UNIVERSIDADE FEDERAL DO RECÔNCAVO DA BAHIA

Centro de Ciências Exatas e Tecnológicas
Bacharelado em Engenharia Mecânica

Adaptação de Software para Análise de Tensão Através do Método dos Elementos Finitos em Python

Luan Mello Vilas Boas

UNIVERSIDADE FEDERAL DO RECÔNCAVO DA BAHIA
Centro de Ciências Exatas e Tecnológicas
Bacharelado em Engenharia Mecânica

Adaptação de Software para Análise de Tensão Através do Método dos Elementos Finitos em Python

Luan Mello Vilas Boas

Trabalho de conclusão de curso submetido à Universidade Federal do Recôncavo da Bahia sob a orientação do professor Me. Vânio Vicente Santos de Souza com a finalidade de cumprir os requisitos para o grau de Bacharel em Engenharia Mecânica.

Comissão de Examinadores:

_____

Professor Me. Vânio Vicente Santos de Souza
CETEC – UFRB (Orientador)

_____

Professor Leonardo Rafael Teixeira Cotrim Gomes
CETEC – UFRB

_____

Professor Esp. Luccas Barbosa Carneiro
CETEC – UFRB

Cruz das Almas – Bahia
12 de Julho de 2019

SUMÁRIO

Lista de Figuras

Lista de Siglas

AEM: Análise de Elementos Finitos.

ASME: American Society of Mechanical Engineers.

CAD: Computer Aided Design.

CAE: Computer Aided Engineering.

CAM: Computer Aided Manufacturing.

cm: centímetro.

IDE: Ambiente de Desenvolvimento Integrado.

MEF: Método dos Elementos Finitos.

m: metro.

N: Newton.

PMTA: Pressão Máxima de Trabalho Admissível.

Pa: Pascal.

Lista de Símbolos

$\boldsymbol{\varepsilon_C}$**:** Deformação Circunferencial

$\boldsymbol{\varepsilon_R}$**:** Deformação Radial

$\boldsymbol{\varepsilon_T}$**:** Deformação Tangencial

$\boldsymbol{\mu}$**:** Coeficiente de Combinação Linear

$\boldsymbol{\nu}$**:** Coeficiente de Poisson

$\boldsymbol{\theta}$**:** Ângulo

$\boldsymbol{\sigma}$**:** Tensão

$\boldsymbol{\sigma_L}$**:** Tensão Longitudinal

$\boldsymbol{\sigma_C}$**:** Tensão Circunferencial

Γ**:** Região da fronteira

$\boldsymbol{n}$**:** Vetor Unitário

$\boldsymbol{A}$**:** Área

$\boldsymbol{C_{ijkl}}$**:** Tensor Rigidez

$\boldsymbol{E}$**:** Módulo de Elasticidade

$\boldsymbol{G_C}$**:** Tenacidade

$\boldsymbol{J}$**:** Jacobiano

$\boldsymbol{K_C}$**:** Tenacidade à Fratura

$\boldsymbol{L}$**:** Comprimento

$\boldsymbol{N_i}$**:** Função Determinante do tipo de Elemento

$\boldsymbol{P}$**:** Pressão

$\boldsymbol{r}$**:** Raio

$\boldsymbol{t}$**:** Espessura

$\boldsymbol{T}$**:** Tensão

$\boldsymbol{U}$**:** Energia Elástica

$\boldsymbol{W}$**:** Trabalho

Resumo

Um dos métodos mais eficazes de análise de tensões em equipamentos é o Método dos Elementos Finitos (conhecido como FEM ou MEF). Este método consiste na aproximação da geometria por vários elementos inifinitesimais que formam uma malha. A tensão é analisada nó a nó e então tratada de maneira individual. O método se torna mais eficaz quando há uma malha com o máximo possível de elementos finitos. Este tipo de técnica é muito utilizado na construção civil, onde é necessária a simulação de vigas para construção de pontes, torres e monumentos, para tanto, utiliza-se softwares de apoio como Ansys e Solidworks, permitindo verificar as tensões e deformações do corpo submetido às condições especificadas. A linguagem de programação Python além de simples e fácil de aprender, é portátil e extensível, bem como costuma ser utilizada como base para a computação gráfica. Com esta ferramenta é possível otimizar a quantidade de linhas de códigos, que além de organizadas por identação, possuem verificação de erros automática. A utilização de matrizes de rigidez, deformação e do tensor de Cauchy é imprescindível quando se trata de programar um simulador de tensões, visto que a forma matricial permite uma leitura dinâmica no ato do processamento dos dados. Para a geração das malhas necessárias foi utilizada a ferramenta Gmsh em sua versão 3.0.0, a qual permite rotular nós e converter malhas triangulares em quadriculadas. Com o Gmsh também é possível atualizar em tempo real a quantidade de cores utilizada em um espectro de tensões, provando sua eficácia como auxiliar na resposta visual às tensões em uma viga em flexão. O trabalho consistiu em adequar um software de análise mecânica criado por Nasser Alkmim em 2016 para as condições necessárias de análise das geometrias estudadas. Ao final deste trabalho, deverão estar presentes as simulações em forma de espectros de tensão (gráfico) e documentos com as tensões em cada um dos nós, permitindo a identificação de pontos de tensão máxima e mínima, assim como a linha neutra. Comparando os resultados obtidos no trabalho com uma simulação feita no software Ansys, notou-se que há semelhança em sua apresentação visual, além de provar que o tempo de simulação em Python foi menor que no software profissional.

Abstract

One of the most effective methods of stress analysis in equipment is the Finite Element Method (known as FEM or MEF). This method consists in the approximation of the geometry by several elements inifinitesimals that form a mesh. The voltage is analyzed node by node and then treated individually. The method becomes more effective when there is a mesh with the maximum possible finite elements. This type of technique is widely used in civil construction, where it is necessary to simulate beams for the construction of bridges, towers and monuments, using support software such as Ansys and Solidworks, allowing to verify the tensions and deformations of the submitted body conditions. The Python programming language is simple and easy to learn, it is portable and extensible, and it is often used as the basis for computer graphics. With this tool it is possible to optimize the number of lines of code, which besides being organized by indentation, have automatic error checking. The use of stiffness, strain and Cauchy tensor matrices is essential when programming a voltage simulator, since the matrix form allows a dynamic reading in the act of data processing. For the generation of the necessary meshes the tool Gmsh was used in its version 3.0.0, which allows to label nodes and to convert triangular meshes into squares. With Gmsh it is also possible to update in real time the amount of color used in a voltage spectrum, proving its effectiveness as an aid in the visual response to the stresses in a bending beam. Thus, the present work began to be elaborated, with the objective of simulating the stresses applied to the walls of both configurations (smooth and parabolic). The work consisted in adapting a software of mechanical analysis created by Nasser Alkmim in 2016 for the necessary conditions of analysis of the studied geometries. At the end of this work, the simulations should be present in the form of voltage (graph) spectra and documents with the voltages at each node, allowing the identification of maximum and minimum voltage points, as well as the neutral line. Comparing the results obtained in the work with a simulation made in the software Ansys, it was noticed that there is similarity in its visual presentation, besides proving that the time of simulation in Python was smaller than in the professional software.

Agradecimentos

Agradeço por ser minoria, por ser diferente, por às vezes – e quase sempre – receber críticas, elas me fortalecem! Fazem de mim alguém mais resistente e emocionalmente mais seguro. Tenho gratidão por toda a natureza que interage comigo, pelo ar respirado, pela água bebida, pelo solo que caminho e por tem o dom dos cinco sentidos (e mais alguns).

Meu orientador, Vânio, me acompanhou desde o começo da graduação na engenharia mecânica e viu meus entraves, dificuldades, e ainda assim enxergou possibilidades nas coisas que pareciam impossíveis. Agradeço a ele do fundo do coração por depositar confiança em mim e aceitar enfrentar comigo um trabalho tão árduo, embora recompensador.

Agradeço aos meus pais, Marcus e Leila, e minha avó, Dadai, por me darem suporte e serem as brisas nas minhas asas e ao mesmo tempo o meu lugar de retorno. Ter uma filha transgênero mexe com as estruturas fincadas pelos moldes sociais, mas o choro de alegria do meu pai, a naturalidade da minha mãe e a preocupação da minha avó, mesmo em sua ignorância, foram essenciais para que eu me sentisse bem comigo.

E agradeço também a Janine, minha psicóloga, que abriu as portas do seu consultório para me receber com muito amor e profissionalismo. Três pessoas que quero levar comigo para sempre, três anjos, três amores.

籠の中の空は狭過ぎるだろ

Flügel der Freiheit

Capítulo 1: Introdução

Os aspectos gerais do trabalho, como sua contextualização, justificativa e seus objetivos gerais e específicos serão abordados no presente trabalho, o qual consiste na apresentação de um projeto desenvolvido pelo professor Me. Vânio Vicente Santos de Souza, docente no curso Bacharelado em Engenharia Mecânica da Universidade Federal do Recôncavo da Bahia. O objetivo central do projeto é a simulação computacional através de uma adaptação de um software de análise de elasticidade, para a análise de tensão em corpos de prova específicos.

A adaptação de um Software capaz de realizar a Análise de Elementos Finitos (AEM) para aplicações específicas se torna um facilitador, visto que a utilização de uma ferramenta mais potente e generalista requer máquinas com maior capacidade de processamento e necessita da modelagem prévia da peça a ser analisada. A linguagem de programação Python foi escolhida para a aplicação do trabalho devido seu caráter de código aberto e a organização das linhas de comando via identação, facilitando a identificação de classes e a detecção de incoerências.

A identificação de tensões em um problema físico através do ambiente virtual promove segurança não somente do local de trabalho, mas também garante que não haja gastos desnecessários ao projetar uma peça ou equipamento. Tensões importantes, como a tensão de ruptura, e entidades notáveis, como a linha neutra, podem ser identificadas de maneira simplificada a depender da plataforma virtual utilizada.

Softwares famosos como Ansys e Solidworks possuem diversas funcionalidades quando se trata de simulação de problemas físicos, no entanto, eles são conhecidos por seus códigos fechados e processamento demorado em máquinas de uso popular. Com esta problemática em mente, Alkmim (2016) criou um software livre em Python, o Elastopy, cuja função é identificar deformações em um sólido previamente modelado em uma plataforma simples e leve, o Gmsh. O resultado foi um programa capaz de identificar em segundos as deformações em cada nó e a posição de cada um deles após a aplicação de forças.

Utilizando como base o software Elastopy, o presente trabalho começou a ser realizado, adaptando as linhas de código para que identificassem as tensões em duas geometrias específicas (aproximadamente retangular e parabólica), com cargas diferentes em vários ensaios, podendo assim identificar tensões em pontos de interesse, bem como observar a posição da linha neutra, percebendo as zonas de tração e compressão. A adaptação também consistiu em gerar um documento de texto contendo as tensões em cada nó.

Um comparativo com o mesmo problema em simulação pelo software Ansys foi realizada para confirmar a confiabilidade dos resultados obtidos no trabalho. Esta comparação utilizou dois critérios, a proximidade dos valores de tensão nos mesmos nós em ambos softwares e a similaridade dos espectros de tensão em cada caso.

1.3. Objetivos

1.3.1 Objetivos Gerais

Gerar um software para a análise de tensões de duas geometrias, sendo a primeira uma viga bi apoiada e a segunda um modelo parabólico, verificando as tensões elementares através do Método dos Elementos Finitos (FEM) possuindo como suporte a linguagem de programação Python e a ferramenta Gmsh para a geração de malhas.

1.3.2. Objetivos Específicos

- Estudar o Método dos Elementos Finitos, verificando quais são os parâmetros e equações necessárias para a construção de um sistema para a análise do caso específico de uma parede de Vaso de Pressão;
- Identificar a problemática e construir os modelos geométricos e malhas a partir de condições de contorno;
- Gerar ao final um espectro de cores relacionadas à tensão em cada elemento finito;
- Gerar juntamente com o espectro de tensões, um arquivo de texto contendo os valores de tensão em cada nó;
- Utilizar o modelo Quad4 de malha (quadrangular) ao invés do modelo triangular usual;

Capítulo 2: Elementos Finitos

Este capítulo possui como objetivo a definição de elemento finito, fornecendo as bases necessárias para que seja possível a construção de algoritmos solucionadores do problema geométrico deste trabalho.

2.1. Teoria dos Elementos Finitos

Dentre os incontáveis problemas da física e engenharia, está presente em diversos casos a famosa Equação de Poisson, e a depender da geometria do domínio no problema (normalmente retangulares ou circulares), é possível obter uma solução analítica através de séries de Fourier. Porém, quando o domínio possui geometria mais elaborada, este método não se aplica de forma viável (GIACCHINI, 2012).

O Método dos Elementos Finitos (MEF) é uma maneira de encontrar solução numérica para domínios complexos, resolvendo equações diferenciais elemento a elemento, discretizando o domínio e, por aproximação geométrica, representando o elemento maior (Forma Forte) através de uma união de pequenos elementos (Forma Fraca) (GIACCHINI, 2012).

Para a utilização do MEF em casos de domínio bidimensional, os elementos costumam ser apresentados na forma de quadriláteros ou triângulos, podendo assim resolver problemas de geometrias não convencionais (Figura 1) (GIACCHINI, 2012).

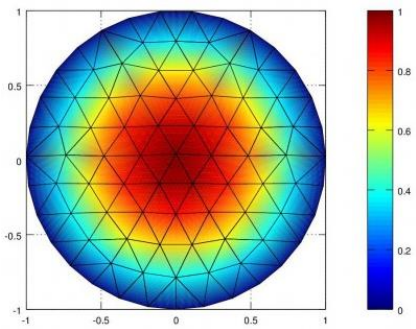


Figura 1: Elementos Finitos em uma Circunferência (Pic-c, 2012).

Para a utilização do MEF, é necessário verificar a natureza dimensional do problema, as condições as quais devem ser obedecidas e as propriedades físicas, então, tudo é traduzido matematicamente, apresentando três possíveis modelos: Forte, Fraco e Variacional, modelos estes equivalentes e que podem ser comparados utilizando o Cálculo Variacional (ALKMIM, 2016).

Para a Forma Forte, também conhecida como Forma Diferencial, o problema é representado por um sistema de equações diferenciais no tempo ou espaço, sendo estas equações parciais ou ordinárias, sendo necessária a especificação de condições de contorno. Este modelo matemático exige que as variáveis desconhecidas sejam especificadas no contorno através de uma condição inicial (Problema de Valor Inicial) (ALKMIM, 2016).

Na Forma forte costumam ser resolvidos problemas simples, como vigas de geometria simplificada, onde basta ser aplicada a técnica de integração apropriada para o caso. Assim, o problema de diferenciação se faz mais simples ao ser aproximado por equações algébricas associadas (ALKMIM, 2016).

