responsiva. Outra versão, similar ao caso anterior, para representar simultaneamente até 6 envelopes de falha sendo definido este máximo para evitar sobrecargas no uso da memória do servidor. E uma terceira versão que usou biblioteca gráfica Chartist (KUNZ, 2017) com a função responsiva plena mas de estética limitada. Estas interfaces se caraterizam por:

a) Indicação do gráfico. Com o intuito de expressar com máxima clareza todos e cada um dos conteúdos gerados por este módulo Django, é indicada a natureza do gráfico gerado mencionando o critério de falha usado, ângulo de rotação da lâmina, e material.

b) Disponibilização do código MATLAB equivalente. Na parte inferior desta interface Django vê-se um link url, da forma de botão conforme o estilo Bootstrap, para transferir o arquivo de extensão *.m (MATLAB versão 7.11 do ano 2010), que permite refazer esse mesmo gráfico na aula prática de qualquer disciplina relacionada com "mecânica de compósitos". Disponibilizar estes códigos para o usuário ratifica o caráter totalmente aberto do sistema informático desenvolvido neste mestrado, e transforma a tarefa de geração de um envelope de falha no MATLAB em um simples ato de transferir um arquivo gerado por este módulo para execução no computador do usuário.

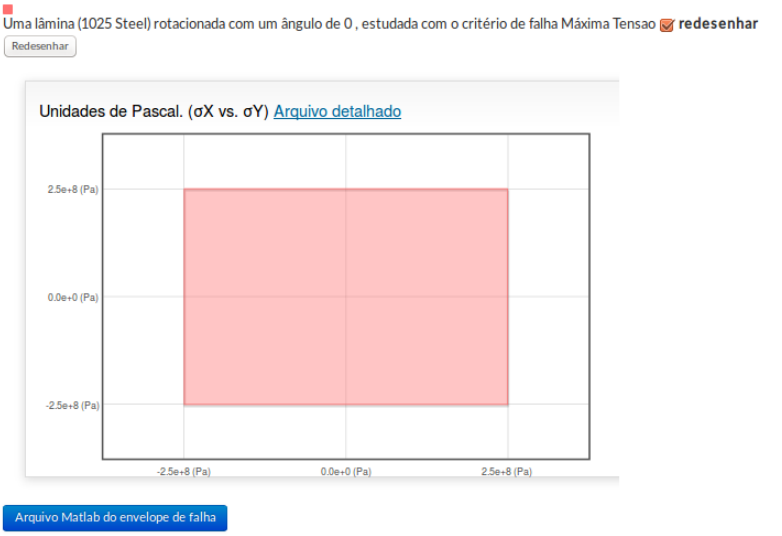

Figura 16: Interface de múltipla representação dos envelope de falha. (Fonte: Autor)

6.1.5 Espelhos WEB associados com o módulo Android

Outro resultado inesperado, do tipo subproduto, são os "espelhos WEB" os quais permitem que o usuário Android possa ver refeito seu cálculo do IF ou envelope num servidor WEB. Estes espelhos são apenas um conjuntos de códigos HTML e javascript que não estão arquitetados no modelo MVT sendo apenas uma combinação extensa de fragmentos de código. Apesar da sua simplicidade arquitetônica reproduz muito bem os valores numéricos que o módulo Django gera, assim como os gráficos gerados tanto no Django como no módulo Android.

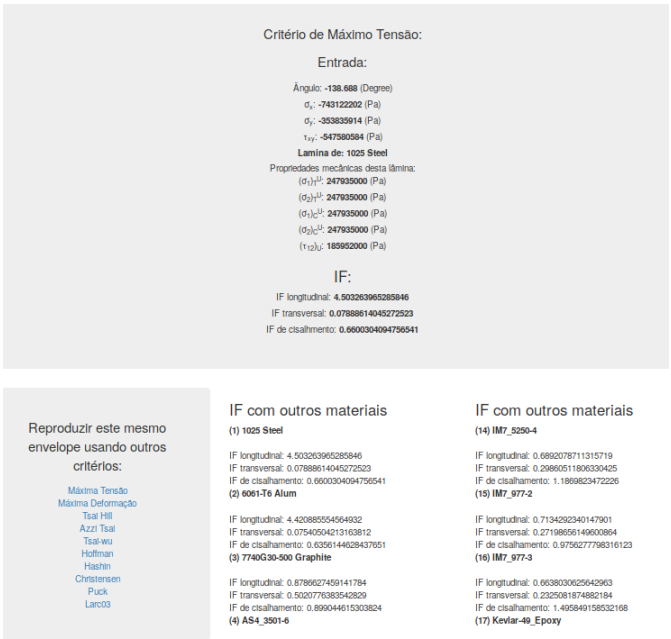

Figura 17: Espelho Web para uma função de critério de falha. (Fonte: Autor)

Estes espelhos recebem os argumentos no endereço URL no formato:

"baseURL?/angulo/lamina/00/sigmaX/sigmaY/tauXY/0/"

sendo preciso esclarecer que o argumento "lamina" é um valor inteiro que se relaciona com o numéro identificador da lâmina no arquivo JSON que contém as propriedades das lâminas, e o fragmento "sigmaX/sigmaY/tauXY" representa um estado de esforços em coordenadas globais (σ_x,σ_y,τ_{xy}). Já o fragmento inicial "baseURL" constitui a parte inicial do endereço URL que depende do servidor onde seja instalada a pasta que contém os espelhos.

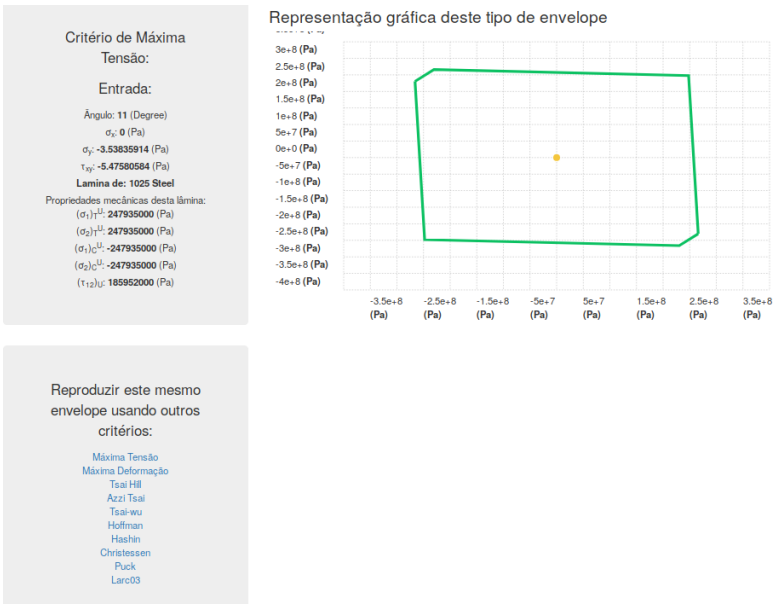

Figura 18: Espelho Web para uma função de Envelope de falha. (Fonte: Autor)

Com o intuito de facilitar a construção destes espelhos foram criadas duas pastas. Uma contém os arquivos com as funções que calculam os critérios de falha e outra contém os envelopes, em cada uma destas pastas principais é colocado o arquivo "macro_laminas.json" que contém as propriedades das lâminas usadas. Este arquivo JSON é o equivalente ao banco de dados estático SQLITE usado na versão Android ou ao banco de dados MySQL usado no módulo Django.

Aprecia-se (Figura 17) que os espelhos WEB não possuem caixas de entrada recebendo apenas argumentos pelo URL, mas permite usar outros critérios de falha considerando como os argumentos de entrada os já usados no cálculo mostrado isto por meio dos links que aparecem na parte esquerda.

No caso do espelho WEB do envelope (Figura 18) é possível aproveitar os argumentos de entrada com o mesmo método de links colocados na parte esquerda da figura. Dado que estes espelhos estão feitos com Bootstrap é possível visualizar sem dificuldade nas telas de telefones celulares do tipo smartphone ou dispositivos com telas menores ao tradicional

monitor de computador do tipo Desktop.

6.2 Resultados didáticos

Cumprindo com os objetivos inicias deste mestrado, foi desenvolvida uma ferramenta com muitos elementos didáticos que devem se analisar do ponto de vista pedagógico. Estes elementos didáticos estão presentes tanto no módulo Django como no módulo Android, e desenvolvidos como funções de uso exclusivo da interface gráfica.

