USO DO SOFTWARE FEniCS NO ENSINO DE ELEMENTOS FINITOS

Soraya de Oliveira Bandeira[1], Romina Mendez Torrez[1], Ariel de Macedo Pereira do Couto[1], Pedro Elias Gonçalves Santos[1], Renato de Queiroz Machado[2], Igor Campos de Almeida Lima[3]

[1]Instituto Alberto Luiz Coimbra de Pós-graduação de Pesquisa e Engenharia (COPPE) - Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil
(soraya.bandeira@coc.ufrj.br)
[2]Instituto Federal do Rio de Janeiro, Nilópolis, Brasil
[3]Universidade do Estado do Rio de Janeiro, Rio de Janeiro, Brasil

Resumo: O ensino de Elementos Finitos tem uma complexidade na compreensão matemática e física que o fundamenta. Este trabalho reflete a importância da plataforma FEniCS, unindo Python e Ubuntu na resolução de equações diferenciais parciais em elementos finitos e descreve o passo a passo de sua utilização para resolver um problema de forma didática, com a inserção do código aberto de programação e a eliminação de caixas-pretas, exemplificando o resultado no programa ParaView.

Palavras-chave: Elementos Finitos; FEniCS; Ubuntu; Python; ParaView.

INTRODUÇÃO

O método dos elementos finitos (MEF) constitui um procedimento numérico para obter soluções aproximadas de problemas de contorno descritos por equações diferenciais parciais (EDPs), cujas soluções analíticas exatas tornam-se complexas quanto a geométrica, heterogeneidade das propriedades dos materiais ou condições de contorno arbitrárias (Zienkiewicz e Taylor, 2000). O método baseia-se na estratégia de partição de um domínio contínuo bidimensional ou tridimensional em um conjunto de subdomínios geometricamente simples e de tamanhos bem definidos, denominados elementos finitos (Reddy, 2019). Esses elementos mantêm conexão mútua por meio de pontos discretos localizados em suas fronteiras e interiores, designados como nós.

Historicamente, as bases matemáticas do método remontam aos trabalhos de aproximação variacional por partições poligonais estabelecidos na primeira metade do século XX (Courant, 1943). Contudo, a sua consolidação e aplicação sistemática voltada para problemas de elasticidade e mecânica dos sólidos ocorreu nas décadas seguintes, impulsionada pela evolução da computação científica (Clough, 1960).

A aplicação prática do MEF segue etapas bem definidas, iniciando pelo estabelecimento da formulação fraca do problema físico, a transição do modelo contínuo para a representação discretizada apoiada em formulações variacionais ou em métodos de resíduos ponderados, com destaque para o critério de Galerkin (Fish e Belytschko, 2007), quando a malha é gerada ocorre a definição dos nós, por elementos e a interpolação das variáveis. (Zienkiewicz e Taylor, 2000). A matriz de rigidez global do sistema é então montada a partir da contribuição individual de cada elemento, possibilitando a imposição das condições de contorno e a resolução numérica das incógnitas nodais, tais como deslocamentos ou distribuições de tensões.

Sob essa perspectiva, o comportamento da variável dependente desconhecida no interior de cada elemento finito é aproximado por meio de combinações lineares de funções polinomiais de interpolação, conhecidas como funções de forma ou funções de base (Cook et al., 2001).

Essas funções de interpolação sobre a formulação fraca do problema resulta em um sistema local de equações algébricas lineares para cada elemento individual e é por isso que o MEF pode ser aplicado em problemas como análise estrutural, transferência de calor, mecânica dos fluidos, eletromagnetismo, biomecânica e engenharia de materiais (Bathe, 2014, Liu et al. 2022; Zienkiewicz et al., 2024).

A consolidação do método dos elementos finitos no ambiente acadêmico e industrial impulsionou o desenvolvimento de diversos pacotes computacionais comerciais amplamente difundidos, tais como ANSYS, ABAQUS e COMSOL Multiphysics (Logg et al., 2012). Embora tais ferramentas ofereçam interfaces gráficas altamente intuitivas e robustas bibliotecas de elementos pré-configuradas, sua arquitetura fechada impõe limitações severas quando comparada a plataformas abertas e orientadas à

formulação matemática direta, como o ecossistema FEniCS (Alnæs et al., 2015). A rigidez operacional e o alto custo de licenciamento são fatores que diminuem o uso desses softwares comerciais para a customização algorítmica avançada e para o desenvolvimento de modelos constitutivos.