Para a Forma Fraca, o problema é resolvido através de uma integral ponderada no domínio, sendo assim, a análise feita ponto a ponto na Forma Forte é substituída por uma análise de média. Utilizando esta técnica, os requerimentos nas condições de contorno se tornam mais fracos (de onde vem o nome do modelo), permitindo a satisfação das condições exigidas (ALKMIM, 2016).

Muitas vezes a Forma Variacional está incluída na Forma Fraca, no entanto, este modelo possui uma característica de tornar funções extremas em formas de integrais. A Energia Potencial Total é um exemplo que é melhor calculado utilizando o modelo matemático Variacional, visto que é um problema de equilíbrio estático e seu valor mínimo indica equilíbrio (RAO, 2013).

A equação abaixo representa uma integral para a Energia Potencial Total dependente de uma variável de campo $u_i(x_i)$, que indica uma multivariável vetorial.

$$\Pi = \int_{\Omega} F\left(u_i, \frac{\partial u_i}{\partial x_i}, \frac{\partial^{2u_i}}{\partial^{2x_i}}, \ldots\right) d\Omega \qquad (10)$$

Dentre as vantagens da utilização da Forma Variável, são destacáveis:

- A unificação das propriedades do problema (condições de contorno de Neumann ou derivadas do campo e leis de conservação);
- A utilização de valores escalares ao invés de vetoriais na função;
- Não é necessária a satisfação das condições de contorno naturais, pois estão implícitas na formulação.

Para a forma variacional tem-se então:

$$\begin{cases} \Pi = \int_0^L \frac{1}{2} EA \left(\frac{du}{dx}\right)^2 dx - \int_0^L ub\, dx - Ru(L) \\ \delta\Pi = 0 \\ u(0) = 0 \\ \delta u(0) = 0 \end{cases} \quad (11)$$

2.2. Diagrama de Tonti

Este modelo foi criado pelo matemático e físico Enzo Tonti e consiste em um diagrama de classificação, expondo as relações entre as variáveis e os elementos espaciais a elas relacionados. O método busca no espaço as variáveis físicas e as características espaciais que as unem (FARIA, 2005).

No diagrama, as variáveis de configuração ficam do lado esquerdo, por outro lado, o espaço à direita é reservado para as variáveis de fonte. A Figura 2 ilustra uma aplicação deste modelo, sendo todas as variáveis de campo submetidas às equações de campo (TONTI, 2013).

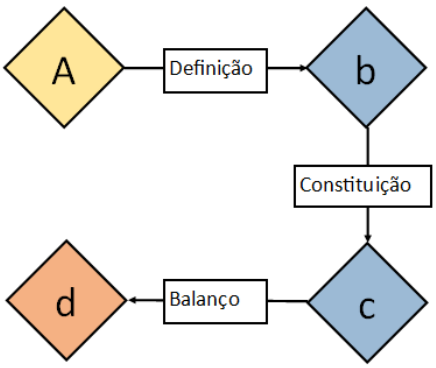


Figura 2: Modelo do Diagrama de Tonti (Autor, 2019).

Nesta figura, A exemplifica uma variável associada a um ponto (consequentemente, possui densidade própria), enquanto b é associada a um elemento de linha (sendo um vetor de linha, com integral resultando em B), c é associada a uma superfície e sua integral de área retorna C, por fim, d é associada a um volume cuja integral retorna o valor D. Apenas d é uma variável conhecida, enquanto a definição, o balanço e a constituição são equações de campo (FELIPPA, 2004).

Embora o diagrama de Tonti seja muito útil para organizar as variáveis de acordo com critérios, Felippa (2004) apresentou um diagrama destrinchando as variáveis e condições de contorno de forma generalizada (Figura 3).

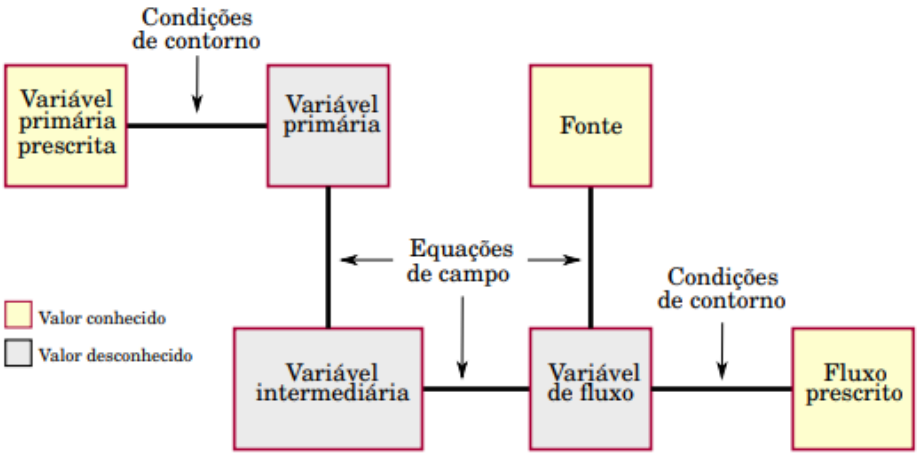


Figura 3: Diagrama de Tonti para condições de contorno (FELIPPA, 2004).

As classificações entre os tipos de variáveis as categorizam em globais, de campo, de fonte e de energia, sendo estes tipos baseados na densidade, cuja definição vem da razão entre a variável e um elemento espacial, podendo ser uma linha, uma superfície ou um volume (TONTI, 2013).

Variáveis globais são aquelas que quando geram densidade, possuem sua geometria e propriedades físicas ocultas. Variáveis representantes desta categoria costumam ser a densidade de Massa (razão entre Massa e Volume) e Pressão (razão entre Força e Área). Este tipo de conexão entre as variáveis globais e o elemento espaço-temporal (pontos, linhas, superfícies e volumes) é favorável para a física computacional, uma vez que permite a discretização através de argumentos físicos (ALKMIM, 2016).

Variáveis de Fonte são aquelas que funcionam como fontes de campos, por exemplo, temperatura para calor e força para deslocamento.

Variáveis de Configuração são aquelas que definem as configurações de todo o sistema, por exemplo, a maneira com a qual a temperatura e o deslocamento traçam respectivamente os campos de configuração de determinado sólido e os campos de temperatura.

Variáveis de Energia são geradas através do cruzamento de Variáveis de Fonte e de Configuração, naturalmente advêm de um produto (TONTI, 2014).

2.3. Relação de Equilíbrio

A lei da conservação de momento linear define a relação de equilíbrio de um corpo, sendo para o caso estático o fenômeno que resulta no balanço de forças sobre um ponto material ou volume infinitesimal. Segundo esta lei, a força pode ser dividida em Força de Superfície e Força de Corpo, bem como em Externa – oriundas de material externo ao corpo – e Interna – originadas no interior do corpo (RAO, 2013).

Para imersão na problemática de um corpo em equilíbrio, pode-se definir um corpo como uma estrutura hiperestática, onde cada ponto material está fortemente conectado com seus vizinhos. Esta consideração assume que o número de graus de liberdade é menor, requerendo menos incógnitas, forças internas e menos equações, embora pode-se compensar esta diferença introduzindo equações de compatibilidade cinemática (RAO, 2013).

Em um corpo podem haver duas maneiras de atuação de força, a primeira é distribuída pelo volume enquanto a segunda é distribuída pela superfície. Quando a força é aplicada na superfície, a resposta do corpo às solicitações de carregamento surge em forma de tensão, que é definida pelo limite da razão entre a força e a área, com o elemento de área tendendo a zero (RAO, 2013).

$$t = \lim_{A \to 0} \frac{T}{A}$$

Onde T é o vetor Força de Superfície resultante em uma área A.

As tensões são transmitidas em um corpo ponto a ponto e o mapeamento conhecido como estado de tensões de um ponto material pode ser descrito através de nove componentes de tensão na forma σ_{ij}, onde i representa a face na direção j, face esta definida pela normal (RAO, 2013).

A Figura 4 ilustra um elemento infinitesimal, o ponto material P, e suas componentes de tensão.

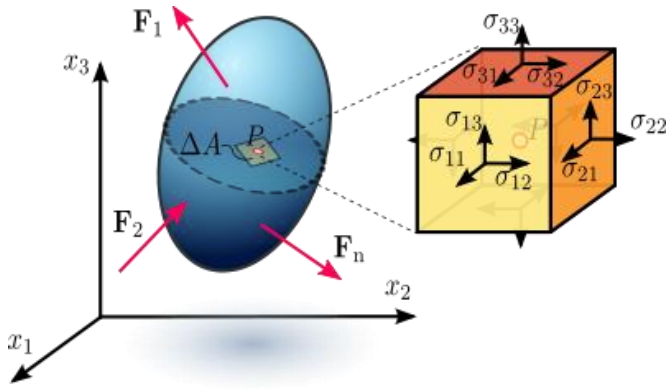


Figura 4: Componentes de Tensão (PILKEY, 1974).

A matriz gerada pelas componentes apresenta uma forma mais completa do estado de tensões. Este modelo é conhecido como Tensor de Tensão de Cauchy.

$$\underline{\sigma} \equiv \sigma_{ij} \equiv \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \equiv \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \qquad (12)$$

O equilíbrio de momento em um dos pontos do elemento concebe as relações abaixo.

$$\sigma_{xy} = \sigma_{yx}$$

$$\sigma_{yz} = \sigma_{zy}$$

$$\sigma_{xz} = \sigma_{zx}$$

Fazendo estas considerações, é possível reduzir o número de componentes independentes de nove para seis, sendo conveniente representá-los através de um vetor.

$$\sigma = [\sigma_{11} \ \sigma_{22} \ \sigma_{33} \ \sigma_{12} \ \sigma_{13} \ \sigma_{23}]$$

O equilíbrio entre as forças que agem nas áreas de um elemento infinitesimal gera o equilíbrio no ponto material, o qual pode ser representado pelo divergente da Tensão (ALKMIM, 2016).

Quando é tomada, a direção $x_2 = y$, tem-se como atuantes as forças Normal e Cisalhante, assim sendo:

$$\left(\frac{\partial \sigma_{22}}{\partial x_2}\right)\Delta x\Delta z + \left(\frac{\partial \sigma_{12}}{\partial x_1}\right)\Delta y\Delta z + \left(\frac{\partial \sigma_{32}}{\partial x_3}\right)\Delta x\Delta y + b_2\Delta x\Delta y\Delta z = 0$$

As parcelas com derivadas parciais representam o termo inicial da série de Taylor, significando que o campo de tensões varia dentro do elemento infinitesimal proporcionalmente à taxa de variação do campo na direção vezes o tamanho do intervalo (ALKMIM, 2016).

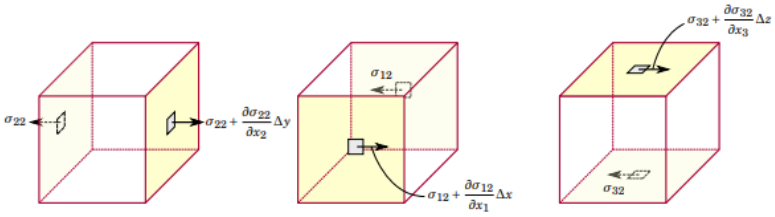


Figura 5: Parcelas infinitesimais para o campo de tensões (ALKMIM, 2016).

É possível expressar o balanço de forças através de uma notação indicial:

$$\sigma_{12,1} + \sigma_{22,2} + \sigma_{32,3} + b_2 = 0 \qquad (13)$$

Também podendo ser expressado para as direções $x_1 = x$ e $x_3 = z$ na forma indicial reduzida:

$$\sigma_{j1,j} + b_1 = 0$$

$$\sigma_{j2,j} + b_2 = 0$$

$$\sigma_{j3,j} + b_3 = 0$$

Assim é possível reduzir as três equações a uma expressão com duas formas:

$$\nabla \cdot \underline{\sigma} + b = 0$$

$$\sigma_{ji,j} + b_i = 0$$

2.4. Relação Cinemática

Relações cinemáticas são aquelas que definem a ligação entre deformações específicas e deslocamentos ponto a ponto. Utilizando variáveis de equilíbrio é possível resolver problemas de natureza mecânica, este tipo de parâmetro é obtido através dos dados do material (BEER, 2012).

Para a cinemática, existem duas configurações de deformação, as ditas não-deformadas (quando ainda não houve deformação) e as deformadas. Para a configuração não-deformada, o campo de deslocamento se apresenta de maneira nula, enquanto para o período de deformação e o estado plenamente deformado, o campo deslocamento é diferente de zero, admitindo assim uma aproximação linear, ignorando termos de segunda ordem para deslocamentos pequenos (BEER, 2012).

2.4.1. Medidas de Deformação

Os tensores são utilizados para medir as deformações em um corpo. Dentre eles é possível destacar Cauchy-Green, Venant e o tensor de deformação linear. Todos eles se baseiam em um mapeamento realizado antes e depois da deformação, avaliando o deslocamento de nós (TONTI, 2013).

Para problemas de deformação pequena, o tensor de deformação linear demonstra resultados aceitáveis. Para que seja realizada a medição da deformação específica é necessário calcular a razão entre a deformação no elemento e seu comprimento inicial, utilizando-se do limite (TONTI, 2013).

$$\varepsilon_{11} \stackrel{\text{def}}{=} \lim_{\Delta x \to 0} \frac{u_x(x + \Delta x, y) - u_x(x, y)}{\Delta x}$$

Para os planos x, y e z pode-se calcular a quantidade de deformação. Onde para a deformação com índices $i = j$, tem-se que o corpo deformou nas direções de elongamento enquanto para $i \neq j$, a deformação é cisalhante e provoca torção (ALKMIM, 2016).

$$\varepsilon_{11} \stackrel{\text{def}}{=} \frac{\partial u_x}{\partial x} \equiv \frac{\partial u_1}{\partial x_1}$$

$$\varepsilon_{22} \stackrel{\text{def}}{=} \frac{\partial u_x}{\partial x} \equiv \frac{\partial u_2}{\partial x_2}$$

$$\varepsilon_{33} \stackrel{\text{def}}{=} \frac{\partial u_x}{\partial x} \equiv \frac{\partial u_3}{\partial x_3}$$

Uma representação da deformação específica cisalhante é $\gamma_{ij} = \gamma_i + \gamma_j$, ela é avaliada através da razão da quantidade de distorção pelo comprimento Δy. Este fenômeno ocorre quando o corpo é deformado em uma direção ao percorrer uma distância ortogonal ao seu plano (ALKMIM, 2016).