6.2.1 Resultados didáticos no módulo Django

No módulo Django as interfaces contém funções especialmente desenhadas para o ambiente acadêmico. Desde o ponto de vista didático estas funções apresentam como vantagens:

a) Visualização de vários envelopes de falha de forma clara: O software comercial estudado Helius não permite a apresentação simultânea de vários envelopes de falha o que pode dificultar o processo didático de apresentar e explicar as diferenças entre os diversos tipos de critérios de falha, limitando a capacidade pedagógica do software gerador de envelopes de falha. O módulo Django deste sistema informático permite uma representação simultânea de vários envelopes de falha, colorindo automaticamente cada curva com uma cor diferente e preenchendo de cor a superfície interna do último envelope de falha construído.

b) Legendas explicitas: A comparação gráfica dos diversos tipos de critérios de falha obriga que se seja claramente identificada cada curva. Este módulo Django permite identificar cada curva ao legendar, de forma automática (Fig.19), cada curva que esteja sendo representada.

c) Reprodução dos envelopes numa linguagem de programação mais simples: A capacidade de programação computacional é uma destreza cobrada com maior enfases nas novas gerações de engenheiros, mas esta habilidade precisa ser cultivada progressivamente e com cuidado no caso dos futuros profissionais daquelas áreas afastadas das ciências da computação, por exemplo com códigos numa linguagem mais simples. Neste caso, as interfaces possuem um botão para obter o código MATLAB associado aos envelopes de falha representados na interface.

(d) Apresentação de conteúdos didáticos em formato HTML: Aproveitando a capacidade do Django, foi configurada uma pasta para guardar os arquivos HTML para ser usadas para apresentar os elementos de apoio visual de uma aula expositiva. Este elemento didático se integra diretamente com o administrador de urls deste framework para facilitar a apresentação dos arquivos HTML conforme os argumentos transmitidos.

Figura 19: Detalhe do Sistema automático de legenda e identificação de cores para distinguir cada curva. (Fonte: Autor)

Figura 20: Detalhamento do histórico de cálculos já efetuados. (Fonte: Autor)

(e) Histórico de cálculos dos índice de falha: Uma funcionalidade incorporada na interface de cálculo dos índices de falha foi o histórico (Fig. 20) dos cálculos já feitos durante a sessão o que permite efetuar comparações didáticas entre diversos materiais e estados de esforços.

6.2.2 Resultados didáticos no módulo Android

A primeira tela que o usuário visualiza ao entrar no módulo Android (Fig. 21(a)) representa um elemento didático caraterizado pela simplicidade que pode provocar um comportamento intuitivo por parte deste. A partir desta tela inicial é possível interagir com outros elementos visuais que apresentam de forma progressiva e pouco densa aquelas funcionalidades de auxilio pedagógico (Fig. 21(b)) tais como funções para ver um histórico

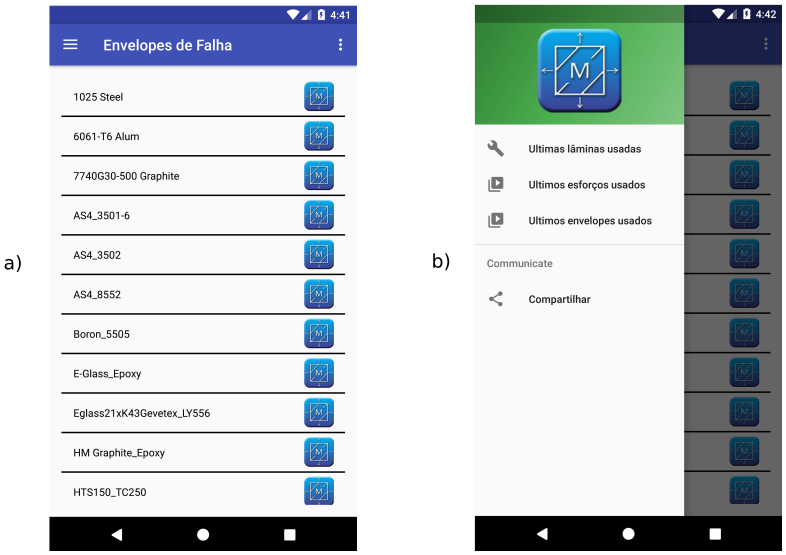

Figura 21: (a) Tela inicial didática do módulo Android. (b) Menu lateral com funções didáticas. (Fonte: Autor)

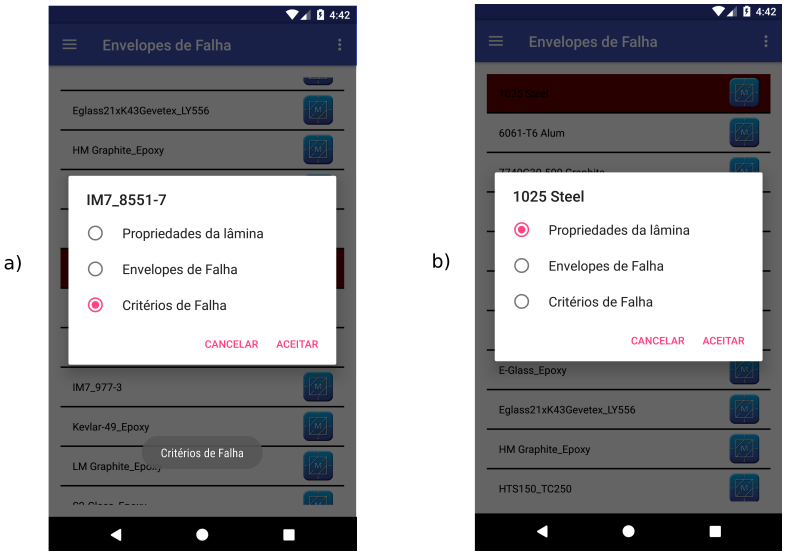

Figura 22: (a) Tela inicial didática do módulo Android. (b) Menu lateral com funções didáticas. (Fonte: Autor)

das últimas lâminas usadas, dos últimos esforços usados, dos últimos envelopes usados.

Outro elemento didático são os menus de funções (Fig. 22) os quais permitem limitar o fluxo de iterações do usuário com o intuito de manter o foco numa função em especifico, seja esta a função de visualizar e estudar as propriedades da lâminas, de calcular um índice de falha ou de construir um envelope. Os menus deste tipo são de escolha única com botões inferiores para indicar explicitamente os caminhos possíveis de retornar ou seguir adiante.

A partir deste menu (Fig. 22) pode-se visualizar a tela de propriedades de cada lâmina (Fig. 23) esquematizada da mesma forma que a tela inicial usando um ListView

para listar as informações que o módulo Android disponibiliza. Esta diagramação visual, uniforme para todas as telas, tem como intuito manter o espirito amigável e simples da interface gráfica e representa outro elemento didático produto deste mestrado. Este aviso temporário acima da tela permite reforçar, sem saturar, o conteúdo visual que em todo momento tenta se manter simples, claro e de fácil interpretação. Assim por exemplo no caso de um envelope de falha de máxima deformação (Fig. 24) este tipo de aviso temporário indica o numero de vértices do envelope.

Nesta mesma tela (Fig. 24) aprecia-se uma horizontalização da imagem, que contrasta com a maioria das telas com orientação vertical usadas no aplicativo, que se justifica na importância de representar com o maior clareza possível os envelopes sendo preciso usar a maior área possível da limitada tela do dispositivo Android.

Figura 23: (a) Tela de propriedades de uma lâmina. (b) Comentário explicativo de cada propriedade exibindo uma pulsação em cada propriedade. (Fonte: Autor)

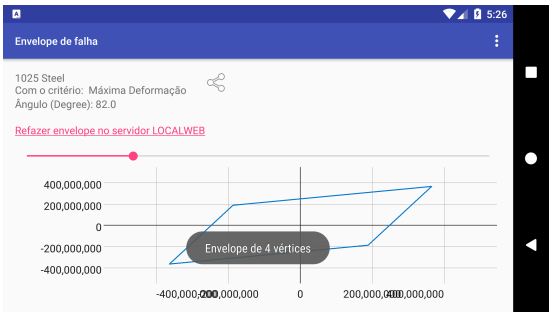

Figura 24: Envelope de falha com aviso temporário detalhando o tipo de envelope com apenas 4 vértices. (Fonte: Autor)

Nesta mesma tela o elemento didático do link URL que está associado com a frase "Refazer este envelope no servidor LOCALWEB" permite que o usuário possa refazer este

tipo de envelope, com a mesma condição de rotação, no servidor WEB com apenas um toque sem ter que transmitir manualmente os argumentos embutidos no link tais como material da lâmina, rotação e tipo de critério usado.