O principal ponto de divergência reside no nível de abstração da formulação fraca das equações diferenciais parciais. Em ferramentas comerciais tradicionais, o usuário fica restrito a classes de problemas predefinidas, enfrentando grande rigidez no acoplamento de novos termos diferenciais não lineares (Langtangen e Logg, 2017). Por outro lado, o FEniCS fundamenta-se em uma linguagem de formas UFL (Unified Form Language), que permite a transcrição quase literal de equações diferenciais em sua forma variacional abstrata, compilando o código em tempo de execução de maneira otimizada (Kirby e Logg, 2006). Nos pacotes de código fechado, qualquer alteração na matriz de rigidez global exige rotinas complexas de programação em linguagens legadas, as quais operam de forma ineficiente fora do núcleo principal do resolvedor numérico.

Ademais, a dependência de interfaces gráficas pesadas e o modelo de licenciamento por núcleos de processamento (CPUs) limitam o uso de softwares comerciais em ambientes de computação de alto desempenho (HPC) e simulações em larga escala (Ölwegård et al., 2020). Enquanto o FEniCS distribui as malhas e os sistemas lineares associados por meio de interfaces MPI e bibliotecas lineares escaláveis como PETSc, os pacotes tradicionais cobram taxas adicionais exorbitantes para execução paralela massiva. Essa parte comercial restritiva prejudica a reprodutibilidade científica e eleva o custo de pesquisas acadêmicas focadas na otimização de formas.

Historicamente, a difusão do MEF esteve associada ao desenvolvimento de softwares comerciais consolidados, como os derivados do NASTRAN e do SAP, que se tornaram populares a partir da década de 1970 e atualmente dominam o mercado de simulação estrutural (Niiranen, 2021). Esses pacotes costumam operar como soluções de "caixa-preta", oferecendo interfaces gráficas robustas sem expor ao usuário as rotinas internas de discretização, montagem e resolução do sistema de equações. .

O FEniCS é uma plataforma de computação científica de código aberto desenvolvida para automatizar a solução numérica de Equações Diferenciais Parciais por meio do MEF, permitindo a implementação prática dessas formulações sem as limitações de transparência associadas a softwares proprietários (Logg et al., 2012). Seu uso eficiente é baseado na integração entre a plataforma FEniCS, a linguagem de programação Python e o sistema operacional Ubuntu 22.04 LTS via Windows Subsystem for Linux (Alnæs et al., 2015).

Este trabalho tem como objetivo apresentar um aplicação didática do FEniCS no ensino de elementos finitos, guiando o leitor pelas etapas de instalação, modelagem e visualização da plataforma FEniCs.

MATERIAL E MÉTODOS

A metodologia computacional empregada neste trabalho se inicia na instalação da plataforma FEniCS para o usuário, posteriormente é apresentado o ambiente operacional. Para representar as tensões e equações pela plataforma foi escolhido uma chapa de aço de emenda de viga ponte (Rodriguez , 2023), que sofreu uma abertura circular. Com as dimensões do objeto e suas descrições, o processo numérico pode ser gerado através de comando, esse conjunto é explicado em dez etapas de forma didática para garantir a reprodutibilidade e a precisão da análise estrutural, feita pela plataforma FEniCS, unida a linguagem de programação Python e o sistema operacional Ubuntu 22.04 LTS via Windows Subsystem for Linux (Alnæs et al., 2015).

Para representação dos objetivos propostos neste estudo, estabeleceu-se uma abordagem explicativa e didática fundamentada em modelagem computacional e simulação numérica avançada. Iniciando com a aplicação do método dos elementos finitos à análise de estruturas que exige um encadeamento sistemático de rotinas algorítmicas, garantindo que a transição entre o modelo físico contínuo e a aproximação discreta que preserve a consistência matemática do problema (Oden e Demkowicz, 2010). Conforme apontam Strang e Fix (2008), o controle estrito sobre cada fase do processo desde a geração da malha até a resolução dos sistemas lineares, é o fator determinante para mitigar erros de truncamento e assegurar a convergência das soluções numéricas.