É possível explicitar a relação de deformação específica cisalhante através da equação abaixo:

$$\gamma_{xy} \stackrel{\text{def}}{=} \lim_{\Delta x, \Delta y \to 0} \frac{u_x(x, y + \Delta y) - u_x(x, y)}{\Delta y} + \frac{u_y(x, y + \Delta x, y) - u_y(x, y)}{\Delta x}$$

Esta equação pode ser reescrita como:

$$\gamma_{xy} \stackrel{\text{def}}{=} \frac{\partial u_x}{\partial y} \equiv \frac{\partial u_y}{\partial x} \equiv \frac{\partial u_1}{\partial x_2} \equiv \frac{\partial u_2}{\partial x_1} \qquad (14)$$

Quando há pequenas deformações, ou seja, $\partial_{x1} u_1 \ll 1$ e $\partial_{x2} u_2 \ll 1$, e por causa das tangentes dos ângulos de γ_1 e γ_2 serem os próprios ângulos deles, $\tan\gamma \approx \gamma$, basta somar os ângulos para obter a deformação cisalhante.

Quando relaciona-se a deformação γ com a deformação ε, obtém-se uma relação que envolve o fator de correção $\frac{1}{2}$ para que os termos de ordem superior possam ser desprezados no tensor de deformação de Venant. Assim sendo, para $i \neq j$ (TONTI, 2013):

$$\varepsilon_{ij} = \frac{1}{2}\gamma_{ij}$$

Expondo as deformações normais e cisalhantes em uma matriz de deformação, é possível obter o tensor deformação linear:

$$\underline{\varepsilon} = \varepsilon_{ij} = \begin{bmatrix} \varepsilon_{11} & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{21} & \varepsilon_{22} & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & \varepsilon_{33} \end{bmatrix} = \begin{bmatrix} \varepsilon_{11} & \frac{1}{2}\gamma_{12} & \frac{1}{2}\gamma_{13} \\ \frac{1}{2}\gamma_{21} & \varepsilon_{22} & \frac{1}{2}\gamma_{23} \\ \frac{1}{2}\gamma_{31} & \frac{1}{2}\gamma_{32} & \varepsilon_{33} \end{bmatrix} \qquad (15)$$

2.5. Elasticidade

Os parâmetros e relações que descrevem o comportamento mecânico dos materiais são conhecidos como relações constitutivas. Estas características do material estão intimamente ligadas à relação tensão-deformação, dentre estes parâmetros, é prudente destacar a elasticidade, característica esta que aponta o quão suscetível a sofrer deformações reversíveis está um material (VAN VLACK, 2000).

Quando as tensões e as deformações se relacionam de forma linear, ocorre o que chama-se de caso Estático Linear, o que indica que ao sofrer pequenas deformações, um sólido nestas configurações possui um campo de deslocamentos gerado pela combinação linear dos elementos do tensor deformação. Para grandes deslocamentos as condições não são aplicadas (VAN VLACK, 2000).

Para atender as condições de linearidade, o material necessita obedecer a equação:

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl} \qquad (16)$$

A equação explicita uma relação de quarta ordem, uma vez que envolve os tensores de deformação e tensão, ambos de segunda ordem. C_{ijkl} caracteriza-se como tensor de rigidez elástica de quarta ordem (VAN VLACK, 2000).

2.6. Condições de Contorno

As restrições impostas ao domínio de um sistema de equações matemáticas relacionadas a um problema físico são conhecidas como condições de contorno. Para um problema mecânico, podem ser estabelecidas duas modalidades deste tipo de restrição, a condição de Dirichlet – na qual os deslocamentos são especificados– e a condição de Neumann – na qual as tensões são especificadas, ambas no contorno (FELIPPA, 2004).

Matematicamente, as condições de contorno para tensão e deslocamento podem ser explicitadas através das equações X e Y respectivamente.

$$u_i(x_i) = \bar{u}_i \qquad x_i \in \Gamma_u$$

$$\sigma_{ij}(n_j) = \bar{t}_i \qquad x_i \in \Gamma_t$$

Nestas equações é possível verificar os termos n_j (vetor unitário normal à superfície do corpo), Γ_u e Γ_t (partes da fronteira nas quais os deslocamentos e as tensões são definidos, respectivamente). As fronteiras específicas unidas formam a fronteira como um todo através da relação abaixo.

$$\Gamma_u \cup \Gamma_t$$

Para que o problema de elasticidade possa ser resolvido através do método dos elementos finitos, é necessário utilizar uma das suas equações na forma integral. Esta equação recebe um termo representando a função peso e graças a isso assume o valor médio dos subdomínios, o que torna uma equação forte em uma equação na forma variacional, a qual é possível retornar para a forma forte fazendo com que não haja variação inicial (FELIPPA, 2005).

A expressão variacional pode ser obtida através de uma definição de energia potencial, por isso sua parte funcional é conhecida como Energia Potencial Total, enquanto seu variacional é chamado de Princípio da Minimização da Energia Potencial. Para que seja obtida a forma variacional faz-se necessário enxergar a energia potencial como parte da deformação elástica e parte sendo o trabalho realizado por forças externas (FELIPPA, 2005).

Para que seja obtida a forma variacional é necessário selecionar os campos de interesse, os quais são a tensão e a deformação, além do deslocamento. É necessário também escolher as equações de campo fracas, uma vez que este tipo de equação utiliza a média dos pontos relacionados ao peso. Constrói-se então a forma fraca selecionando os pesos para as ligações fracas e os integrando com relação ao domínio. Neste ponto, toma-se como verdade que a variação de um parâmetro físico pode ser interpretado como o peso, e assim, identifica-se o funcional para que seja feita a análise da sua minimização, que deve retornar para uma forma diferencial (FELIPPA, 2005).

Para a seleção de variável primária, foi selecionada a variável deslocamento. Para a seleção das variáveis fracas, as equações de equilíbrio. É possível verificar o modelo de equações fracas na Figura 6.

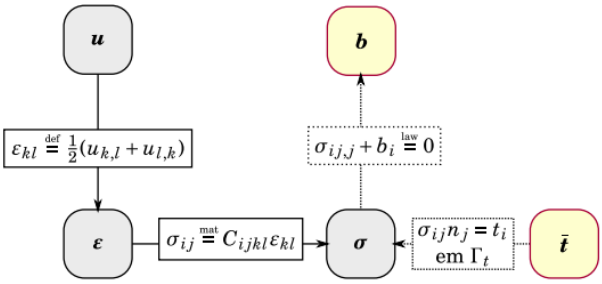


Figura 6: Diagrama de Tonti aplicado ao problema de elasticidade (FELIPPA, 2005).

Integrando no domínio obtém-se:

$$\int_{\Omega} (\sigma_{ij,j} + b_i) v_i d\Omega = 0$$

Após a integração e aplicação dos passos citados acima, obtém-se a expressão 17 correspondente à densidade de energia de deformação elástica, a qual pode ser reduzida graças à linearidade do problema.

$$\mathcal{U} \overset{\text{def}}{=} \frac{1}{2}\sigma_{ij}\varepsilon_{ij} \equiv \frac{1}{2}\sigma : \varepsilon \qquad (17)$$

2.7. Equações na Forma Matricial

Para que seja possível simplificar o pós-processamento do software, optou-se por utilizar o caso de elasticidade linear plana (como assumido na seção anterior). Para tanto, deverão ser escritos os tensores de Cauchy em sua forma matricial, como demonstrado abaixo.

$$\sigma = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix}$$

$$\varepsilon = \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{bmatrix}$$

O modelo de elasticidade linear em sua forma matricial deve utilizar as equações do diagrama da figura 7.

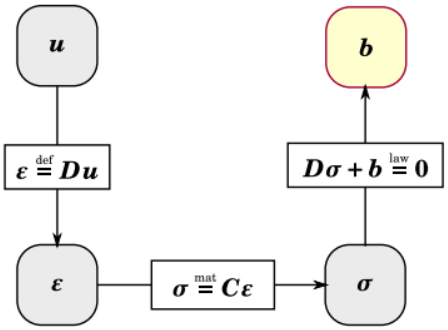


Figura 7: Forma matricial do modelo de elasticidade linear no Diagrama de Tonti (FELIPPA, 2004).

Em suas formas matriciais tem-se:

$$\varepsilon = Du \rightarrow \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} \partial_x & 0 \\ 0 & \partial_y \\ \partial_y & \partial_x \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix} \quad (18)$$

$$\sigma = C\varepsilon = \rightarrow \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu^2}{2} \end{bmatrix} \quad (19)$$

$$D^T\sigma + b = 0 \rightarrow \begin{bmatrix} \partial_x & 0 & \partial_y \\ 0 & \partial_y & \partial_x \end{bmatrix} \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (20)$$

2.8. Discretização do Problema

A discretização de um problema possui o intuito de limitar o domínio ao reduzir o número de graus de liberdade. A aproximação de uma geometria consiste na escolha de funções interpoladoras e o tipo de elemento que será utilizado, sendo muitas vezes escolhidos os elementos triangulares ou quadrangulares. A figura 8 demonstra que a quantidades de elementos na qual uma geometria é dividida faz com que ele se aproxime mais ou menos da geometria real (ALKMIM, 2016).

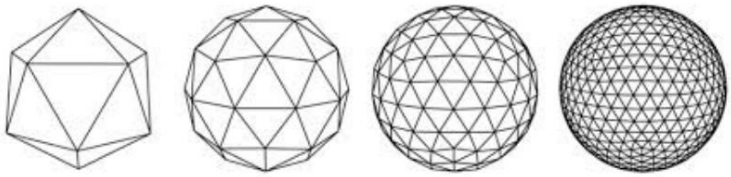


Figura 8: Interferência da quantidade de elementos na precisão geométrica de uma esfera (ALKMIM, 2016).

O software Elastopy (ALKMIM, 2016) utilizou como base o modelo de elemento finito quadrangular, também conhecido como quad4. Nele será definida uma função de aproximação para cada elemento, que pode ser generalizada pela equação 21.

$$u(x) = \begin{bmatrix} u_x(x) \\ u_y(x) \end{bmatrix} = \begin{bmatrix} N_1(x)\mu_{x1} + N_2(x)\mu_{x2} + N_3(x)\mu_{x3} + N_4(x)\mu_{x4} \\ N_1(x)\mu_{y1} + N_2(x)\mu_{y2} + N_3(x)\mu_{y3} + N_4(x)\mu_{y4} \end{bmatrix} \quad (21)$$

As incógnitas μ são os coeficientes de combinação linear, sendo os números de 1 a 4 os índices correspondentes a cada um dos vértices da forma quadrangular do elemento, também chamadas de nós. Como existem quatro componentes de dois graus de liberdade para cada elemento, totalizando tem-se oito graus de liberdade. N_i trata-se da função que determina o tipo de elemento, como um quad4 possui quatro nós, então $i = 1,2,3,4$. As funções possuem as características que as definem como lagrangianas, o que faz com que valham 1 em determinado ponto e sejam nulas nos demais. Para fácil visualização, a matriz da aproximação foi expressa na forma abaixo.

$$u(x) = \begin{bmatrix} N_1(x) & 0 & N_2(x) & 0 & N_3(x) & 0 & N_4(x) & 0 \\ 0 & N_1(x) & 0 & N_2(x) & 0 & N_3(x) & 0 & N_4(x) \end{bmatrix} \begin{bmatrix} \mu_{x1} \\ \mu_{y1} \\ \mu_{x2} \\ \mu_{y2} \\ \mu_{x3} \\ \mu_{y3} \\ \mu_{x4} \\ \mu_{y4} \end{bmatrix} = \boldsymbol{N\mu}$$

As funções que definem o espaço podem ser reescritas em coordenadas isoparamétricas e assumem a forma geral:

$$N_a(\xi) = \frac{1}{4}(1 + \xi_a\xi(1 + \eta_a\eta)$$

A mudança de coordenadas pode ser feita através de uma transformação $x = G(\xi) \equiv x(\xi)$:

$$x(\xi) = \begin{bmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{bmatrix} = \begin{bmatrix} x_1N_1(\xi,\eta) + x_2N_2(\xi,\eta) + x_3N_3(\xi,\eta) + x_4N_4(\xi,\eta) \\ y_1N_1(\xi,\eta) + y_2N_2(\xi,\eta) + y_3N_3(\xi,\eta) + y_4N_4(\xi,\eta) \end{bmatrix}$$

Agora que é de conhecimento a matriz transformação, é possível aplicar o determinante na matriz Jacobiana formada por ela, sendo obtida:

$$J = \left(\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}\right) = \det\begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \qquad (22)$$

O Jacobiano obtido corresponde à escala entre os elementos de área no sistema de coordenadas, isto significa que em uma transformação, seu valor se relaciona diretamente com a área do espaço isoparamétrico.

2.9. Matriz Rigidez e Vetores Carregamento

Para que vetores e matrizes possam ser representados no sistema adotado, é de grande utilidade definir a função generalizada do elemento, para tanto faz-se uso das definições de relação constitutiva e cinemática.

Em termos de deslocamento, tem-se:

$$\int_{\Omega^e} \delta\varepsilon^T CDN\mu d\Omega^e - \int_{\Omega^e} \delta u^T b d\Omega^e - \int_{\Gamma^e} \delta u^T \bar{t} d\Gamma^e = 0$$

Aplicando os cálculos vistos em (ALKMIM, 2016), é possível obter a expressão que representa os deslocamentos unitários virtuais:

$$\int_{\Omega^e} B^T CBud\Omega^e - \int_{\Omega^e} N^T bd\Omega^e - \int_{\Gamma^e} N^T \bar{t} d\Gamma^e = 0 \qquad (26)$$

A matriz Rigidez de um elemento pode ser retirada da primeira parcela da equação acima (Equação 24), enquanto a segunda parcela fornece o vetor carregamento devido às forças do corpo (Equação 25) e a última parcela indica o vetor carregamento devido às tensões impostas e deformações iniciais (Equação 26).

$$K = \sum_{\Gamma}^{2} \sum_{S}^{2} B^T(\xi_\Gamma, \eta_S) CB(\xi_\Gamma, \eta_S) J(\xi_\Gamma, \eta_S) \quad (24)$$

$$P_C = \sum_{\Gamma}^{2} \sum_{S}^{2} N^T(\xi_\Gamma, \eta_S) bJ(\xi_\Gamma, \eta_S) \quad (25)$$

$$P_e = \sum_{\Gamma}^{2} \sum_{S}^{2} B^T(\xi_\Gamma, \eta_S) C\varepsilon_0 J(\xi_\Gamma, \eta_S) \quad (26)$$

Capítulo 3: Computação Gráfica

Durante este capítulo serão abordados termos relacionados à computação e sua associação com o estudo da mecânica.

Tomando por base a formulação da geometria por Euclides, o sistema de coordenadas desenvolvido por Descartes e as notações matriciais de Sylvester, a computação gráfica surgiu da necessidade militar estadunidense no período da segunda guerra mundial de simular voos e criar um sistema de defesa aéreo, os respectivos projetos Whirlwind e SAGE (COSTA, 2002).

A partir das noções de geometria e do avanço do desenho técnico, a análise mecânica de sólidos e fluidos deixou de ser um trabalho puramente manual para se aliar à tecnologia. Com um conjunto de coordenadas e condições de contorno, é possível descobrir tensões de ruptura, elongamentos, vorticidades e outros fatores que até então eram observáveis apenas em ensaios reais (HELLMEISTER, 2010).