6.3 Comparações com software comercial

O software comercial de boa reputação para o estudo dos envelopes e critérios de falha, difundido intensamente dentro da comunidade acadêmica, é o Helius Composite (versão 2014) (AUTODESK, 2004-2016) da AutoDesk que possui uma ampla gama de produtos informáticos de excelente qualidade focados para o setor da engenharia mecânica e civil. Pode-se comparar os envelopes de falha gerados pelo Helius com aqueles gerados por este sistema informático e comentar duas diferenças principais:

(a) Necessidade de indicar o número de pontos:

No software Helius, é preciso indicar o número de pontos para construir um envelope de falha. O fato de ser necessário indicar este valor como entrada do software, sugere o uso do algoritmo de Gramham-Scan (CORMEN, 2009) que consome muita memória por ter que estudar vários pontos dentro e fora da curva poligonal correspondente com o envelope teórico. De fato, algum destes pontos podem se visualizar com clareza nos gráficos gerados pelo software comercial, seja numa representação de apenas 20 pontos (Fig.25 a), ou numa representação de 200 pontos (Fig. 25 b) observando-se um sério problema de construção do canto do envelope apesar do aumento no número de pontos calculados.

A impossibilidade de representar com exatidão os vértices dos envelopes de valores extremos, seja de máxima tensão ou máxima deformação, nem mesmo incrementando o número de pontos, cria um problema didático que pode gerar uma transmissão de conceitos errôneos. Isto não acontece neste sistema porque o algoritmo de construção procura diretamente, e exclusivamente, cada vértice deste tipo de envelope.

Assim, pode-se inferir que o algoritmo aqui desenhado consome menos memória e menos poder computacional. Cabe salientar que comparações do tipo quantitativo são difíceis de obter pelo fato que o software comercial não é aberto, o que impossibilitaria a separação do fragmento de código que representa o algoritmo puro de construção do envelope.

(b) Necessidade de indicar o passo:

Reforçando a hipótese do uso do algoritmo de Graham-Scan no software comercial, vê-se que no Helius é exigida a indicação do passo de iteração (Fig. 26) que é colocado por este, de forma inicial e automática, como um valor proporcional às resistências da lâmina estudada. Um valor elevado deste passo pode gerar ruído (fig. 26a) ou mesmo destruir totalmente o envelope (Fig. 26b).

Por outro lado, o algoritmo aqui implementado não gera nem ruído nem deformação,

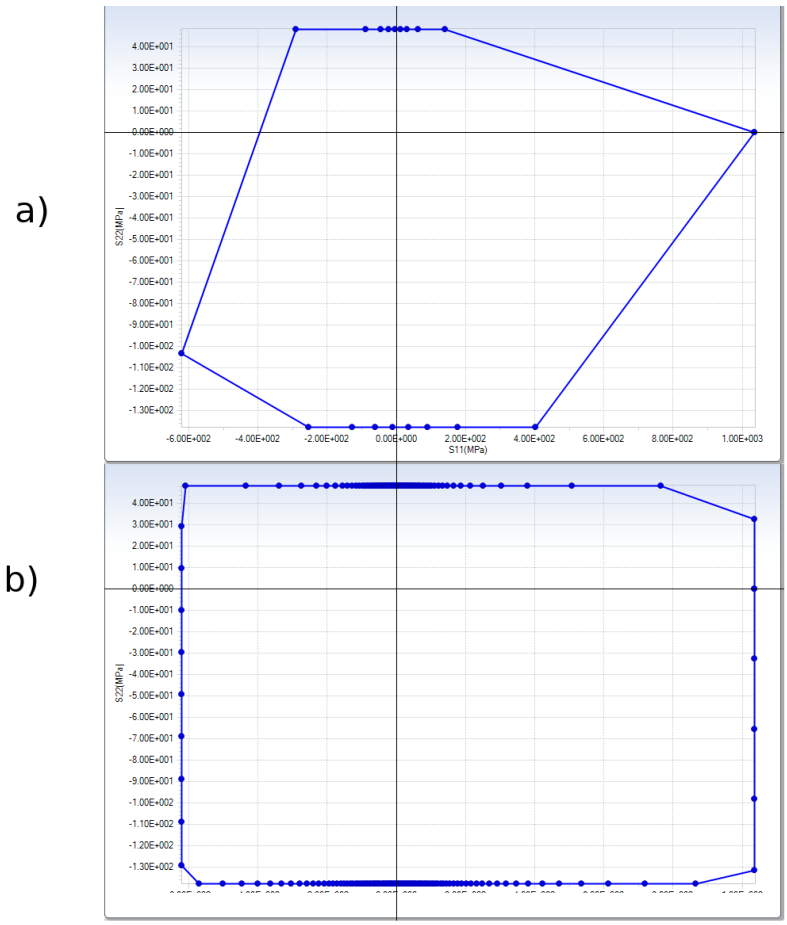

Figura 25: Envelope de máxima tensão gerado pelo Helius para uma lâmina com rotação nula. (a) Com 20 pontos. (b) Com 200 pontos. (Fonte: Autor)

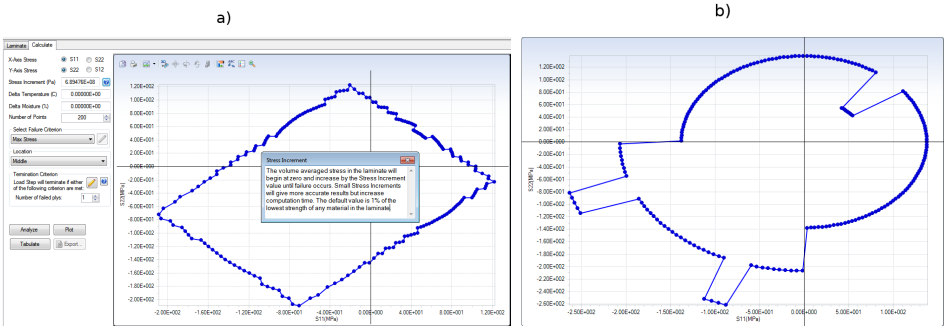

Figura 26: a) Interface do Helius para indicar o valor do passo. Ruído gerado por um valor elevado deste (lâmina com rotação de 45°). b) Ruído excessivo gerado por um valor elevado do passo (lâmina com rotação de 45°). (Fonte: Autor)

garantindo uma forma estável do envelope em todo momento.

6.4 Estudos de caso

Resulta interessante analisar tanto metais como materiais compósitos reforçados por fibra, dado que a geometria de cada envelope pode mudar por efeitos da rotação e por influência do parâmetro τ_{xy}. A incorporação de vértices no envelope, por efeitos da rotação é algo verdadeiramente indesejado já que, em tese, o envelope deveria ser uma curva suave. Efeito este que se observa principalmente nos metais isotrópicos, como no caso da lâmina "1025 Steel" onde apenas com uma rotação de 19° aprecia-se a aparição simétrica de 2 vértices (Fig. 27) nos planos de esforços com o parâmetro $\tau_{xy} = 0$.

Um fenômeno geométrico interessante acontece com os envelopes de máxima deformação onde aparecem, com pequenos ângulos de rotação da lâmina, envelopes de 5 vértices em lâminas reforçadas. Por exemplo, no caso do "HTS150 TC250" no intervalo [8°,18°] aparecem envelopes de 5 vértices (Fig. 31) fenômeno que não foi observado nos materiais metálicos para ângulos similares.

Este fenômeno de incorporação de novos vértices não aparece quando rotacionada uma lâmina do material compósito "HTS 150 TC250" (Fig. 28) o que se observa nos envelopes de máxima tensão construído com o parâmetro $\tau_{xy} = 0$ e os ângulos de 0°, 10°, 20°, 30°, 40°, 50°, 60°, 70°, 80°, 90°.

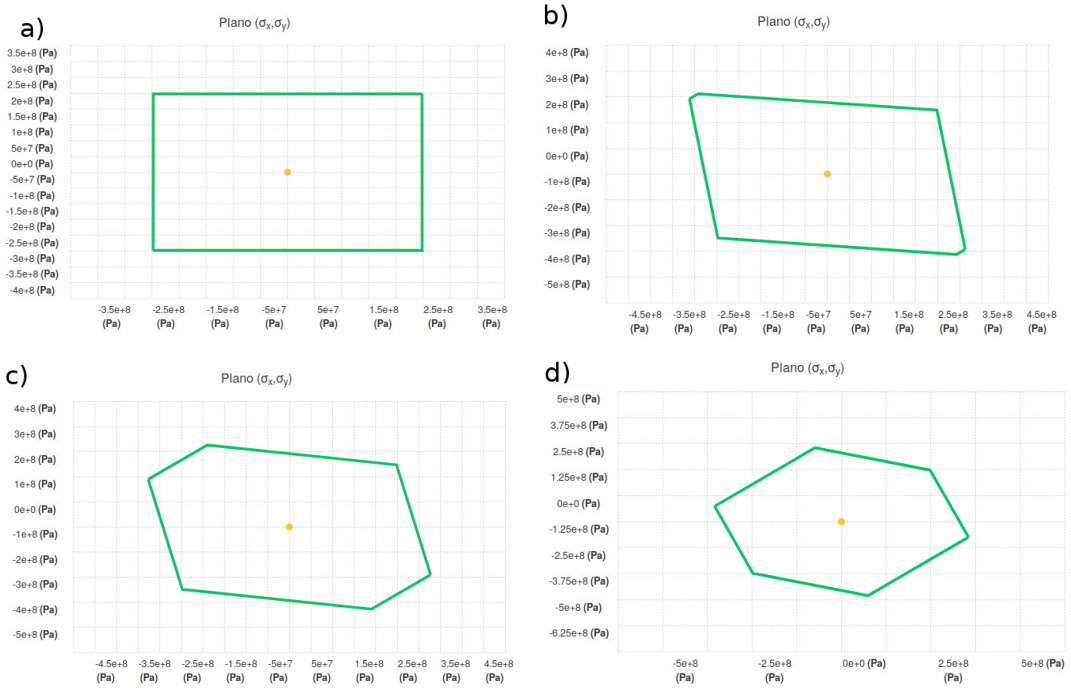

Figura 27: Envelope de máxima tensão da Lâmina de "1025 Steel" com rotações de (a) 0°, (b) 19°, (c) 23°, (d) 30°, e $\tau_{xy} = 0$. (Fonte: Autor)