A imagem resultante é apresentada de forma visual pelo progama ParaView na modelagem e simulação numérica via MEF junto ao FEniCS, sendo realizada a construção de uma chapa onde sofreu uma abertura circular cujo resultado são as tensões provocadas por descontinuidades geométricas em regime de estado plano de tensões (EPT). Para a elaboração da malha foram utilizados elementos triangulares na discretização, que ilustra a forma do comportamento mecânico das linhas de fluxo de tensão e as implicações práticas de projeto estrutural é baseada nos critérios de falha de Von Mises.

RESULTADOS E DISCUSSÃO

O ponto de partida para a utilização do FEniCS em sistemas Windows consiste na ativação do Windows Subsystem for Linux (WSL), que permite a execução de uma distribuição Linux nativa sem a necessidade

de máquinas virtuais pesadas. A instalação é realizada através do terminal PowerShell em modo administrador, utilizando o comando `wsl --install`, que configura por padrão o Ubuntu. Após o reinício do sistema e a criação de credenciais de usuário, o ambiente Linux torna-se acessível como um terminal independente, fornecendo a estabilidade necessária para gerenciar as bibliotecas de alto desempenho que compõem o núcleo do software.

Após o reinício do sistema e a criação de credenciais de usuário, o ambiente Linux torna-se acessível como um terminal independente, fornecendo a estabilidade necessária para gerenciar as bibliotecas de alto desempenho que compõem o núcleo do software.

A instalação do Python é validada via terminal, permitindo a criação de ambientes virtuais isolados onde as bibliotecas de cálculo podem ser atualizadas sem interferir no sistema operacional. O Python atuará como a interface de alto nível que traduzirá os comandos do estudante para as rotinas de baixo nível em C++ do FEniCS.

A instalação da plataforma FEniCS no ambiente Ubuntu ocorre de maneira automatizada através do Gerenciador de Pacotes APT ou via repositórios Conda-Forge. Utilizando o comando:

```
sudo add-apt-repository ppa:FEniCS-packages/FEniCS
```

O sistema vincula-se aos pacotes oficiais mantidos pelos desenvolvedores do projeto. A execução do comando de instalação (`sudo apt-get install FEniCS`) baixa e compila simultaneamente todos os componentes essenciais, como a Unified Form Language (UFL) e o motor de discretização DOLFIN, consolidando o ambiente de simulação no computador do usuário.

O FEniCS não possui uma "tela inicial" de interface gráfica (GUI) proprietária, o mesmo opera como um ecossistema de bibliotecas integrado ao ecossistema Python. Existem dois ambientes na utilização do FEniCS, que chamaremos de camadas, abaixo na figura 1 tem a primeira camada:

Figura 1. Sistema operacional Ubuntu, sendo 1ª camada. Fonte: Autora Soraya Bandeira, 2026.

Ao abrir o terminal do Ubuntu, a linha de comando apresenta a seguinte estrutura:

```
ical2026@DESKTOP-44HD03V:~$
```

Essa linha é dividida em duas partes fundamentais: a Identificação do Ambiente (Prompt) e a Instrução de Execução (Comando). O usuário inicia a sessão com o comando, conforme a figura 2, por exemplo:

```
nano simulation.py
```

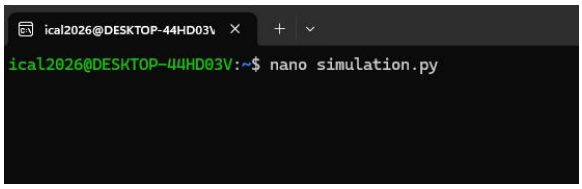

Figura 2. Sistema operacional Ubuntu com comando ao arquivo salvo (1ª camada). Fonte: Autora Soraya Bandeira, 2026.

01. Ambiente Operacional

O Ubuntu (figura 1) é uma distribuição do sistema operacional Linux baseada em Debian, amplamente utilizada em computação científica devido à sua robustez e eficiência no gerenciamento de pacotes e recursos computacionais, (Langtangen, 2016). É o ambiente nativo ideal para a compilação e execução das complexas bibliotecas de álgebra linear em C++ que rodam no núcleo do FEniCS (como PETSc, SLEPc e SCOTCH).