As ferramentas computacionais atuais utilizam o Método dos Elementos Finitos para realizarem análises de esforços, softwares CAD são utilizados como forças de projetar em uma faixa de segurança ou mesmo para realizar projetos de engenharia reversa. A utilização do MEF na análise mecânica reduz o custo de projeto ao evitar a fabricação de peças de testes para ensaios prematuros, em especial aqueles de característica destrutiva (BORGES, 2014).

Costumam haver três etapas na aplicação da computação gráfica em um projeto de engenharia. São elas as fases CAM, CAD e CAE (Figura 9).

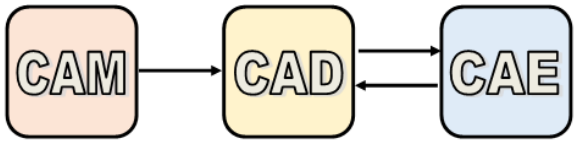


Figura 9: Etapas da Computação Gráfica Aplicada à Engenharia (Autor, 2019).

Na fase CAD é criado o modelo tridimensional, obedecendo tolerâncias pré-estabelecidas e definindo os materiais para a construção do modelo real.

Após a fase CAD, a etapa CAM (Computer-Aided Manufaturing) pode ser iniciada em paralelo com a fase CAE. Nela são fabricados os protótipos para ensaios mecânicos reais com corpos de prova que podem ter custo mais baixo que o produto final (BORGES, 2014).

A fase CAE (Computer-Aided Engineering) acontece em paralelo com a fase CAM. Aqui, toda a física é testada em condições virtuais, as quais podem variar desde exposição à vibração até fenômenos climáticos, esforços de tração, compressão, vácuo, turbulência, e muitos outros. Durante a fase CAE, são identificadas 3 etapas:

- Pré-processamento: Definição da geometria do modelo, condições de contorno, propriedades intrínsecas do material (módulo de elasticidade, rigidez, coeficiente de Poisson e outros);
- Processamento: Imposição de carregamentos (módulo, direção e sentido) e condições posicionais sob as quais a peça ensaiada está submetida (esgastamento, pino, balanço, etc);
- Pós-Processamento: Interpretação numérica da solução gerada pela análise computacional das duas fases anteriores. Esta fase fornece dados de saída (deformações, deslocamentos, tensões, respostas à vibração, etc).

Após a etapa CAE, o usuário deve retornar à fase CAD e reajustar seu modelo até que se adeque às condições necessárias para que a peça funcione em plenitude.

A utilização das três etapas da computação gráfica aplicada se mostra ainda mais útil quando trata-se de ensaios arriscados em corpos de provas caros, como é o caso de Vasos de Pressão. Graças às simulações, não há perda de vasos em testes hidrostáticos (destrutivos), o que reduz os custos e riscos de acidentes durante eventuais ensaios na fase de ensaios reais.

É importante ressaltar que a simulação de uma peça não isenta o fabricante de testá-la em ensaios mecânicos reais.

3.1. Softwares Proprietários

A Autodesk Inc lançou em 1982 a primeira versão do seu software CAD, o AutoCAD. Ao longo dos anos, o programa passou por aprimoramentos significativos até alcançar o ambiente de modelagem em três dimensões e a possibilidade de simular situações de natureza mecânica simples. Dentre as vantagens, está sua interface amigável e relativamente leve, facilitando os usuários novatos no ramo e a possibilidade de edição de dados através de linhas de comando, no entanto, para que as simulações sejam feitas, é necessária a

utilização do pacote de softwares complementares da empresa, como o MoldFlow, Inventor e Nastran, além de DLC (Downloadable Content) (Portal Educação, 2018).

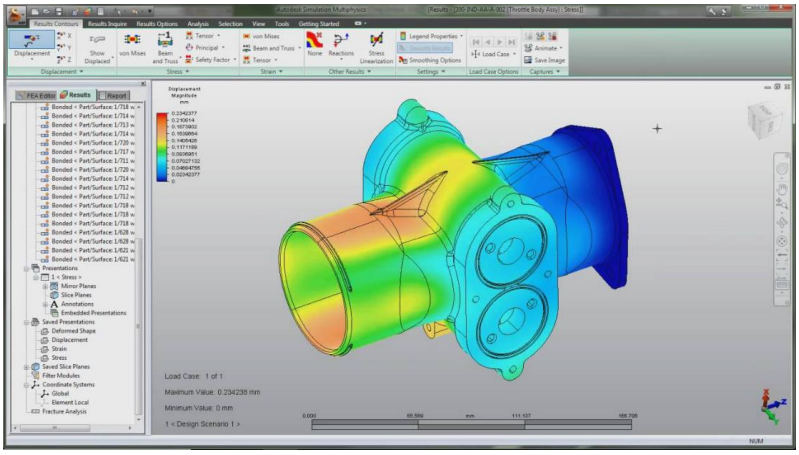


Figura 10: Tela de Simulação do Autodesk AutoCAD (3DPrint, 2016).

Com a intenção de criar um ambiente de modelagem tridimensional para o projeto de peças mecânicas, o Solidworks da SKA nasceu em 1993. O software apresenta um visual mais robusto, porém, muitas das funções são automatizadas, fazendo com que o usuário economize tempo e reduzindo seu trabalho e a possibilidade de falhas humanas. O programa também apresenta mais funções em relação ao produto da Autodesk, fornecendo até mesmo informações de projeto sem que o usuário construa o protótipo, como o peso. O tempo de processamento do Solidworks também é reduzido em relação ao Autocad (Portal Educação, 2018).

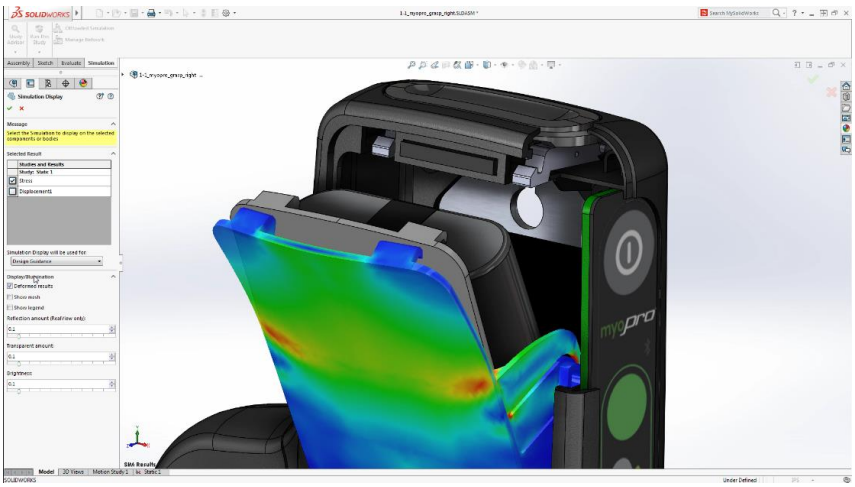


Figura 11: Tela de Simulação do SKA Solidworks (Solidworks, 2016).

O software Solid Edge, desenvolvido pela Siemens, foi criado com a intenção de facilitar a conversão de modelos 2D para 3D e a geração de páginas web, além de modelos para a criação de documentos técnicos através da extensão Solid Edge Insight. Embora seja um software com várias bibliotecas especializadas em simulação, o Solid Edge possui uma velocidade de processamento inferior em relação ao Solidworks e AutoCAD (Portal Educação, 2018).

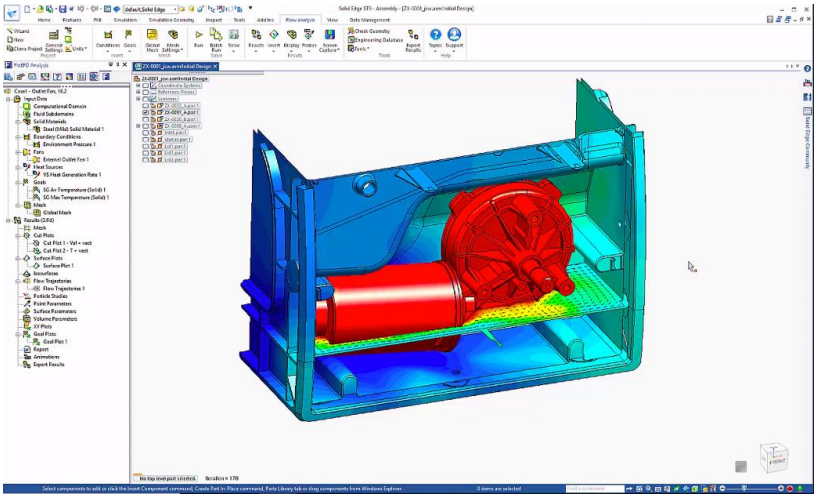


Figura 12: Tela de Simulação do Siemens Solid Edge (MGC, 2018).

Desenvolvido pela Mathworks, o software Matlab possui grande flexibilidade nas áreas de atuação (projetos de engenharia, química, comunicações, controle, entre outros). Sua velocidade de processamento é a maior dentre os softwares comparados, além de poder ser instalado em sistemas operacionais Windows, Linux e Mac. Com este software é possível coletar os dados de simulação de maneira agilizada e resolver problemas numéricos com auxílio de suas ferramentas integradas, além da possibilidade de associar os resultados ao Microsoft Excel (Engenharia Cotidiana, 2019).

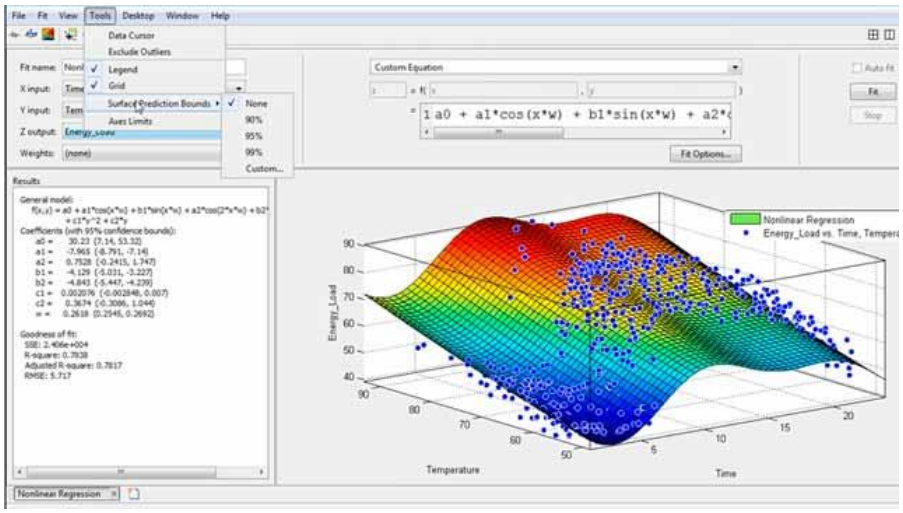


Figura 13: Tela de Simulação do Mathworks Matlab (Mathworks, 2018).

3.2. Softwares Livres

Desenvolvido pelo engenheiro civil e mestre em engenharia estrutural pela UnB, Nasser Alkmim, o software Elastopy foi criado para estudar problemas estáticos de natureza específica. O software utiliza como auxiliar o componente Gmsh para a geração de malhas. A velocidade média de processamento é relativamente rápida, competindo com o Matlab da Mathworks. Com este software é possível fazer análise de deformação de peças (ALKMIM, 2016).

O software Elastopy utiliza o método dos elementos finitos com elementos do tipo Quad4 (geometria quadrangular), foi utilizado como base para a criação de um outro software do mesmo autor, o Scikit, o qual ainda está em fase de desenvolvimento.

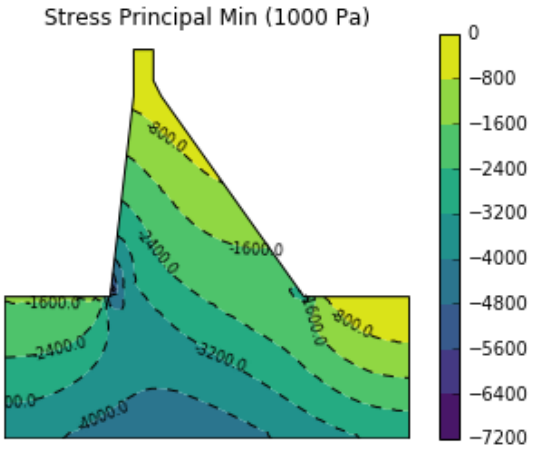


Figura 14: Saída do Elastopy (ALKMIM, 2016).

3.3. Python como Linguagem

Sendo a terceira linguagem de programação mais utilizada, perdendo apenas para as linguagens C e Java, o Python apresenta como vantagens uma sintaxe simples e códigos identados (facilitando a organização), além de uma coleção de bibliotecas muito grande. Por ser uma linguagem de programação aberta, muitos fóruns disponibilizam tutoriais para leigos e profissionais (Sempre, 2018).

O Python possui ferramentas voltadas à análise de problemas físicos e matemáticos através das bibliotecas Numpy e Networkx, bem como um pacote visual, o Matplotlib. Graças à sua popularidade, a linguagem Python possui ambientes repletos de softwares, bibliotecas e exemplos em plataformas como o PyPi e GitHub (Sempre, 2018).

A empresa de coleta de dados relacionados à tecnologia de informação, Imperva, realizou uma pesquisa em 2017 sobre a popularidade das linguagens de programação em ambientes de mercado e universidades (Figura 15). Segundo a pesquisa, o Python superou linguagens famosas, como SQL, Ruby, C++ e Matlab (Sempre, 2018).

Set/2018	Set/2017	Linguagem	Aprovação
1	1	Java	17,438%
2	2	C	15,447%
3	5	Python	7,653%
4	3	C++	7,384%
5	8	Visual Basic .NET	5,308%

Figura 15: Popularidade de Linguagens de Programação em 2017 (Adaptado de Sempre, 2018).

Aplicativos famosos como YouTube, Google, Spotfy, Blender3D e Dropbox utilizam a linguagem Python em sua base, assim como empresas de computação gráficas como a Industrial Light & Magic (Star Wars, Harry Potter, Piratas do Caribe e Vingadores) utilizam a linguagem durante o processo produtivo de seus filmes (Oráculo, 2017).

Foi então tomada a decisão de espelhar-se no software Elastopy (ALKMIM, 2016) e na linguagem de programação Python para a elaboração de uma simulação de tensões em dois tipos de geometria apossando-se das vantagens do Python como linguagem de programação aberta e eficaz.

Capítulo 4: Metodologia

O objetivo deste capítulo é definir a organização do algoritmo e apresentar as principais partes do código, como módulos, bibliotecas utilizadas e arquivos gerados.

4.1. Diagrama de Fluxo

Para que seja possível detectar falhas e pontos de melhoria no software, é necessário identificar cada uma das etapas que devem ser feitas, desde a obtenção da geometria especificada e da malha até o cálculo e plotagem do gráfico de tensão. O software foi desenvolvido utilizando a IDE Pycharm 2018.1.3 x64 em Windows 10 atualizado pela última vez em Janeiro de 2019.