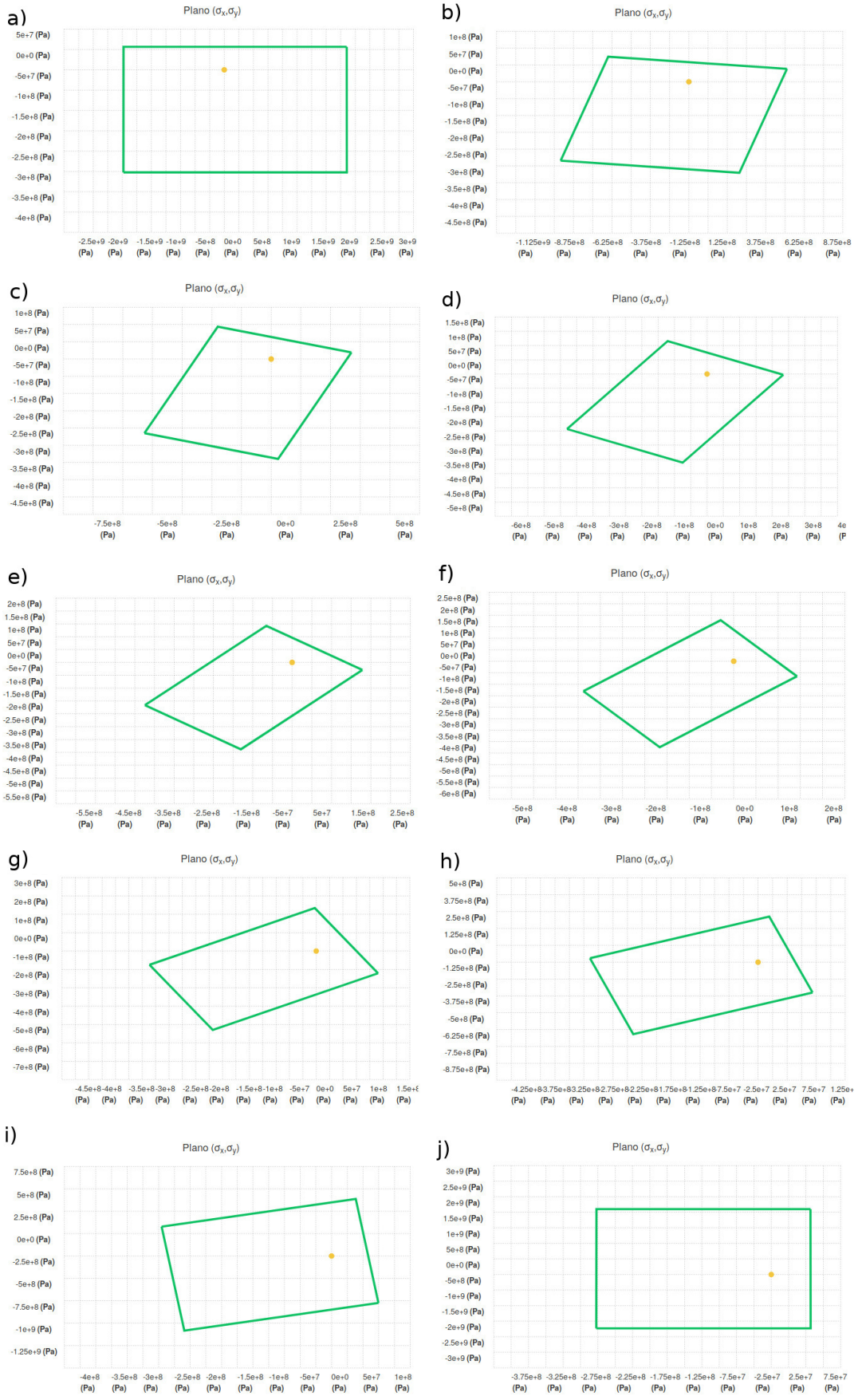

Figura 28: Envelope de máxima tensão da lâmina reforçada com fibra "HTS150 TC250" com $\tau_{xy} = 0$ e rotações de a) 0°, b) 10°, c) 20°, d) 30°, e) 40°, f) 50°, g) 60°, h) 70°, i) 80°, j) 90°. (Fonte: Autor)

No caso do software comercial Helius não foi possível confiar nos envelopes dado que ao parecer estes são uma aproximação qualitativa. A versão 2017 deste software. na sua distribuição demonstrativa gratuita não funcionou sendo apenas possível usar a versão 2016 de onde foram obtidos vários envelopes cujos pontos não geram um IF unitário. Do ponto de vista geométrico, o envelope de máxima tensão precisa apenas de poucos pontos para ser construído portanto a geração de numerosos pontos por parte do Helius junto com a desconsideração do parâmetro τ_{xy} que este software faz, são duas desvantagens consideradas antididáticas.

A qualidade gráfica do envelope de Azzi-Tsai (Fig. 29) depende do número de pontos usados na construção da elipse, sendo possível um envelope bastante continuo indicando um elevado número de pontos.

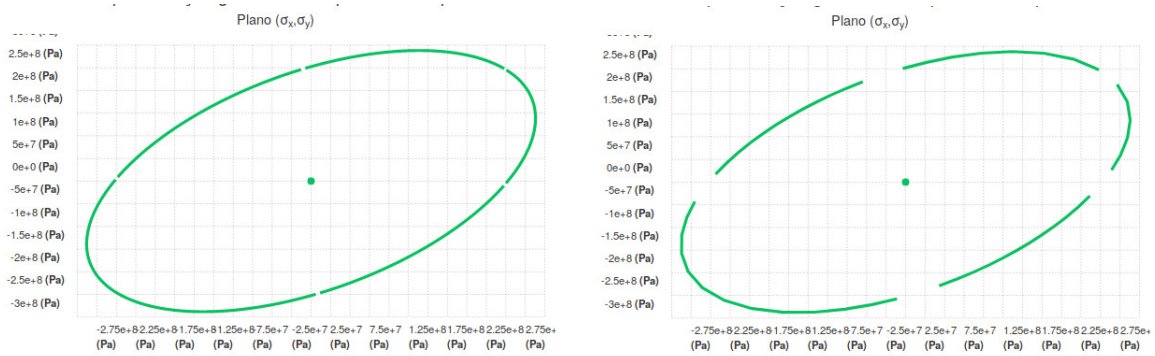

Figura 29: Envelope "Azzi-Tsai" , do "1025 Steel" com a) 550, b) 55 pontos. (Fonte:Autor)

O critério de Tsai-Hill (Fig. 30) gerado pelo softwware Helius parece com o critério de Azzi-Tsai, sendo este descompasso antididático.