Tabela 1. Explicação do comando.

Elemento	Termo Técnico	Função / Significado
ical2026	Username (Usuário)	Identifica a conta de usuário que está operando o sistema.
@	Conector (At)	Indica em qual máquina/servidor o usuário está logado.
DESKTOP-44HD03V	Hostname (Máquina)	O nome do computador (padrão de instalações WSL no Windows).
:	Separador	Delimita as informações de rede do caminho do diretório.
~	Home Directory (Pasta Pessoal)	Atalho que indica que o usuário está na sua pasta raiz (/home/ical2026).
nano	Command (Invocação do Programa)	O nome do editor de texto baseado em terminal que o sistema deve executar.
simulation.py	Argument (Parâmetro/ Arquivo Alvo)	O nome do arquivo que o editor nano deve abrir. Se o arquivo não existir, o sistema o criará automaticamente.
.py	File Extension (Extensão)	Identifica o formato do arquivo, indicando ao sistema e ao editor que o conteúdo é um script em Python.

02. Interface de Edição e Controle

Após apertar a tecla "`Enter`", a segunda camada é apresentada, onde mostra o GNU nano (figura 4), que é um programa rodando dentro do terminal, sendo um editor de texto para programadores. A figura 4 apresenta um código feito pelo usuário com as dimensões da chapa, parâmetros do EPT e a identificação da geometria de abertura na chapa.

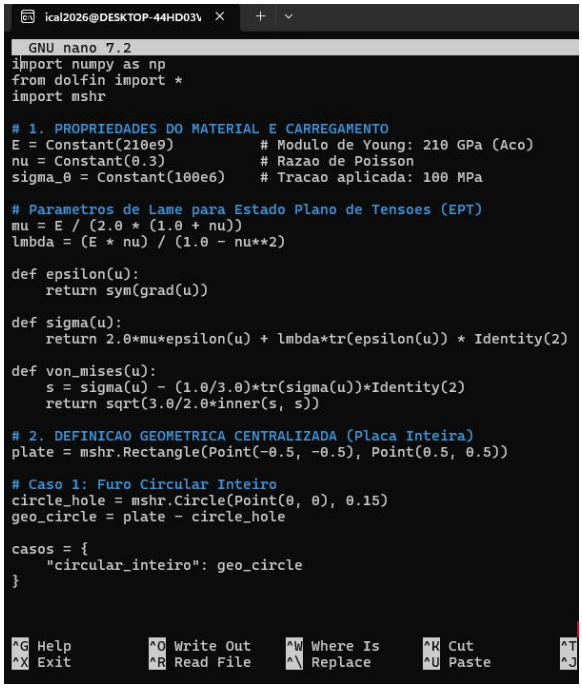

Figura 4. 2ª camada com o código Python feito pelo usuário no GNU nano. Fonte: Autora Soraya Bandeira, 2026.

Para o gerenciamento, escrita e modificação do código-fonte do script (simulation.py), utilizou-se o terminal integrado do Linux em conjunto com o editor de texto GNU nano 7.2. Esta interface em modo texto (linha de comando) permite o controle ágil do ambiente de simulação e a execução direta dos comandos de compilação e teste, minimizando o consumo de memória RAM do sistema.

03. Automação e Linguagem Base

Esta etapa do programa é inteiramente trabalhada na linguagem Python 3. O interpretador executa o script de forma sequencial, realizando a importação automatizada dos módulos numéricos fundamentais (NumPy), mecânicos (DOLFIN) e geométricos (mshr). O Python atua como uma linguagem de alto nível para automatizar chamadas de solvers compilados em linguagens de baixo desempenho (C/C++), conforme destacado por na imagem a seguir:

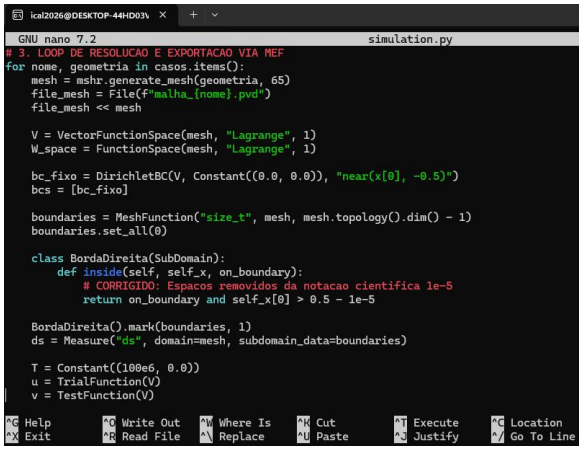

Figura 5. Segunda parte do código Python no GNU nano na 2ª camada. Fonte: Autora Soraya Bandeira, 2026.