O fluxograma da Figura 16 indica cada uma das etapas no momento de execução do software.

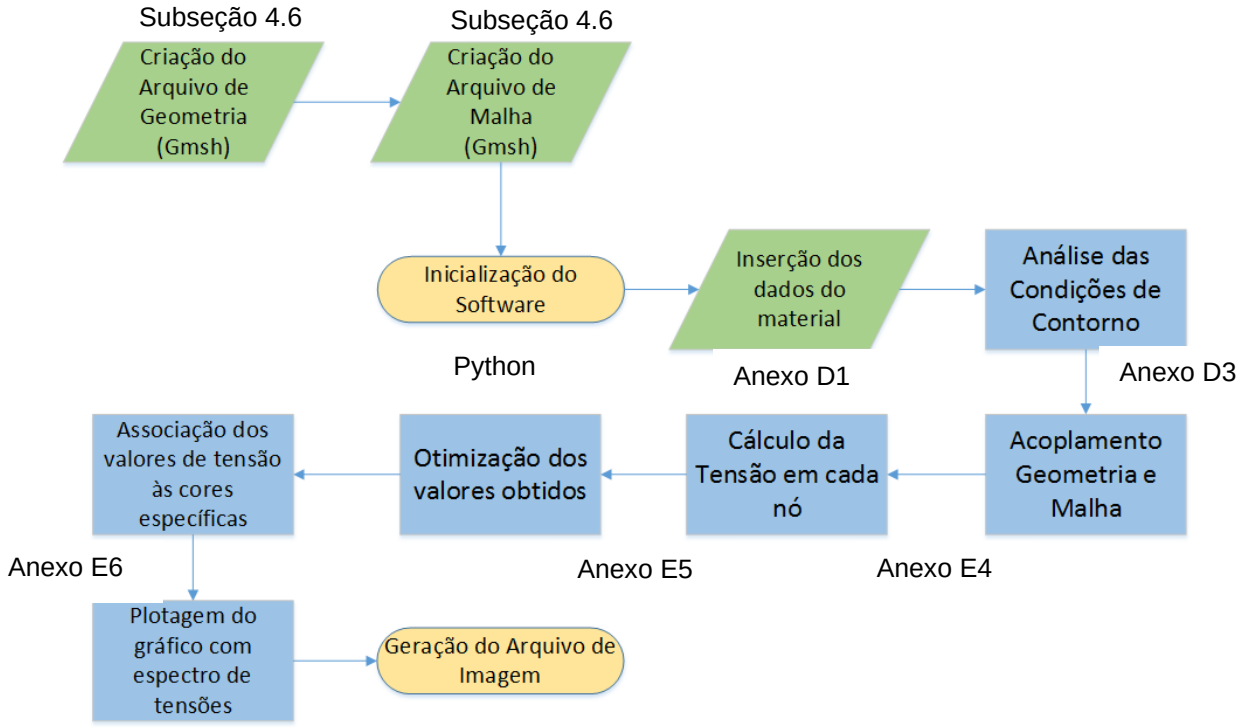


Figura 16: Diagrama de Fluxo da Solução (Autor, 2019).

4.2. Malha

A geometria e a malha foram feitas utilizando o software Gmsh em sua versão 3.0.6. Os códigos assumem as formas do quadro A e B para geometria e malha, tendo como saída arquivos nas extensões .geo e .msh. Estes arquivos fornecem coordenadas e endereça linhas, superfícies e nós com etiquetas (labels).

4.3. Entradas Específicas

As condições de contorno são analisadas assim que o software recebe e lê a geometria e a malha através dos arquivos gerados no gmsh para garantir que a fronteira é respeitada e que os vetores são suficientes para armazenar as informações. O software solicita como dados de entrada:

- Módulo de Elasticidade [MPa];
- Coeficiente de Poisson;
- Direção e Módulo da Força Aplicada [N];
- Ponto de aplicação da Força (caso concentrada);

- Superfície de aplicação da Força.

4.3. Acoplamento Geometria x Malha

Ao verificar que o problema pode ser lido pelo software, a malha e a geometria são unidas para que o arquivo de imagem seja gerado.

4.4. Análise de Tensão

Em paralelo ao acoplamento da geometria com a malha, é feita a análise da tensão elemento a elemento, e baseado no valor da tensão apresentado no elemento, uma etiqueta correspondente é atribuída, possibilitando a plotagem com diferença de cores.

4.5. Módulos

Em Python é necessário dividir algoritmos mais complexos em módulos, para que as funções a eles atribuídas possam ser acessados sem que haja necessidade de executar todo o programa. Por exemplo, caso o software precise calcular a tensão em um nó, o módulo responsável pelo cálculo de tensões será acessado e atuará. Caso não houvesse um esquema modular, todo o software seria executado sempre que uma determinada tarefa fosse requerida, gerando carregamento desnecessário. Baseado no software Elastopy do desenvolvedor Nasser Alkmim, foram utilizados os seguintes módulos:

- **Fronteira:** Responsável por verificar as condições de fronteira na geometria;
- **Construtor:** Constrói o elemento finito;
- **Elemento:** Aplica a física no elemento finito;

- **Material:** Definição do material, útil no momento de calcular as matrizes, especialmente a matriz rigidez;
- **Modelo:** Cria um modelo estrutural com os atributos definidos pela malha;
- **Marcador:** Cálculos das matrizes Rigidez, Massa, Pesos e Tração;
- **Tensão:** Cálculo da tensão aplicada ponto a ponto;
- **Quad4:** Definição da matriz aproximação para o cálculo dos elementos finitos do tipo quadrangular;
- **Gmsh:** Leitura e interpretação dos arquivos de geometria e malha;
- **Desenho:** Une a geometria e a malha lidas por Gmsh;
- **Plotagem:** Saída da imagem com os dados de eixos, legendas, cores e corpo do gráfico;
- **Estat:** Definições do problema estático.

4.6. Bibliotecas

Para que o algoritmo possa ser executado, é preciso utilizar bibliotecas que auxiliam os cálculos e interpretações. Algumas das bibliotecas utilizadas poupam o programador de precisar definir conceitos básicos até mesmo para problemas de natureza simples, como bases da física e matemática. As bibliotecas utilizadas foram:

- **Numpy:** Estabelece definições para cálculos de natureza discreta ou contínua;
- **Matplotlib:** Associa dados gerados numericamente a interfaces gráficas;
- **Networkx:** Define bases da mecânica estrutural.

4.6. Dados de Aplicação

4.6.1. Especificação do Problema

Para a simulação, a geometria especificada na Figura 17 (esquerda) foi utilizada, obedecendo as laterais retas de 8,9 cm, a base reta de 10 cm e a parte

superior uma parábola com vértice no ponto V(5 cm; 4 cm). Deve ser aplicada uma força variável e acumulada nos pontos R(0 cm; 8,9 cm) e S(10 cm; 8,9 cm). O material definido para a peça é o Acrílico Extrudado, cujas propriedades encontram-se no Anexo B, cujo módulo de elasticidade é igual a 3,22 GPa e possui coeficiente de Poisson de 0,22. Este foi sinalizado como Simulação A.

Para a simulação, a geometria especificada na Figura 17 (direita) foi utilizada, obedecendo as laterais retas de 9 cm, e as bases retas de 10 cm. Deve ser aplicada uma carga distribuída de 0,850 N ao longo da base superior. O material definido para a peça é o Acrílico Extrudado, cujas propriedades encontram-se no anexo B, cujo módulo de elasticidade é igual a 3,22 GPa e possui coeficiente de Poisson de 0,22. Este foi sinalizado como Simulação B.

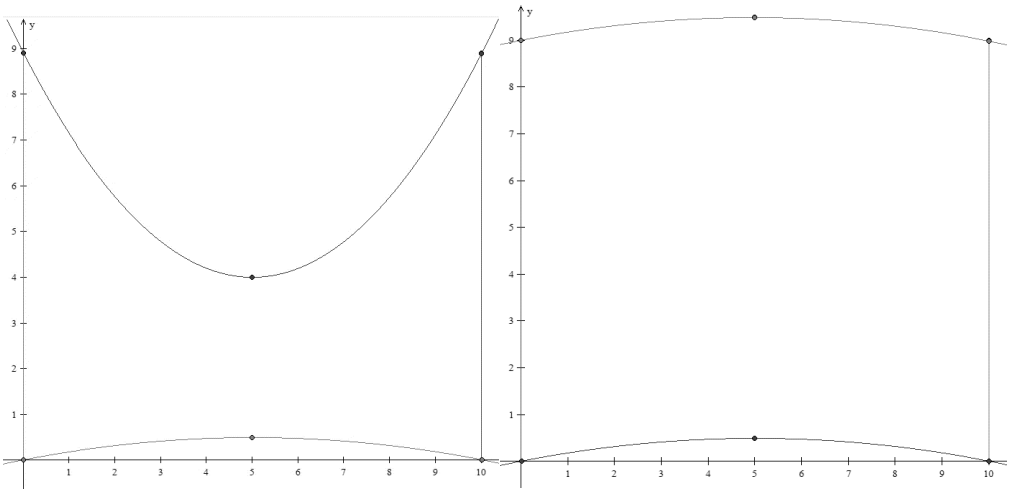


Figura 17: Geometria das Configurações A (esquerda) e B (direita) (Autor, 2019).

4.6.2. Arquivos de Geometria e Malha

Para a geração dos arquivos de Geometria e malha, a Figura 17 foi tomada como base. No entanto, ao utilizar o Gmsh, percebeu-se que era necessário aplicar ponto a ponto da parábola, então, para otimizar o problema, aproximou-se a parábola em pequenas retas, enquanto a base e as laterais permaneceram como retas únicas. Após feita a geometria, foram definidos no Gmsh todos os rótulos dos pontos para facilitar a organização do arquivo e então foi gerada a malha (Mesh).

4.6.2.1. Geração de Modelo para Simulação A

Para a criação da geometria parabólica, o modelo para ensaio desenvolvido por MELO, 2019 foi tomado como base. No entanto, sua reprodução a olho nu não poderia ser feita de maneira satisfatória utilizando o Gmsh, sendo assim, a partir de pontos específicos das parábolas, foram identificadas as duas funções correspondentes ao formato da parte superior e inferior do modelo.

Para a parábola superior, cuja concavidade é voltada para cima (Parábola A), a função descoberta foi $f_A(x) = 0{,}196x^2 - 1{,}968x + 8{,}900$. O mapa de pontos variando de $x = 0$ até $x = 10$ com intervalo de $0{,}5$ apresentou-se da seguinte forma (Tabela 1):

Tabela 1: Relação de Pontos da parábola A (Autor, 2019).

X	f(x)	x	f(x)
0,0	8,900	**5,5**	4,013
0,5	7,965	**6,0**	4,164
1,0	7,128	**6,5**	4,413
1,5	6,389	**7,0**	4,760
2,0	5,748	**7,5**	5,205
2,5	5,205	**8,0**	5,748
3,0	4,760	**8,5**	6,389
3,5	4,413	**9,0**	7,128
4,0	4,164	**9,5**	7,965
4,5	4,013	**10,0**	8,900
5,0	3,960	**5,5**	4,013

Para a segunda parábola, cuja concavidade é voltada para baixo (Parábola B), também foi estudada e gerou uma função, a qual se apresenta na forma: $f_A'(x) = -0{,}019x^2 + 0{,}200x - 0{,}003$. A partir da função descoberta, foi possível traçar os pontos correspondentes à geometria da parábola e construir a Tabela 2.

Tabela 2: Relação de Pontos da parábola B (Autor, 2019).

x	f(x)	x	f(x)
0,0	0,000	**5,5**	0,492
0,5	0,092	**6,0**	0,477
1,0	0,177	**6,5**	0,452
1,5	0,252	**7,0**	0,417
2,0	0,317	**7,5**	0,372
2,5	0,372	**8,0**	0,317
3,0	0,417	**8,5**	0,252
3,5	0,452	**9,0**	0,177
4,0	0,477	**9,5**	0,092
4,5	0,492	**10,0**	0,000
5,0	0,497	**5,5**	0,492

Assim, o arquivo de geometria pôde ser plotado no Gmsh (Figura 18), obedecendo as limitações geométricas e aproximando as suavizações para pequenas retas. Também foram criadas retas paralelas ao eixo y, ligando as duas parábolas, para que a superfície pudesse ser criada.

4.6.2.2. Geração de Modelo para Simulação B

Para criar o modelo da situação B, também foi preciso descobrir as funções que regem as curvaturas da geometria. Para a parábola superior (Parábola C), o mapa de pontos (Tabela 3) se deu pela função $f_B(x) = -0{,}02x^2 + 0{,}20x + 9{,}00$.

Tabela 3: Relação de Pontos da parábola C (Autor, 2019).

X	f(x)	X	f(x)
0,0	9,000	**5,5**	9,495
0,5	9,095	**6,0**	9,480
1,0	9,180	**6,5**	9,455
1,5	9,255	**7,0**	9,420
2,0	9,320	**7,5**	9,375
2,5	9,375	**8,0**	9,320
3,0	9,420	**8,5**	9,255
3,5	9,455	**9,0**	9,180
4,0	9,480	**9,5**	9,095
4,5	9,495	**10,0**	9,000

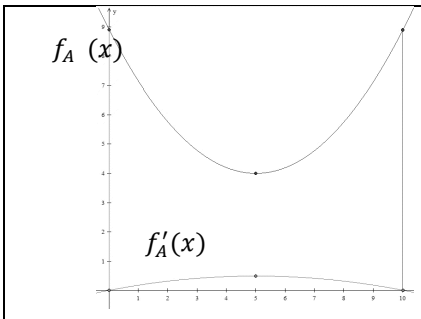
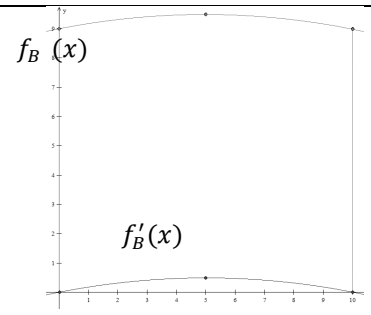
5,0	9,500	**5,5**	9,495

Para a parábola inferior (Parábola D), o mapa de pontos (Tabela 4) se deu pela função $f_B'(x) = -0{,}02x^2 + 0{,}20x$.

Tabela 4: Relação de Pontos da parábola D (Autor, 2019).

X	f(x)	X	f(x)
0,0	0,000	5,5	0,495
0,5	0,095	6,0	0,480
1,0	0,180	6,5	0,455
1,5	0,255	7,0	0,420
2,0	0,320	7,5	0,375
2,5	0,375	8,0	0,320
3,0	0,420	8,5	0,255
3,5	0,455	9,0	0,180
4,0	0,480	9,5	0,095
4,5	0,495	10,0	0,000
5,0	0,500	5,5	0,495

Foi plotado o arquivo de geometria e logo em seguida a malha (Figura 18) foi construída.