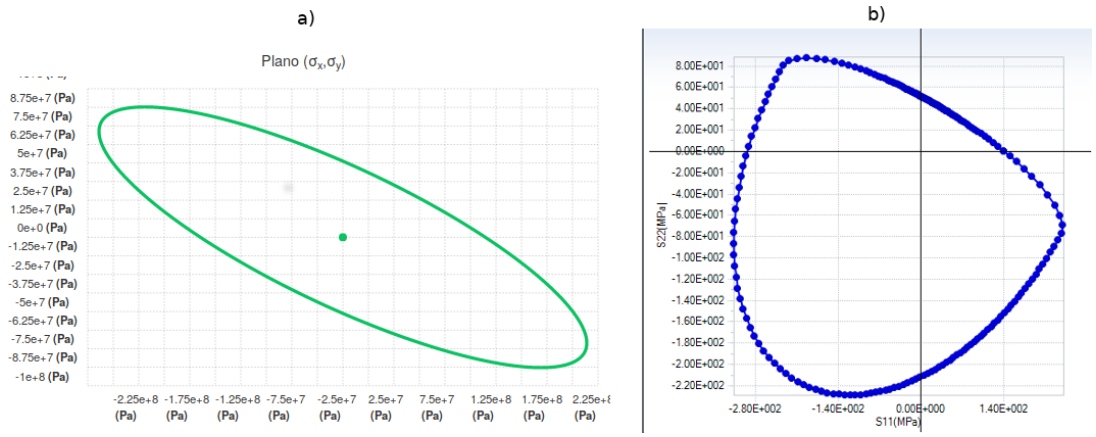

Figura 30: Envelopes de Tsai-Hill, de uma lâmina de Graphite 7740G30-500 com rotação de 30°, construídos: a) com o software deste mestrado com paramêtro $\tau_{xy} = 0$, b) como o software Helius. (Fonte:Autor)

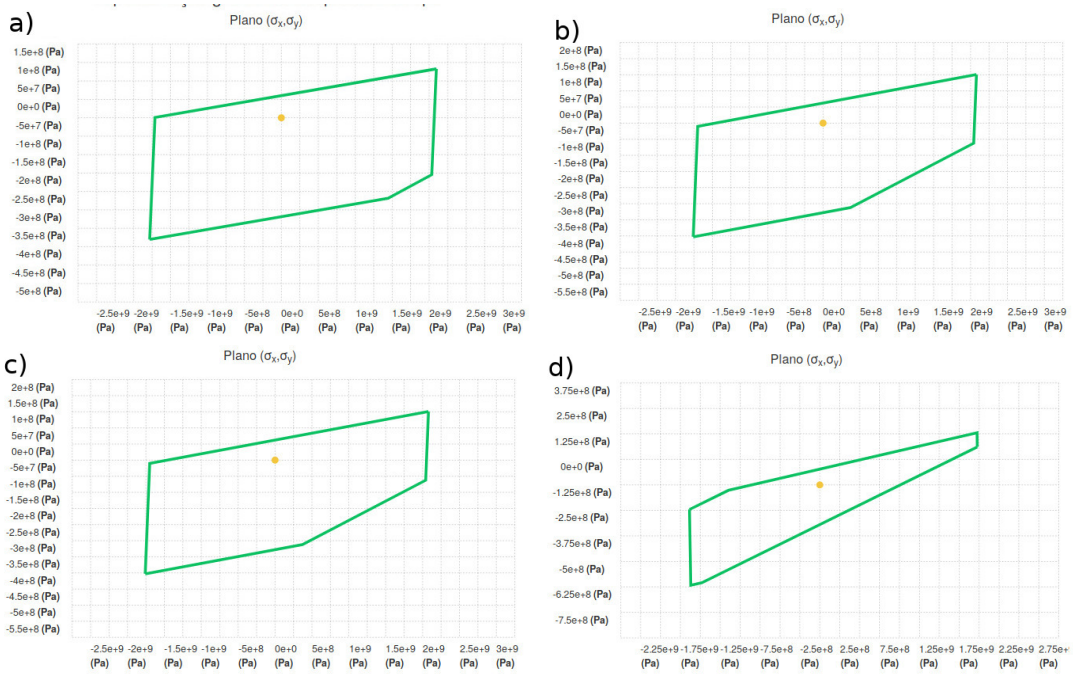

Figura 31: Envelopes de Máxima Deformação, de uma lâmina "HTS150 TC250" com rotação de a) 8°, b) 10°, c) 14°, d) 18°. (Fonte:Autor)

É preciso lembrar que os critérios de falha aqui estudados definem um volume no espaço (σ_x,σ_y,τ_{xy}) e por isso é essencial considerar o valor de cada tipo de esforço seja este como variável principal ou como parâmetro. Sendo estranho que o valor de τ_{xy} seja desconsiderado pelo software Helius na construção de todos os envelopes que ele oferece.

Neste sistema informático o envelope de "máxima deformação" recebe como parâmetro de entrada a deformação γ_{xy} o que se justifica no fato que as propriedades apresentadas, na interface de saída de este envelope (Fig. 32), são apenas as deformaççoes extremas da lâmina.

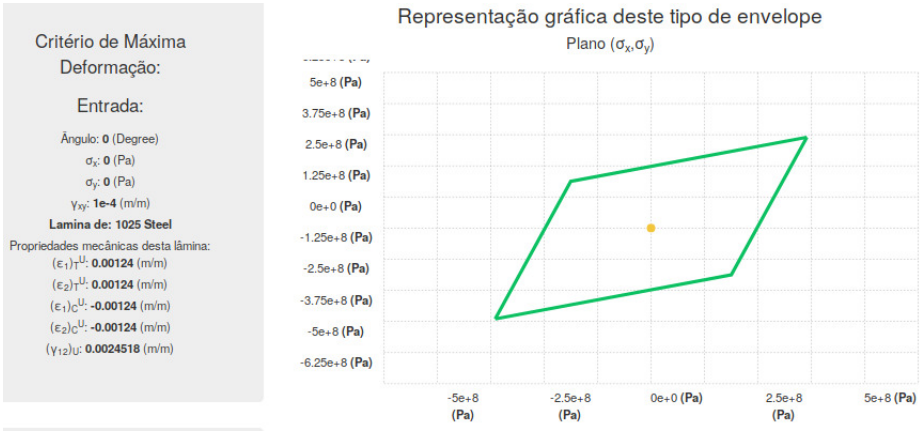

Figura 32: Interface do envelope "Máxima Deformação", apresentando propriedade da lâmina do tipo deformações extremas (Fonte:Autor)

O envelope de Hoffman (Fig. 33) assim como o envelope de Tsai-Wu é de fácil construção no caso de mateirias isotrópicos (Fig. 34).

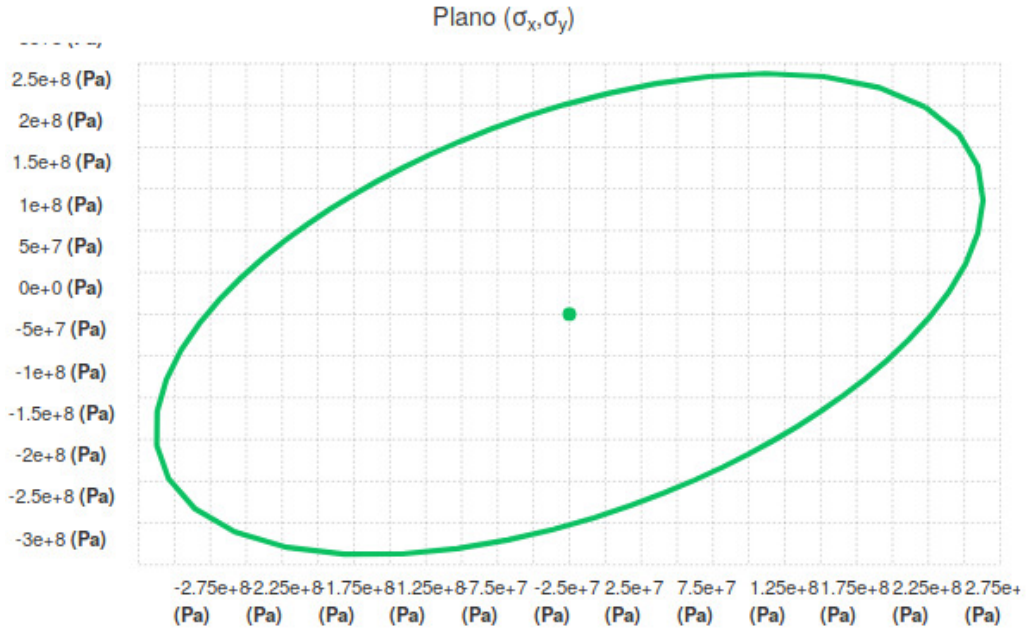

Figura 33: Envelope de "Hoffman" de uma lâmina de "1025 Steel" com rotação de 0°. (Fonte:Autor)