Tabela 2. Comandos de MEF.

Linha de Código (Comando)	Conceito no MEF	Função Técnica no Projeto
mshr.generate_mesh()	Discretização do Domínio	Divide a geometria contínua em subdomínios menores, de problema infinito para problema finito com nós e elementos.
VectorFunctionSpace()	Espaço de Interpolação	Define as funções (polinômios de Lagrange de Grau 1) ao aproximar o comportamento do deslocamento dentro de cada elemento.
DirichletBC()	Condição de Contorno Essencial	Define as restrições físicas ou apoios do modelo (engastes, apoios fixos).
TrialFunction / TestFunction	Formulação Fraca (Galerkin)	Define as variáveis para o método de resíduos ponderados. A Trial (u) representa a incógnita real e a Test (v) representa os pesos virtuais.
a_form e L_form	Equações Integrais	Representam a física do problema. a_form é o trabalho virtual interno (matriz de rigidez) e L_form é o trabalho virtual externo (vetor de forças/tração).
solve(a_form == L_form, ...)	Resolução do Sistema Linear	O comando principal do MEF. Ele monta o sistema global de equações algébricas ($K \cdot U=F$), aplica as condições de contorno e resolve o sistema para encontrar os deslocamentos nos nós (u_sol).

Essas duas camadas" diferentes no ambiente de desenvolvimento no Ubuntu (rodando via WSL no Windows) trabalham juntas, mas servem para coisas totalmente distintas. A segunda camada serve exclusivamente para editar as linhas de código do arquivo `simulation.py` como o módulo de Young, a Razão de Poisson, a geometria da chapa com o furo circular. As duas telas trabalham juntas, o fluxo de

trabalho com o FEniCS vai ser sempre esse ciclo: no Editor (nano) se digita o código do FEniCS, define a malha, as propriedades do material e as condições de contorno. Quando terminar é necessário salvar com a sequência (`Ctrl + O`), `Enter` e sair com (`Ctrl + X`), voltando para primeira camada conforme a figura 6, onde basta apertar `Enter` para obter a resposta do comando.

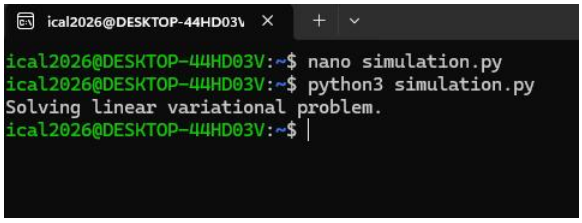

Figura 6. Primeira camada onde se apresenta a resposta, nesse caso o arquivo foi gerado. Fonte: Autora Soraya Bandeira, 2026.

Para acessar no computador, segue o exemplo deste usuário (ical2026):

Abra o Explorador de Arquivos do Windows (`Win + E`). No menu lateral esquerdo, role a barra até o final, onde tem uma baleia ou a palavra Linux. No exemplo deste trabalho:

Clique em Linux ➔ Ubuntu ➔ home ➔ `ical2026`.

Todos os arquivos do projeto estarão dentro dessa pasta `ical2026`.