	
$f_A\ (x) = 0{,}196x^2 - 1{,}968x + 8{,}900$	$f_B\ (x) = -0{,}02x^2 + 0{,}20x + 9{,}00$
$f_A'(x) = -0{,}019x^2 + 0{,}200x - 0{,}003$	$f_B'(x) = -0{,}02x^2 + 0{,}20x$

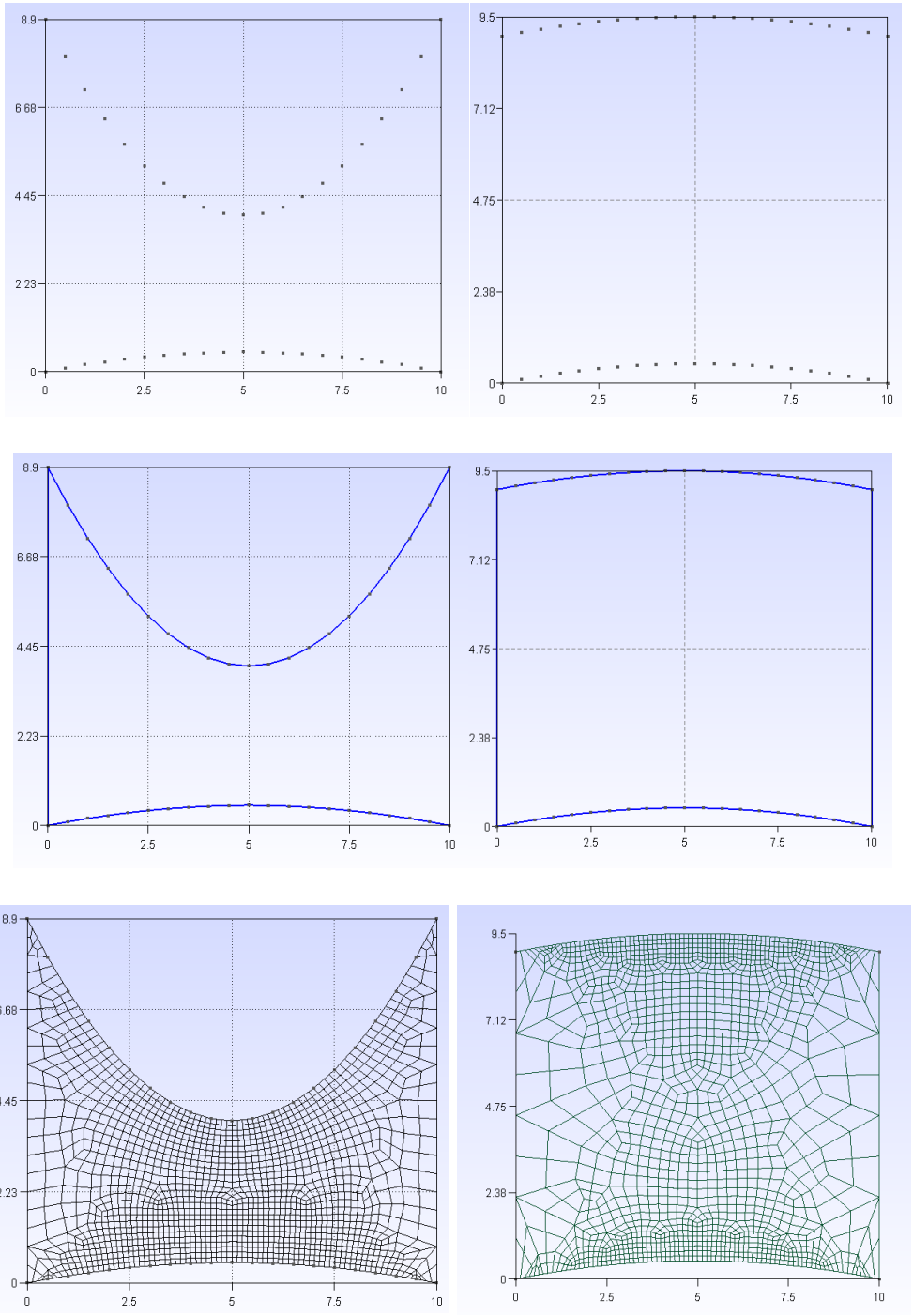


Figura 18: Malha preparada para a Simulação B. (Autor, 2019)

Durante a plotagem, verificou-se que não havia simetria nas malhas, então, percebeu-se que o Gmsh trabalha com duas opções: gerar elementos de malha que se acumulam e influenciam nos elementos seguintes ou verificar a simetria do problema geométrico e espelhar a malha a partir de um eixo de simetria. Neste caso, optou-se pelo eixo de simetria y de ponto (5;0) para que o

problema pudesse apresentar um espectro de tensão simétrico ao final da execução do programa.

4.6.3. Saída

Após a criação dos arquivos de geometria (.geo) e malha (.msh), ambos foram importados para a pasta onde localiza-se o arquivo executável do algoritmo e o programa foi executado (Anexo E1 e Anexo E2).

4.6.3.1. Simulação tipo A

A simulação do corpo de prova A foi realizada obedecendo os critérios e métodos feitos no ensaio de MELO, 2019. Sendo adicionadas cargas de maneira progressiva. A Tabela 5 mostra o valor de cada carga, sendo iniciada a simulação com a carga A e havendo a adição das demais cargas até que todas estejam sobre a superfície do corpo de prova, levou-se em consideração o apoio de 0,192 kg em cada ensaio. A Figura 19 representa a configuração de forças no problema.

Tabela 5: Dados das cargas para aplicação (Autor, 2019).

Bloco	Massa [kg]	Carga [N]	Carga Acumulada [N]
A	0,872	8,554	8,554
B	0,864	8,476	17,030
C	0,867	8,505	25,535
D	0,866	8,495	34,030
E	0,861	8,446	42,476

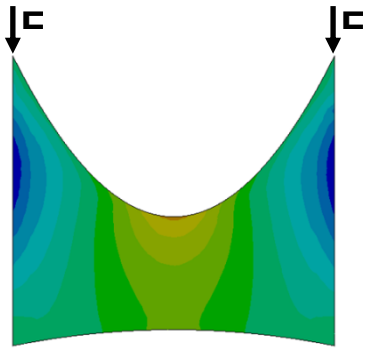


Figura 19: Configuração de Forças da Simulação A. (Autor, 2019)

Sem nenhum código de erro, o programa gerou um arquivo .msh, que ao ser aberto através do Gmsh, pôde-se constatar o espectro de tensões e a faixa de tensão mínima e máxima calculados pelo software, ilustrado na figura 20 em forma de uma combinação de imagens e separadamente no Anexo B. As alterações nos espectros se referem às alterações do esquema de cores.

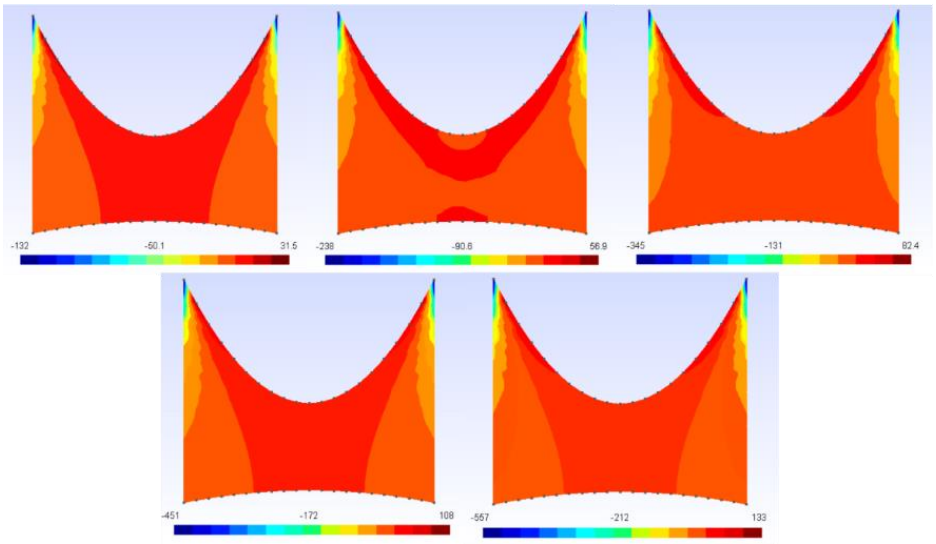


Figura 20: Imagens Geradas na Simulação A. (Autor, 2019)

4.6.3.1. Simulação tipo B

A simulação do corpo de prova B aconteceu de maneira análoga à simulação do corpo de prova A, obedecendo a Tabela 5, no entanto, apenas as cargas C, D e E foram utilizadas por MELO, 2019. A Figura 21 mostra os resultados do espectro de tensões para cada um dos ensaios realizados. Para as imagens separadas, consultar o Anexo C. As alterações nos espectros se referem às alterações nos esquemas de cores. A carga foi distribuída ao longo de toda a superfície superior do corpo de prova.

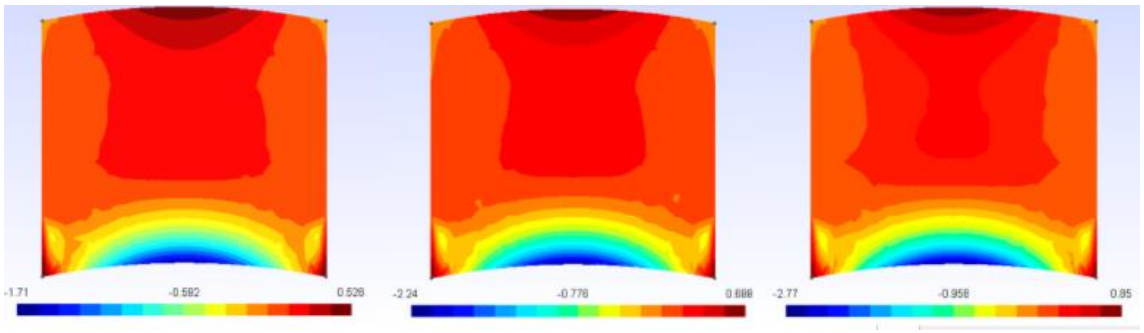


Figura 21: Imagens Geradas na Simulação B. (Autor, 2019)

Capítulo 5: Análises e Discussões

5.1. Resultados Esperados

Segundo os ensaios em laboratórios realizados por MELO, 2019 utilizando as geometrias especificadas, as tensões de compressão se concentram nos pontos de aplicação da carga, ou seja, nos vértices onde y = 8,9 para a configuração parabólica (A) e uma carga distribuída ao longo da base superior para a configuração retangular (B).

Ao longo da geometria, precisamente partindo na direção horizontal na reta onde x = 4, encontra-se uma seção onde as tensões deverão se apresentar em tom alaranjado, indicando valores próximos a zero e apontando o sentido e a configuração da linha neutra.

Os resultados esperados do trabalho são valores de tensão em escala MegaPascal (MPa). O arquivo de texto com as tensões deve apresentar valores próximos a zero nos nós correspondentes à linha neutra visível no espectro gerado e sua configuração deve se aproximar da verificada em laboratório.

Também foram realizadas simulações das geometrias no software Solidworks para que haja um comparativo dos espectros de tensão. O esquema de cores foi definido para 16 cores na simulação do software livre para que a comparação seja feita nas mesmas condições. A simulação pelo Solidworks já apresentou uma quantidade de cores definida.

Como é possível verificar na Figura 22, os espectros de tensão simulados em software livre são próximos aos simulados em um software profissional.

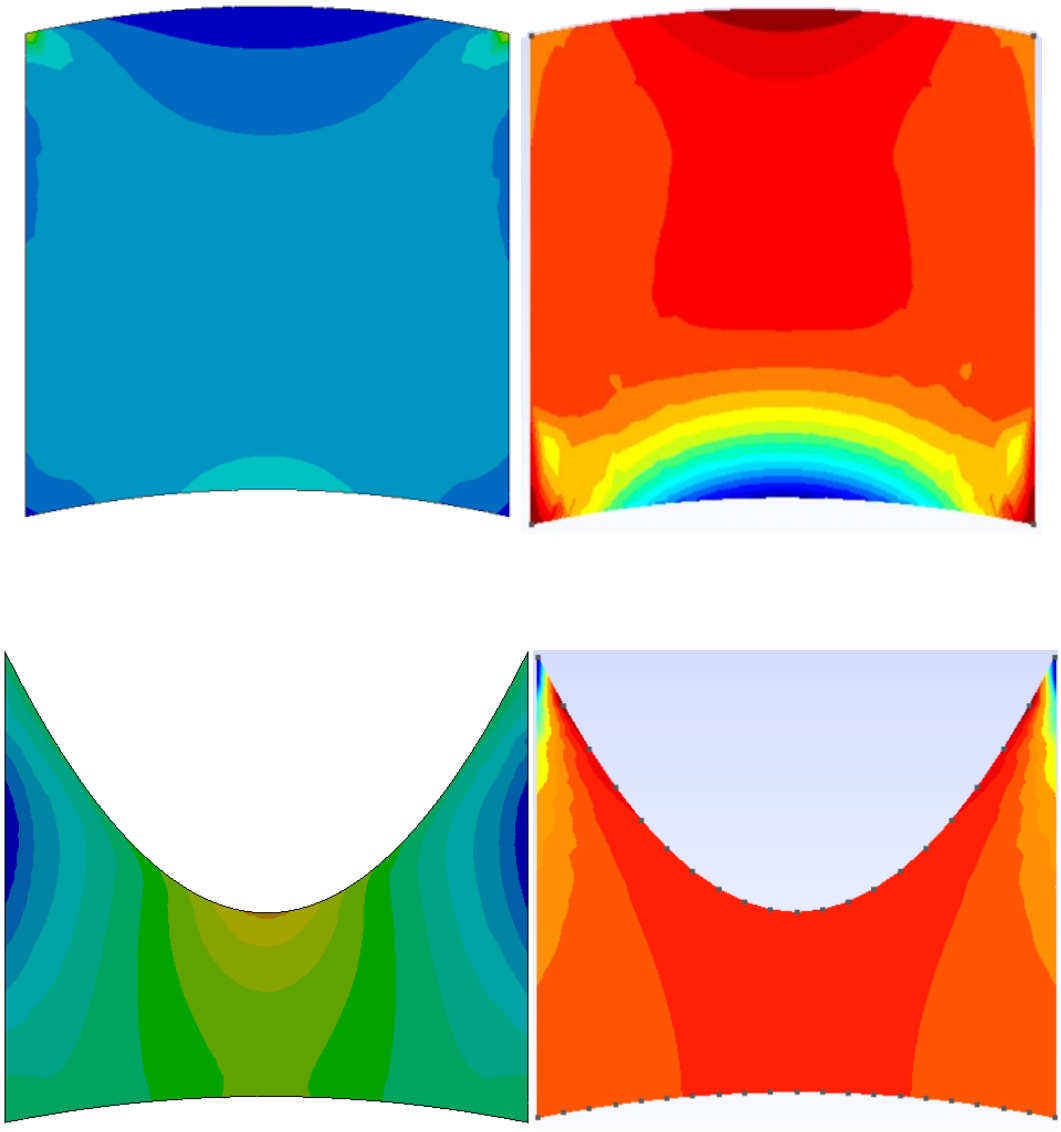


Figura 22: Comparação Solidworks (esquerda) e software livre (direita). (Autor, 2019)

5.2. Resultados Obtidos

5.2.1. Simulação A

A partir dos dados inseridos de cargas para cada etapa do ensaio, as tensões foram calculadas pelo software e o espectro de cores foi gerado, tendo como resultado a figura 24 (Anexo B). Além disso, o arquivo correspondente aos nós de cada tensão foi gerado para a possível identificação da linha neutra

(Anexo G). As Tabelas no Anexo D.1 apresentam as tensões nos principais nós da geometria A.