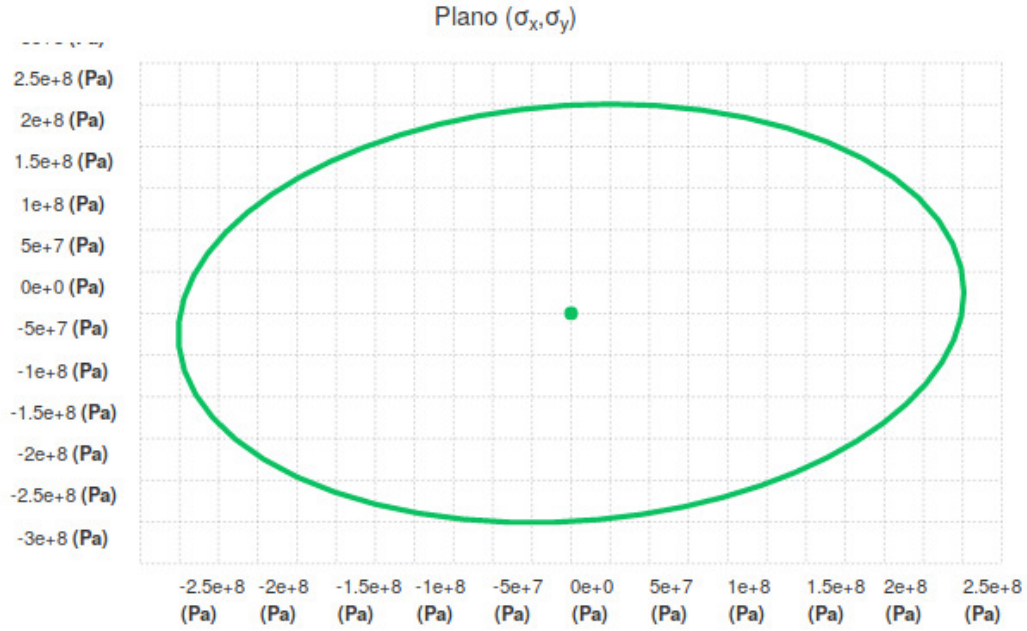

Figura 34: Envelope de "Tsai-Wu" de uma lâmina de "1025 Steel" com rotação de 0° e parâmetro biaxial $\sigma_{biaxial}$ de 185 GPa. (Fonte:Autor)

O envelope de Christensem também depende do número de pontos usados na

construção da curva cônica sendo possível obter resultados claros com apenas 55 pontos (Fig. 35)

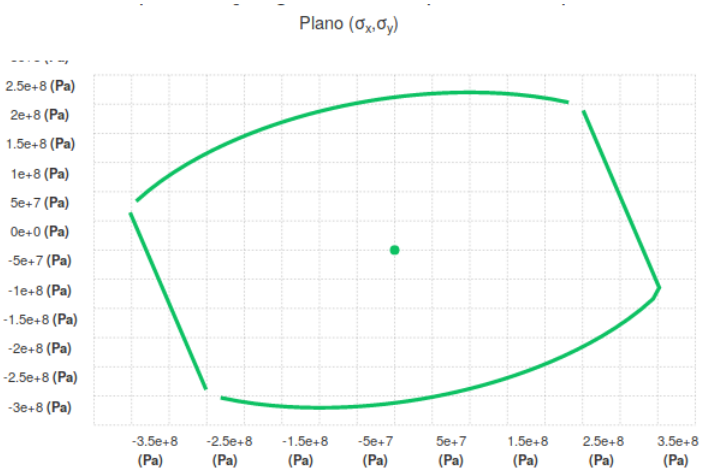

Figura 35: Envelope de "Christensen" de uma lâmina de "1025 Steel" com rotação de 30° (Fonte:Autor)

O envelope de "Hashin" (Fig. 36) está representado num plano que contêm os eixos do plano (σ_1, σ_2) com o intuito de facilitar a interpretação dos resultados, dado que este critério considera vários fenômenos tanto na fibra como na matriz.

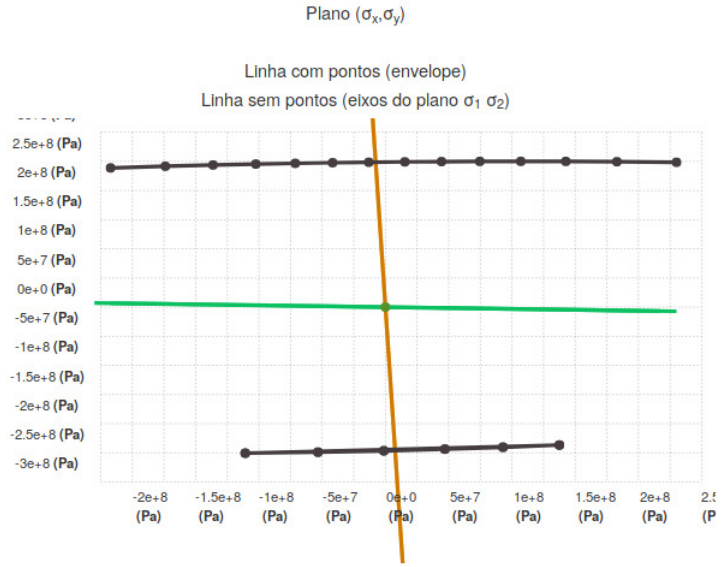

Figura 36: Envelope de "Hashin" de uma lâmina de "1025 Steel" com rotação de 10° (Fonte:Autor)

Os envelopes de Puck (Fig. 37) e Larc03 (Fig. 38), com rotação nula ($\theta = 0°$), os quais foram programados com ajuda do método Graham-Scan apresentam um comportamento ruídoso similar às distorções vistas no software Helius (Fig. 26) e que

apenas podem ser superadas com ajuda de ténicas de filtragem próprias da computação gráfica.

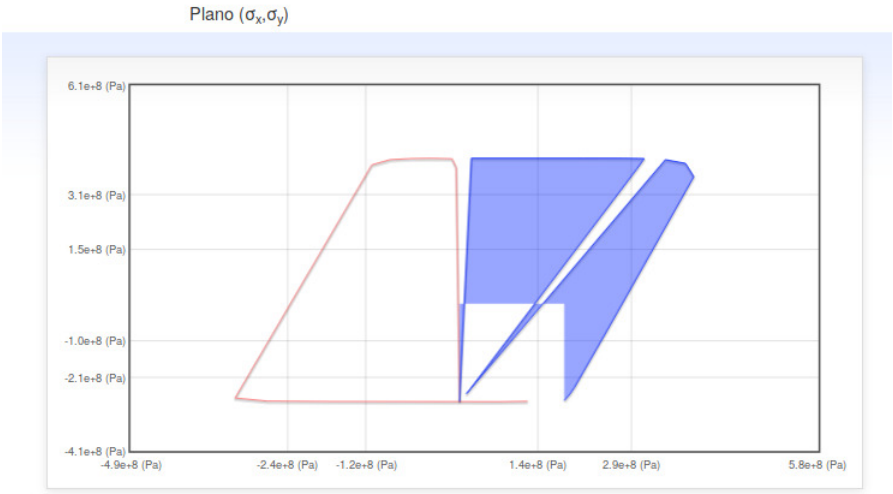

Figura 37: Envelope de "Puck" de uma lâmina de "1025 Steel" $\theta = 0°$ (Fonte:Autor)

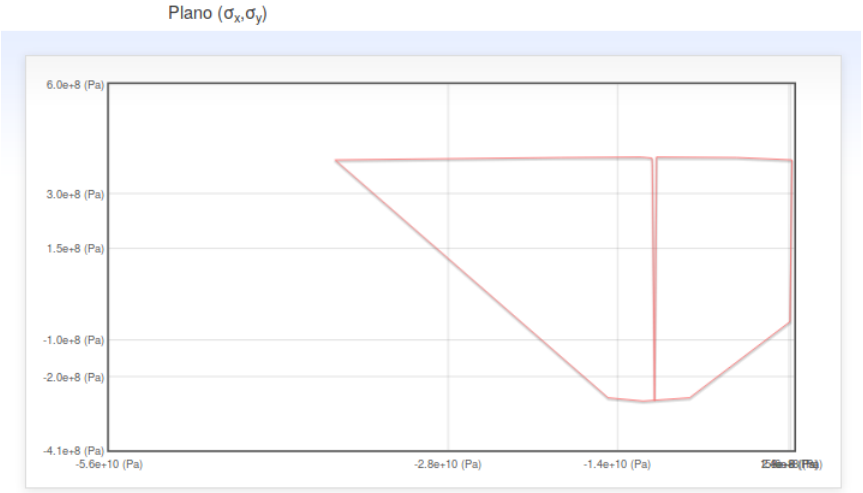

Figura 38: Envelope de "Larc03" de uma lâmina de "1025 Steel" $\theta = 0°$ (Fonte:Autor)

7 CONCLUSÕES

Após analisar e dissertar com as ideias sobre critérios e envelopes de falha mecânica usadas no desenvolvimento deste sistema informático de uso educativo, é possível concluir que:

Os critérios de falha constituem uma ótima ferramenta no estudo da mecânica dos materiais compósitos dado que é possível considerar princípios mecânicos e caraterísticas de cada material. Quando o critério for usado de forma puramente numérica este representa uma medida útil para o projetista que deseja estudar um caso específico. E quando for usado na forma de envelope de falha, permite projetar aquelas considerações de desenho mecânico com uma ótica global. O software comercial parece estar mais focado em gerar envelopes aproximados dado que na etapa inicial de desenho os valores numéricos não são detalhados, mas é essencial que no ambiente acadêmico tanto a função para calcular o IF como o envelope sejam numericamente exatos para não desvirtuar conceitos essências da mecânica.