5.2.1. Simulação B

A partir dos dados inseridos de cargas para cada etapa do ensaio, as tensões foram calculadas e o espectro de tensões foi gerado, tendo como resultado a figura 25 (Anexo C). Também foi gerado um arquivo de texto com as tensões em cada nó, permitindo a identificação da linha neutra. As Tabelas no Anexo D.2 apresentam as tensões nos principais nós da geometria B. As linhas neutras foram marcadas na Figura 27 (Anexo G).

5.3. Comparativo com Cálculo Manual

Com o intuito de verificar a confiabilidade do método utilizado, cinco nós principais foram selecionados para que fossem realizados cálculos manuais, os quais foram posteriormente comparados com os valores de tensão obtidos via software. Os nós selecionados encontram-se na Tabela 7.

Tabela 7: Regiões Selecionadas (Autor, 2019).

Rótulo	Coordenada	Área Aproximada [mm²]
I	0;8,9	0,063
II	10;8,9	0,063
III	0;0	1,331
IV	10;0	1,331
V	5;2,4	0,010

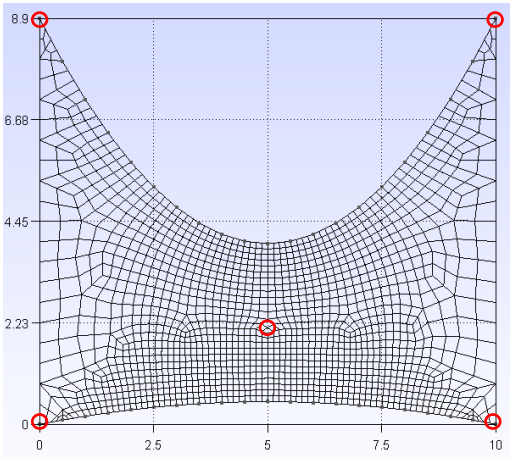


Figura 23: Guia para os Nós Selecionados. (Autor, 2019)

Utilizando a situação de carga A (8,554 N) e utilizando a premissa de que os nós inferiores sofrem ação dos nós acima dele, foi possível calcular as tensões nos nós. As áreas aproximadas dos nós levaram em consideração o ponto médio entre o nó e seus nós vizinhos. O cálculo utilizou a expressão de razão entre Força (N) e área (mm²) para obter uma saída de tensão (MPa), respeitando as direções dos carregamentos (compressão e tração). Os resultados obtidos estão na tabela 8.

Tabela 8: Tensões Calculadas em Nós (Autor, 2019).

Rótulo	Tensão [MPa]
I	-135,78
II	-135,78
III	-6,43
IV	-6,43
V	1,4080

Os dados calculados foram comparados com os dados obtidos via software, resultando na Tabela 9.

Tabela 9: Comparativo de Tensões (Autor, 2019).

Rótulo	Tensão [MPa]	Tensão Software [MPa]
I	-135,78	-131,70
II	-135,78	-131,70
III	-6,43	-6,32
IV	-6,43	-6,32

V	1,4080	1,4409

O erro médio calculado foi de aproximadamente 1,689 MPa.

Outras tensões notáveis encontram-se no Anexo D1.

5.4. Conclusões

As simulações A e B mostraram-se compatíveis com os resultados obtidos em laboratório. Visto que houve uma aproximação da geometria real para que se torna-se possível a simulação mais ágil, os valores de tensão fazem sentido ao que se refere aos movimentos trativos e compressivos e a forma do espectro de tensões é similar ao espectro obtido na simulação feita em Ansys.

É possível verificar o posicionamento da linha neutra tanto no espectro de tensões gerado, quanto nos valores de tensão no documento de tensão nó a nó. Portanto, a adaptação do software Elastopy ao problema foi bem sucedida, possibilitando o estudo de pontos concentradores de tensão e abrindo espaço para uma análise de deformações.

A identificação de cada nó pode ser obtida através da opção Node Labels (Rótulos de Nós) no software Gmsh, muito embora também possa ser descoberto o rótulo de cada um dos nós através do arquivo .msh criado. Este arquivo pode ser facilmente aberto em editores de texto (recomenda-se o Bloco de Notas do Microsoft Windows). Os rótulos de superfície, geometria e física do problema também podem ser abertos de forma análoga através do arquivo .geo.

5.5. Considerações Finais

O trabalho de conclusão de curso foi bem sucedido, uma vez que seu objetivo era adaptar um software de análise de elasticidade para a análise de

tensões nó a nó e a geração de um gráfico relacionando cores às tensões em cada elemento. O estudo de elementos finitos possibilitou uma verificação ponto a ponto e a obtenção de geometrias infinitesimais adequadas ao problema.

Dentre os desafios do trabalho, encontraram-se a escassez de material relacionado à programação de Análise de Tensão pelo Método dos Elementos Finitos em Python. Portanto, fez-se necessário consultar o senhor Nasser Alkmim, criador de softwares como Elastopy e Scikit, Engenheiro Civil pela UnB e mestre em estruturas com foco em mecânica computacional, métodos numéricos e programação em Python, também pela UnB.

Ao final de vários meses de trabalho, o software encontra-se operante e preparado para receber entradas de Força, Coeficiente de Poisson, Posição da Força, Módulo de Elasticidade e geometrias variadas, sendo necessária sua associação com o software Gmsh, preferencialmente em sua versão 3.0.

5.6. Sugestões para Trabalhos Futuros

Para trabalhos futuros, sugere-se:

- O aprimoramento do método de elemento finito, optando por geometrias triangulares (Anexo F), uma vez que se acomodam de forma mais natural às geometrias menos regulares.
- Uma melhoria na paleta de cores, permitindo a identificação de tensões específicas importantes sem o auxílio do arquivo de texto.
- Que a geração do o arquivo de texto, possa ser criada automaticamente uma tabela que permita uma visualização mais limpa das tensões.
- Uma identidade visual para o software para que, aqueles que não possuem familiaridade com linhas de código, possam inserir os dados de forma mais confortável e não seja necessária a instalação de muitos pacotes de arquivo (Python, Pypi, Numpy, entre outros);
- Criar um banco de dados de geometrias e malhas para que o acesso ao Gmsh se restrinja apenas às geometrias não convencionais;

- Associar o software às bibliotecas de aprendizado (Keras, TensorFlow e OpenCv), possibilitando o reconhecimento de imagens, geometrias, malhas, tensões e a leitura de espectros similares.

6. Referências

3D Print. **Uploads.** Disponível em: <https://3dprint.com/wp-content/uploads/2016/08/autodesk.jpg> Acesso em: 20 de Fevereiro de 2019.

ALKMIM, Nasser Samir. **Implementação Computacional da Solução de Problemas Térmicos e Mecânicos pelo Método dos Elementos Finitos em Python**. Universidade de Brasília, 2016

ASHBY, Michael F., JONES, David R. H.. **Engenharia de Materiais – Uma Introdução a Propriedades, Aplicações e Projetos – Volume I – 3ª Edição**. Editora Elsevier – Rio de Janeiro, 2007.

BEER, F. P., JOHNSTON, R. E.. **Mecânica Vetorial para Engenheiros – Estática**. 9ª Ed. Editora McGraw Hill. São Paulo, 2012.

BÉRKEL. **Informativo Técnico**. Disponível em: <https://docplayer.com.br/13432671-Informativo-tecnico-acrilico-extrudado.html> Acesso em: 20 de Dezembro de 2018.

BORGES, Lúcio Mauro Souza. **Análise de Tensões Integrada a Sistema de Engenharia Reversa para Projeto e Confecção de Próteses**. Dissertação de Mestrado. Universidade Federal do Rio Grande do Norte. Natal – RN, 2014.

COSTA, Antônio Cardoso. **História da Computação Gráfica**. Departamento de Engenharia Informática. ISEP/IPP, 2002.

Engenharia Cotidiana. **Como o Matlab pode ajudar sua empresa ou seu curso de engenharia?** Disponível em: < http://engenhariacotidiana.com/como-o-matlab-pode-ajudar-sua-empresa-ou-seu-curso-de-engenharia> Acesso em: 20 de Fevereiro de 2019.

FARIA, **Antonio Neto. Formulação Variacional para o Eletromagnetismo e suas Aplicações**. Tese de doutorado. Universidade Federal de Itajubá, 2005.

FELIPPA, C. A.. **Introduction to Finite Element Methods**. Department of Aerospace Engineering Sciences. University of Colorado, 2004.

FELIPPA, C. A.. **Advanced Finite Element Methods**. Department of Aerospace Engineering Sciences. University of Colorado, 2005.

GIACCHINI, Breno L. ; BURGARELLI, Denise ; BIEZUNÉ, Rodney J.. Elementos Finitos. Ouro Preto, MG: 6º Encontro Mineiro de Equações Diferenciais, UFOP, 2012

HELLMEISTER, Luiz A.. **Computação Gráfica: Ferramenta Didática de Comunicação, Design e Produção**. World Congress of Communication and Arts. 2010.

Historic Achievement Recognized. **Shippingport Atomic Power Station – A National Historic Mechanical Engineering Landmark,** 2006.

KELLY, P. A.. **Mechanics Lecture Notes: An Introduction to Solid Mechanics**. Disponível em: <http://homepages.engineering.auckland.ac.nz/~pkel015/SolidMechanicsBooks/Part_I/BookSM_Part_I/07_ElasticityApplications/07_Elasticity_Applications_03_Presure_Vessels.pdf> Acesso em: 5 de Agosto de 2018.

Mathworks. **Matlab**. Disponível em: < http://www.ung.br/noticias/conheca-4-softwares-utilizados-na-engenharia-mecanica-0> Acesso em: 20 de Fevereiro de 2019.

MELO, Lucas R.. **Análise de Tensão por Fotoelasticidade em um Corpo de Prova Submetido à Flexão**. Trabalho de Conclusão de Curso. Universidade Federal do Recôncavo da Bahia, 2019.

MGC. **Mechanical Solid Edge.** Disponível em: < https://mgc-images.imgix.net/mechanical/solidedge1-0A29B3FD.png?q=80&w=1920&fit=max> Acesso em: 20 de Fevereiro de 2019.

Oráculo TI. **Quais são as Aplicações Famosas Feitas em Python?** Disponível em: < https://oraculoti.com.br/2017/03/24/quais-sao-as-aplicacoes-mais-famosas-feitas-em-python/> Acesso em: 20 de Fevereiro de 2019.

Pic-c. **2D Finite Element Method in Matlab**. Disponível em: <https://www.particleincell.com/2012/matlab-fem/> Acesso em: 20 de Fevereiro de 2019.

PILKEY, Walter D., PILKEY, Orrin H.; **Mechanics of Solids**. S.I.: s.n..

Portal Educação. **Autocad Versus Solidworks**. Disponível em: < https://www.portaleducacao.com.br/conteudo/artigos/informatica/autocad-versus-solidworks/48936> Acesso em: 20 de Fevereiro de 2019.

RAO, Singiresu S.. **The Finite Element Method in Engineering**. 6ª Edição. Editora BH. 2013.

Sempre Update. **Python é a Linguagem Preferida dos Hackers.** Disponível em: < https://sempreupdate.com.br/python-e-a-linguagem-preferida-dos-hackers/> Acesso em: 20 de Fevereiro de 2019.

Solidworks. **Simulation Display.** Disponível em: < https://blogs.solidworks.com/solidworksblog/wp-content/uploads/sites/2/2016/09/simulation_display.png> Acesso em: 20 de Fevereiro de 2019.

TONTI, E. **The Mathematical Structure of Classical and Relativistic Physics.** Springer New York – NY, 2013.

TONTI, E. **Why Starting from Differential Equations for Computational Physics?**. Journal of Computational Physics, 2014.

USP E-Disciplinas. **Fratura de Membros Trincados**. Disponível em: < https://edisciplinas.usp.br/pluginfile.php/1946181/mod_resource/content/1/Aula%201-%20FRATURA%20DE%20CORPOS%20TRINCADOS.pdf> Acesso em: 6 de Agosto de 2018.

VAN VLACK, Lawrence H., CAMARGO, Luiz Paulo Ferrão. **Princípios de Ciência dos Materiais**. Disponível em: <http://www.ufrgs.br/petmateriais/graduacao/bibliografias/LAWRENCE%20H.%20VAN%20VLACK-%20LUIZ%20PAULO%20CAMARGO%20FERRAO%20PRINCIPIOS%20DE%20CIENCIA%20DOS%20MATERIAIS%20%20%20%202000.pdf/view> Acesso em: 16 de Janeiro de 2019.

7. Anexos

7.1. Anexo A

Tabela 6: Características Mecânicas do Acrílico Extrudado (BÉRKEL, 2018).

	Método ASTM	Valor (Espessura 0,250”)
Resistência à Tração	D638	69 Mpa
Alongamento à Ruptura	D638	4,5%
Módulo de Elasticidade	D638	3300 Mpa
Resistência à Compressão	D695	117 Mpa
Dureza Rockwell	D785	M-93

7.2. Anexo B

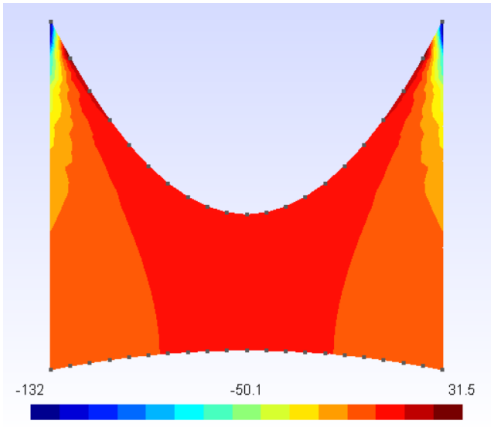


Figura 24: Simulação do Corpo de Prova A para carga de 8,554 N (Autor, 2019).

7.3. Anexo C

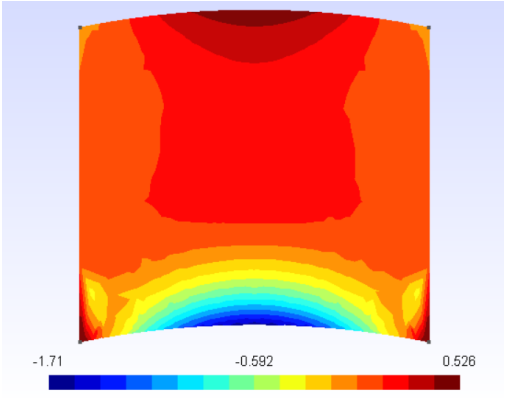


Figura 25: Simulação do Corpo de Prova B para carga de 25,535 N (Autor, 2019).

7.4. Anexo D1

Tabela 7: Tensões Principais nos Nós para Carga A (Autor, 2019).

Ponto (Rótulo)	Coordenada	Tensão (MPa)
A (1)	(0;8,9)	-131,697
B (2)	(10;8,9)	-131,697
C (172)	(5,4)	0,1748
D (3)	(0,0)	-6,319
E (4)	(10,0)	-6,319
F (24)	(9,5;7,9)	31,490
G (6)	(0,5;7,9)	31,490
I (331)	(5;2,23)	0,0054

Tabela 8: Tensões Principais nos Nós para Carga A+B (Autor, 2019).

Ponto (Rótulo)	Coordenada	Tensão (MPa)
A (1)	(0;8,9)	-238,024
B (2)	(10;8,9)	-238,024
C (172)	(5,4)	0,3160
D (3)	(0,0)	-11,421
E (4)	(10,0)	-11,421
F (24)	(9,5;7,9)	56,9141
G (6)	(0,5;7,9)	56,9136
I (331)	(5;2,23)	0,0098

Tabela 9: Tensões Principais nos Nós para Carga A+B+C (Autor, 2019).

Ponto (Rótulo)	Coordenada	Tensão (MPa)
A (1)	(0;8,9)	-344,677
B (2)	(10;8,9)	-344,677
C (172)	(5,4)	0,4575
D (3)	(0,0)	-16,537
E (4)	(10,0)	-16,537
F (24)	(9,5;7,9)	82,415
G (6)	(0,5;7,9)	82,415
I (331)	(5;2,23)	0,0143

Tabela 10: Tensões Principais nos Nós para Carga A+B+C+D (Autor, 2019).

Ponto (Rótulo)	Coordenada	Tensão (MPa)
A (1)	(0;8,9)	-451,267
B (2)	(10;8,9)	-451,267
C (172)	(5,4)	0,599
D (3)	(0,0)	-21,653
E (4)	(10,0)	-21,653
F (24)	(9,5;7,9)	107,901
G (6)	(0,5;7,9)	107,901
I (331)	(5;2,23)	0,0187

Tabela 11: Tensões Principais nos Nós para Carga A+B+C+D+E (Autor, 2019).

Ponto (Rótulo)	Coordenada	Tensão (MPa)
A (1)	(0;8,9)	-557,243
B (2)	(10;8,9)	-557,243
C (172)	(5,4)	0,740
D (3)	(0,0)	-26,739
E (4)	(10,0)	-26,736
F (24)	(9,5;7,9)	133,241
G (6)	(0,5;7,9)	133,241
I (331)	(5;2,23)	0,023

7.5. Anexo D2

Tabela 12: Tensões Principais nos Nós para Carga C (Autor, 2019).

Ponto (Rótulo)	Coordenada	Tensão (MPa)
A (1)	(0;8,9)	-557,243
B (2)	(10;8,9)	-557,243
C (172)	(5,4)	0,740
D (3)	(0,0)	-26,739
E (4)	(10,0)	-26,736
F (24)	(9,5;7,9)	133,241
G (6)	(0,5;7,9)	133,241
I (331)	(5;2,23)	0,023

7.6. Anexo E1

```
import skmech
import numpy as np
E, nu = 3.22e9, 0.25  # Pa, -
mesh_file = 'simulaa'
msh = skmech.Mesh(f'{mesh_file}.msh')
displbc = {46: (None, 0), 22: (0, 0)}
mat = skmech.Material(E={47: E},
                      nu={47: nu})   # Plane stress
trac = {1: (0, -844.411), 21: (0, -344.411)} # N
model = skmech.Model(
    msh, material=mat, displacement_bc=displbc,
    thickness=1, traction=trac)
u = skmech.statics.solver(model)
# Extrapola Gauss para os nós
stress_component = 1                                  # [0, 1, 2] -> [sx, sy, szy]
sig = skmech.postprocess.stress_recovery_smoothed(model) # dic {nid: (sx, sy, sxy)}
sig_component = {nid: sig[nid][stress_component] for nid in model.nodes.keys()} # x-component dic {nid: sx}
# Geração TXT
np.savetxt(f'{mesh_file}.txt', np.c_[list(sig_component.keys()),
                                     list(sig_component.values())],
           fmt=['%d', '%.4f'])
# Salvar em Msh
skmech.gmshio.write_field(sig_component, mesh_file, '___', ndim=1)
```

Bloco 1: Código para Simulação A (Autor).

7.7. Anexo E2

```
import skmech
E, nu = 3.3e6, 0.22
msh = skmech.Mesh('nome_mesh')
displbc = {8: (None, 0), 3: (0, 0)}
mat = skmech.Material(E={9: E},
                      nu={9: nu})
trac = {5: (0, -44.411)}
model = skmech.Model(
    msh, material=mat, displacement_bc=displbc,
    thickness=1, traction=trac)
u = skmech.statics.solver(model)
sig = skmech.postprocess.stress_recovery_smoothed(model)
sig_x = {nid: sig[nid][0] for nid in model.nodes.keys()} # x-component
# Geração TXT
np.savetxt(f'{mesh_file}.txt', np.c_[list(sig_component.keys()),
                                     list(sig_component.values())],
           fmt=['%d', '%.4f'])
# Salvar em Msh
skmech.gmshio.write_field(sig_x, 'geooriginalB', 'Sigma x', ndim=1)
```

Bloco 2: Código para Simulação B (Autor).

7.8. Anexo E3

```
def __init__(self, mesh, material=None, traction=None,
             displacement_bc=None, body_forces=None, zerolevelset=None,
             imposed_displ=None,
             num_quad_points=2, thickness=1., etypes=[3],
             microscale=False, homogenized_c=None):
    self.mesh = mesh
    self.material = material
    self.traction = traction
    self.body_forces = body_forces
    self.displacement_bc = displacement_bc
    self.imposed_displ = imposed_displ
    self.thickness = thickness
    self.dof_displacement = 0
    self.etypes = etypes
    self.elements = self._get_elements(mesh.elements)
    self.nodes = mesh.nodes
    self.num_ele = len(self.mesh.elements)
    self.num_quad_points = self._get_number_quad_points(num_quad_points)
    self.num_nodes = len(self.mesh.nodes)
    self.num_dof_node = self._get_number_dof()
    self.num_dof = self.num_nodes * self.num_dof_node
    self.nodes_dof = self._generate_dof()
    self.id_f, self.id_r = self.get_free_restrained_dof()
    if zerolevelset is None:
        self.xfem = None
    else:
        self.xfem = Xfem(self.nodes, self.elements,
                         zerolevelset, material)
        self.num_dof = self.xfem.num_dof
```

7.9. Anexo E4

```
def __init__(self, filename):
    geo_path = os.path.join(filename+'.geo')
    geo_file = open(geo_path, 'r')
    self.name = filename
    self.physical_line = {}
          self.physical_surf = {}
         self.surf = {}
    self.line_loop = {}
    self.line = {}
    for txt_line in geo_file:
        num_list = find_num(txt_line)
            nl = [int(f) - 1 for f in num_list]
            self.physical_line[nl[0]] = nl[1]
        if txt_line.startswith('Plane Surface'):
            nl = [int(f) - 1 for f in num_list]
            self.surf[nl[0]] = nl[1]
        if txt_line.startswith('Physical Surface'):
            nl = [int(f) - 1 for f in num_list]
            self.physical_surf[nl[0]] = nl[1]
        if txt_line.startswith('Line('):
            nl = [int(f) - 1 for f in num_list]
            self.line[nl[0]] = nl[1:]
        if txt_line.startswith('Line Loop'):
            nl = [abs(int(f)) - 1 for f in num_list]
            self.line_loop[nl[0]] = nl[1:]
    msh_path = os.path.join(filename+'.msh')
    msh_file = open(msh_path, 'r')
```

Bloco 4: Código para Malha (Autor).

7.10. Anexo E5

```
start = time.time()
    print('Starting statics solver at {:.3f}h '.format(t / 3600), end='')
    K, P = 0, 0
    for eid, [etype, *edata] in model.elements.items():
        element = constructor(eid, etype, model)
        k = element.stiffness_matrix(t)
        # pb = element.load_body_vector(model.body_force, t)
        # pe = element.load_strain_vector(t)
        K += k
        # P += pb + pe
    Pt = neumann(model)
    P = P + Pt
    Km, Pm = dirichlet(K, P, model)
    U = np.linalg.solve(Km, Pm)
    # add current dof displacement to model
    model.set_dof_displacement(U)
    u = dof2node(U, model)
    end = time.time()
    print('Solution completed in {:.3f}s!'.format(end - start))
    return u)
```

Bloco 5: Código para Cálculo de Tensão (Autor).

7.11. Anexo E6

```
def show():
    plt.show()
def initiate(aspect='equal', axis='off'):
    fig = plt.figure()
    ax = fig.add_axes([.1, .1, .8, .8])
    ax.set_aspect(aspect)
    if axis == 'off':
        ax.set_axis_off()
    return fig, ax
def model(model, name=None, color='k', dpi=100, ele=False, ele_label=False,
          surf_label=False, nodes_label=False, edges_label=False):
    fig = plt.figure(name, dpi=dpi)
    ax = fig.add_subplot(1, 1, 1)
    ax.set_xlabel(r'x')
    ax.set_ylabel(r'y')
    ax.set_aspect('equal')
    draw.domain(model, ax, color=color)
if ele is True:
        draw.elements(model, ax, color=color)
    if ele_label is True:
        draw.elements_label(model, ax)
    if surf_label is True:
        draw.surface_label(model, ax)
    if nodes_label is True:
        draw.nodes_label(model, ax)
    if edges_label is True:
        draw.edges_label(model, ax)
    return None
```

```
def stresses(model, SIG, ftr=1, s11=False, s12=False, s22=False, spmax=False,
        spmin=False, dpi=100, name=None, lev=20, vmin=None, vmax=None,
        cbar_orientation='vertical', title=''):
    fig, ax = initiate()
    ax.set_xlabel(r'x')
    ax.set_ylabel(r'y')
    if s11 is True:
        ax.set_title(title)
        draw.tricontourf(model, SIG[:, 0]/ftr, ax, 'spring', lev=lev,
                vmin=vmin, vmax=vmax, cbar_orientation=cbar_orientation)
    if s12 is True:
        ax.set_title(title)
        draw.tricontourf(model, SIG[:, 2]/ftr, ax, 'cool', lev=lev,
                vmin=vmin, vmax=vmax, cbar_orientation=cbar_orientation)
    if s22 is True:
        ax.set_title(title)
        draw.tricontourf(model, SIG[:, 1]/ftr, ax, 'autumn', lev=lev,
                vmin=vmin, vmax=vmax, cbar_orientation=cbar_orientation)
```

Bloco 6: Código para Associação de Cores (Autor).

7.12. Anexo F

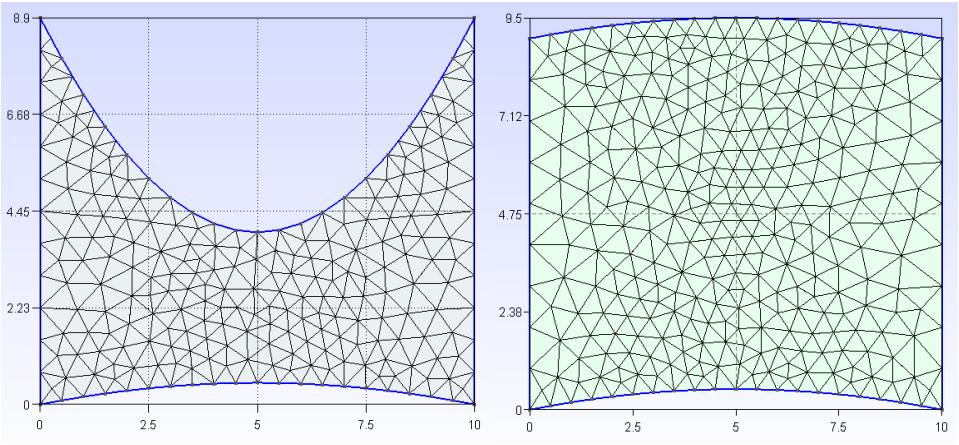


Figura 26: Malhas Triangulares para as geometrias do trabalho (Autor, 2019).

7.13. Anexo G

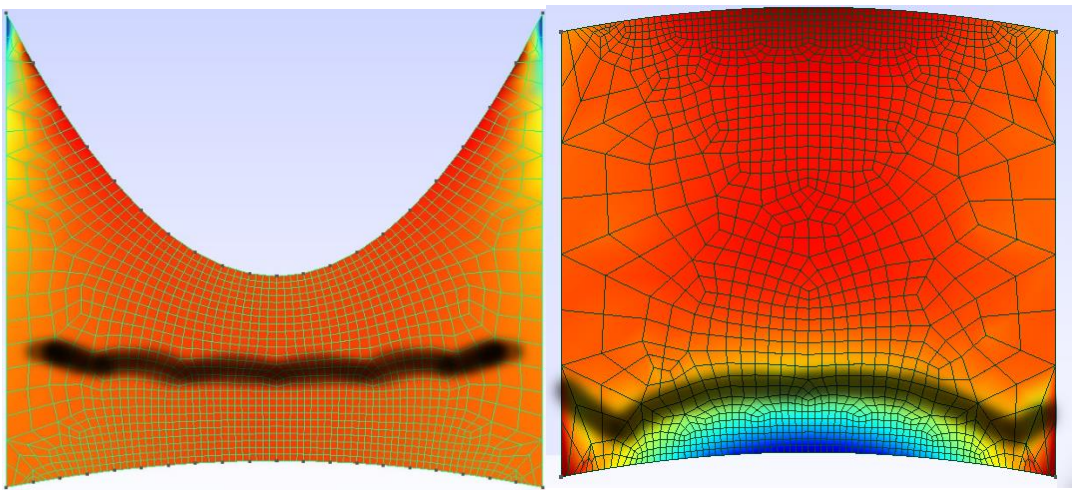


Figura 27: Malhas Triangulares para as geometrias do trabalho (Autor, 2019).