No desenho de um sistema informático, de uso educativo, pode-se aproveitar as mais diversas tecnologias sempre que sejam combinadas de forma adequada, cuidadosa e acorde com a realidade educativa considerada. O desenvolvimento torna-se eficiente quando se tiver domínio dos algoritmos numéricos conhecendo com exatidão o poder de computacional disponível. Como neste mestrado onde foram desenhados, os algoritmos de forma analítica para que estes precisassem do menor poder de computação possível dado que o sistema informático estaria focado em dispositivos de pouco poder computacional como no caso dos telefones celulares com sistema operacional Android. Este domínio numérico garante uma rápida codificação mesmo sem ajuda de bibliotecas especializadas dado que a falta destas e a capacidade de programação numérica permite codificar aquelas funções algébricas necessárias. Este mesmo entendimento numérico permite arquitetar o software dentro do framework Django e projetar os devidos testes, com entradas aleatórias, para verificar o cumprimento das proposições geométricas formuladas.

O desenho e construção das interfaces é uma tarefa informática e pedagógica onde os conceitos de modelo pedagógico e a experiência prévia de ensino focado no software Mech-Gcomp permitiu criar um sistema usando o passe de argumentos por URL, hiperlinks, eventos conforme a experiência didática relacionada com as disciplinas de materiais compósitos, e a entrega oportuna de segmentos de código para facilitar o ensino. Desconsiderar estes elementos pedagógicos limitaria os códigos desenvolvidos para serem fragmentos isolados de cálculo numérico cujos resultados além de difíceis de comparar, não forneceriam uma ferramenta de rápida configuração para ser usada na sala de aula.

8 SUGESTÕES PARA TRABALHOS FUTUROS

- Com o intuito de otimizar os códigos escritos tanto em python como em Java, faz necessário a reescrita dos mesmos usando o paradigma da "programação funcional" dado que estas duas linguagens de programação, nas suas versões mais recentes, permitem programação muito conveniente no ambiente acadêmico e nos projetos onde seja preciso programar várias formulações numéricas.
- O estudo dos critérios de falha pode se beneficiar da criação de novas telas para construção dos envelopes de falha em outros sistemas de eixos coordenados, diferentes ao sistema do plano (σ_x, σ_y) usado neste mestrado. Para este fim, é necessário a reformulação das funções computacionais que implementem os envelopes de falha prévio estudo algébrico das formulações de cada um dos critérios aqui estudados.
- Os fragmentos de código python desenvolvidos para cada critério e envelope de falha podem ser testados no microcomputador educativo "Raspberry Pi" dado que os mesmos implementam algoritmos simples que funcionam em sistemas equivalentes como os Smartphones Android.
- Melhorar as interfaces do módulo Android incorporando fragments para adicionar um comportamento responsivo.
- Dado que o projeto MECH-G está em contante crescimento sendo alvo de diversas técnicas de programação conforme os moldes do framework Django, considera-se pertinente recomendar o uso do método de prototipagem inicial no qual se defina um Módulo Inerte de Programação Inicial (MIPI) tal como se fez neste trabalho. De forma empírica se aprendeu que o MIPI representa uma ótima forma para incorporar novas funcionalidades, resultando imperioso recomendar seu uso para eventuais expansões do código.

REFERÊNCIAS

AHO, A. et al. *Compilers: Principles, Techniques, and Tools*. [S.l.]: Pearson Education, 2011. ISBN 9780133002140. Citado 3 vezes nas páginas 39, 40 e 43.

ALLEN, G.; OWENS, M. *The Definitive Guide to SQLite*. [S.l.]: Apress, 2011. (Books for professionals by professionals). ISBN 9781430232261. Citado na página 41.

ARFKEN, G.; WEBER, H. *Mathematical Methods For Physicists International Student Edition*. [S.l.]: Elsevier Science, 2005. ISBN 9780080470696. Citado 11 vezes nas páginas 16, 21, 30, 41, 49, 51, 54, 56, 60, 68 e 80.

AUTODESK. *Helius Composite*. 2004–2016. <http://www.autodesk.com/products/helius-composite/overview>. Citado 3 vezes nas páginas 17, 48 e 89.

AZZI, V. D.; TSAI, S. W. Anisotropic strength of composites. *Experimental Mechanics*, v. 5, n. 9, p. 283–288, 1965. ISSN 1741-2765. Citado na página 28.

BARBERO, E. *Introduction to Composite Materials Design, Second Edition*. [S.l.]: Taylor & Francis, 2010. (Composite Materials). ISBN 9781420079159. Citado na página 23.

BARROSO, L. et al. *Calculo Numerico - Com Aplicaçoes*. [S.l.]: HARBRA EDITORA, 1987. ISBN 9788529400891. Citado na página 60.

BASSETT, L. *Introdução ao JSON: Um guia para JSON que vai direto ao ponto*. [S.l.]: NOVATEC, 2015. ISBN 9788575224519. Citado na página 41.

BENNETT, J. *Practical Django Projects*. Berkely, CA, USA: Apress, 2008. ISBN 1590599969, 9781590599969. Citado 3 vezes nas páginas 38, 40 e 61.

BONILLA, M.; PRETTO, N. D. L. *Inclusão digital: polêmica contemporânea*. [S.l.]: SciELO - EDUFBA, 2011. ISBN 9788523212063. Citado na página 44.

BRESSERT, E. *SciPy and NumPy*. [S.l.]: O'Reilly, 2012. (Oreilly and Associate Series). ISBN 9781449305468. Citado na página 80.

CANALTECH. *Aparelhos Android dominam 70móveis, diz pesquisa - Mercado*. 2014. Disponível em: <https://canaltech.com.br/mercado/Aparelhos-com-Android-dominam-70-das-vendas-de-dispositivos-moveis-diz-pesquis/>. Citado na página 42.

CHRISTENSEN, R. *Mechanics of Composite Materials*. [S.l.]: Dover Publications, 2012. (Dover Civil and Mechanical Engineering). ISBN 9780486136660. Citado 2 vezes nas páginas 23 e 32.

CONVERSE, T.; PARK, J. *PHP: a bíblia*. [S.l.]: CAMPUS, 2003. ISBN 9788535211306. Citado 2 vezes nas páginas 39 e 66.

CORMEN, T. *Introduction to Algorithms*. [S.l.]: MIT Press, 2009. ISBN 9780262033848. Citado 4 vezes nas páginas 48, 54, 60 e 89.

DATE, C. *SQL e Teoria Relacional: Como escrever códigos SQL precisos*. [S.l.]: Novatec Editora, 2015. ISBN 9788575224335. Citado na página 41.

DAVILA, C. G.; CAMANHO, P. P.; ROSE, C. A. Failure criteria for frp laminates. *Journal of Composite Materials*, v. 39, n. 4, p. 323–345, 2005. Citado na página 35.

DAYLEY, B. *Sams Teach Yourself Django in 24 Hours*. [S.l.]: Pearson Education, 2008. (Sams Teach Yourself). ISBN 9780132715492. Citado 2 vezes nas páginas 40 e 63.

DAYLEY, B. *jQuery and JavaScript in 24 Hours, Sams Teach Yourself*. [S.l.]: Pearson Education, 2013. (Sams Teach Yourself). ISBN 9780133414196. Citado na página 39.

DEITEL, P.; DEITEL, H. *Java How to Program (Early Objects)*. [S.l.]: Pearson, 2015. (Always learning). ISBN 9780133807806. Citado na página 42.

DOYLE, J.; STRETCH, D. The classification of programming languages by usage. *International Journal of Man-Machine Studies*, v. 26, n. 3, p. 343 – 360, 1987. ISSN 0020-7373. Citado na página 42.

FILHO, W. da S. *Costa Ribeiro: ensino, pesquisa e desenvolvimento da física no Brasil*. [S.l.]: SciELO - EDUEPB, 2013. ISBN 9788578792763. Citado na página 43.

FLANAGAN, D. *JavaScript: The Definitive Guide: The Definitive Guide*. [S.l.]: O'Reilly Media, 2006. ISBN 9780596554477. Citado 2 vezes nas páginas 39 e 41.

FREIRE, P. *Educação como prática da liberdade*. [S.l.]: Paz e Terra, 2000. ISBN 9788521901099. Citado na página 44.

FREIRE, P.; NOGUEIRA, A. *Que fazer: teoria e prática em educação popular*. [S.l.]: Vozes, 1989. Citado na página 44.

FRIESEN, J. *Learn Java for Android Development*. [S.l.]: Apress, 2013. ISBN 9781430257233. Citado na página 42.

GATLIN, J. *Bill Gates: The Path to the Future*. [S.l.]: HarperCollins, 2009. ISBN 9780061967887. Citado 2 vezes nas páginas 38 e 61.

GEHRING, J. *Grapview*. 2017. Disponível em: <http://www.android-graphview.org/>. Citado 2 vezes nas páginas 54 e 64.

GOOGLE. *GTS: Plotting library*. 2007–2014. <https://github.com/flot/flot>. Citado 3 vezes nas páginas 39, 45 e 54.

GOUVEA, A. d. R. et al. Critérios de falha e otimização de estruturas de materiais compósitos usando o método dos elementos de contorno. [sn], 2006. Citado na página 32.

GUELICH, S.; GUNDAVARAM, S.; BIRZNIEKS, G. *CGI Programming with Perl*. [S.l.]: O'Reilly Media, Incorporated, 2000. (Nutshell handbooks). ISBN 9781565924192. Citado na página 39.

GUIHOT, H. *Pro Android Apps Performance Optimization*. [S.l.]: Apress, 2012. ISBN 9781430240006. Citado na página 42.

GUREWICH, O.; GUREWICH, N. *Borland C++ Multimedia Programming*. [S.l.]: Sybex, 1994. ISBN 9780782115505. Citado na página 39.

HASHIN, Z. Failure criteria for unidirectional fiber composites. *Journal of applied mechanics*, American Society of Mechanical Engineers, v. 47, n. 2, p. 329–334, 1980. Citado na página 31.

HOFFMAN, O. The brittle strength of orthotropic materials. *Journal of Composite Materials*, SAGE Publications Sage UK: London, England, v. 1, n. 2, p. 200–206, 1967. Citado na página 31.

HUNT, B. et al. *A Guide to MATLAB: For Beginners and Experienced Users*. [S.l.]: Cambridge University Press, 2006. ISBN 9781139452533. Citado na página 41.

JANERT, P. *Gnuplot in Action*. [S.l.]: Manning Publications Company, 2016. ISBN 9781633430181. Citado 2 vezes nas páginas 63 e 66.

KARCH, M. *Android for Work: Productivity for Professionals*. [S.l.]: Apress, 2010. (Apresspod Series). ISBN 9781430230007. Citado na página 38.

KATTAN, P. *MATLAB Guide to Finite Elements: An Interactive Approach*. [S.l.]: Springer Berlin Heidelberg, 2010. ISBN 9783540706984. Citado 6 vezes nas páginas 9, 17, 20, 23, 26 e 42.

KAW, A. *Mechanics of Composite Materials, Second Edition*. [S.l.]: CRC Press, 2005. (Mechanical and Aerospace Engineering Series). ISBN 9781420058291. Citado na página 23.

KUNZ, G. *Chartist.js*. 2017. Citado 3 vezes nas páginas 45, 54 e 82.

LAMBERT, L. et al. *Internet: A Historical Encyclopedia*. [S.l.]: ABC-CLIO, LLC, 2005. ISBN 9781851096596. Citado na página 38.

LAURIE, B.; LAURIE, P. *Apache: The Definitive Guide*. [S.l.]: O'Reilly Media, Incorporated, 2003. (Definitive Guide Series). ISBN 9780596002039. Citado na página 39.

LECHETA, R. *Google Android 4ª edição: Aprenda a criar aplicações para dispositivos móveis com o Android SDK*. [S.l.]: Novatec Editora, 2015. ISBN 9788575224403. Citado na página 17.

LIBBY, A.; GUPTA, G.; TALESRA, A. *Responsive Web Design with HTML5 and CSS3 Essentials*. [S.l.]: Packt Publishing, 2016. ISBN 9781783553082. Citado na página 62.

LINASCHKE, J. *Getting the Most from Instagram*. [S.l.]: Pearson Education, 2011. ISBN 9780132875776. Citado na página 39.

LUBIN, G. *Handbook of Composites*. [S.l.]: Springer US, 2013. ISBN 9781461571391. Citado 2 vezes nas páginas 17 e 23.

LUTZ, M. *Learning Python*. [S.l.]: O'Reilly Media, 2013. (Safari Books Online). ISBN 9781449355692. Citado na página 39.

MACHARLA, P. *Android Continuous Integration: Build-Deploy-Test Automation for Android Mobile Apps*. [S.l.]: Apress, 2017. ISBN 9781484227961. Citado na página 66.

MANTIČ, V. *Mathematical Methods and Models in Composites*. [S.l.]: Imperial College Press, 2013. (Computational and Experimental Methods in Structures). ISBN 9781783264117. Citado na página 33.

MARTELLI, A. *Python in a Nutshell: A Desktop Quick Reference*. [S.l.]: O'Reilly Media, 2006. (In a Nutshell (O'Reilly)). ISBN 9781449379100. Citado na página 40.

MASSARI, V. *Agile Scrum Master no Gerenciamento Avançado de Projetos:*. [S.l.]: Brasport, 2016. ISBN 9788574527796. Citado na página 42.

MISES, R. v. Mechanik der festen körper im plastisch-deformablen zustand. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, v. 1913, p. 582–592, 1913. Citado na página 27.

MONTEIRO, J. *Google Android: crie aplicações para celulares e tablets*. [S.l.]: Casa do Código, 2014. ISBN 9788566250916. Citado na página 64.

NEGUS, C. *Linux Bible*. [S.l.]: Wiley, 2015. (Bible). ISBN 9781118999882. Citado na página 45.

OKUYAMA, F.; MILETTO, E.; NICOLAO, M. *Desenvolvimento de Software I: Conceitos Básicos - Série Tekne*. [S.l.]: Bookman Editora, 2014. (Tekne). ISBN 9788582601464. Citado na página 42.

PANOSSO, G. *Analysis of failure criteria based on physical phenomenas for laminate composite materials*. Dissertação (Mestrado) — Universidade de Campinas., 2012. Citado na página 32.

PATEL, J. *Data Structure, Algorithms and Design Techniques*. [S.l.]: Jitendra, 2015. ISBN 9781105490941. Citado na página 41.

PUCK, A.; SCHÜRMANN, H. Failure analysis of frp laminates by means of physically based phenomenological models. *Composites Science and Technology*, Elsevier, v. 58, n. 7, p. 1045–1067, 1998. Citado 4 vezes nas páginas 9, 33, 34 e 35.

SANCHEZ, T. *Programando com Perl*. [S.l.]: BRASPORT, 2010. ISBN 9788574524856. Citado na página 39.

SCHWARTZ, R. T. et al. Book; Book/Illustrated. *Composite materials workshop : [papers]*. [S.l.]: Stamford, Conn. : Technomic Pub. Co, 1968. "Proceedings of the summer workshop: Physical aspects of composite materials, held at Washington University, St. Louis, Mo., July 13-21, 1967.". Citado na página 27.

SERSON, R. *Programação Orientada a Objetos com Java 6 - Curso universitário*. [S.l.]: BRASPORT, 2000. ISBN 9788574522234. Citado na página 42.

SOUZA, J. D. *Microsoft Windows 10*. [S.l.: s.n.], 2016. Citado na página 45.

STEWART, J. *Python for Scientists*. [S.l.]: Cambridge University Press, 2014. ISBN 9781107061392. Citado na página 49.

SUEHRING, S. *MYSQL - A BÍBLIA*. [S.l.]: CAMPUS, 2011. ISBN 9788535210842. Citado 2 vezes nas páginas 41 e 66.

TAHCHIEV, P.; LEME, F.; MASSOL, V. *JUnit in Action*. [S.l.]: Manning, 2010. (Manning Pubs Co Series). ISBN 9781935182023. Citado na página 66.

TANENBAUM, A. *Computer Networks*. [S.l.]: Prentice Hall PTR, 2003. (Computer Networks, p. 3). ISBN 9780130661029. Citado 3 vezes nas páginas 38, 39 e 40.

TANENBAUM, A. *Sistemas operacionais modernos*. [S.l.]: Prentice-Hall do Brasil, 2010. ISBN 9788576052371. Citado na página 42.

TATNALL, A.; OSORIO, J.; VISSCHER, A. *Information Technology and Educational Management in the Knowledge Society: IFIP TC3 WG3.7, 6th International Working Conference on Information Technology in Educational Management (ITEM) July 11-15, 2004, Las Palmas de Gran Canaria, Spain*. [S.l.]: Springer US, 2004. (IFIP Advances in Information and Communication Technology). ISBN 9780387240442. Citado 2 vezes nas páginas 38 e 43.

TOSI, S. *Matplotlib for Python Developers*. [S.l.]: Packt Publishing, Limited, 2009. (From technologies to solutions). ISBN 9781847197917. Citado na página 47.

TSAI, S. W.; WU, E. M. A general theory of strength for anisotropic materials. *Journal of composite materials*, SAGE Publications, v. 5, n. 1, p. 58–80, 1971. Citado na página 30.

VARANASI, B. *Introducing Gradle*. [S.l.]: Apress, 2015. ISBN 9781484210314. Citado na página 75.

YENER, M.; DUNDAR, O. *Expert Android Studio*. [S.l.]: Wiley, 2016. ISBN 9781119110712. Citado na página 42.