04. Modelagem Geométrica da chapa com furo

Como exemplo de trabalho utilizamos a chapa de Rodriguez (2023) aplicação, conforme a figura abaixo:

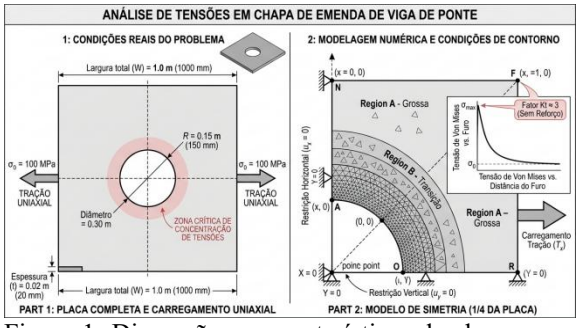

Figura 1: Dimensões e características da chapa com a abertura circular. Fonte: Adaptado de Rodriguez (2023)

As dimensões Externas da chapa quadrada foi de 1.0 m × 1.0 m, ocupando o domínio $[-0.5, 0.5] \times [-0.5, 0.5]$. A abertura do furo circular teve raio de 0.15 m , sendo o Diâmetro = 0.30 m.

A geometria bidimensional do domínio analisado foi construída de maneira analítica através do módulo geométrico mshr. O modelo consiste em uma chapa quadrada centralizada na origem do plano cartesiano, da qual é subtraído um furo circular perfeito. A construção matemática do domínio final de análise (Ω) é regida pelas seguintes equações de subdomínios:

Domínio da chapa (Ω_{chapa})**:** Retângulo definido pelos pontos diagonais $P_1(-0,5; -0,5)$ e $P_2(0,5; 0,5)$:

$$\Omega_{placa}=\{(x,y)\in\mathbb{R}^2 \mid -0,5 \le x \le 0,5e-0,5 \le y \le 0,5\} (1)$$

Domínio do Furo (Ω_{furo}) : Círculo posicionado no centro do plano (0,0) com raio r = 0,15:

$$\Omega_{furo}=\{(x,y)\in\mathbb{R}^2 \mid x^2+y^2\le(0,15)^2\} \quad (2)$$

Domínio Final de Análise (Ω): Operação booleana de diferença entre os conjuntos geométricos:

$$\Omega=\Omega_{placa} \setminus \Omega_{furo} \quad (3)$$

05. Discretização e Geração de Malha

Com base na geometria definida, realizou-se a discretização espacial do domínio contínuo em subdomínios triangulares (elementos finitos lineares de Lagrange), gerando a malha computacional através da biblioteca mshr com um fator de refinamento igual a 65. Esse processo divide a geometria contínua em um número finito de subdomínios para viabilizar a resolução numérica pelas aproximações do método dos elementos finitos (Fish e Belytschko, 2007). Segue leitura e discretização do domínio contínuo por meio de malhas triangulares (2D):

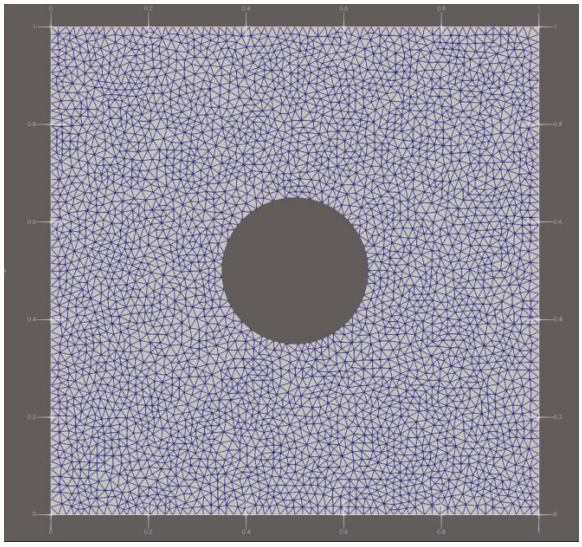

Figura 7. Discretização da chapa no ParaView. Fonte: Soraya Bandeira, 2026.

A aplicação das condições de contorno essenciais (Dirichlet) e naturais (Neumann) especificadas são necessárias nesse processo do sistema.

$$\mathbf{K} \cdot \mathbf{U} = \mathbf{F} \qquad (4)$$

K (Matriz de Rigidez Global): É uma matriz que mapeia a geometria da malha e as propriedades mecânicas do material (como o Módulo de Young e Coeficiente de Poisson). Ela dita o quão rígida é cada parte da estrutura. No código, ela é gerada a partir da forma bilinear `a_form.`

U (Vetor de Deslocamentos Nodais): É o vetor que contém as incógnitas do problema. Ele armazena os valores de deslocamento (em x e y) que cada nó da malha sofrerá. É a variável primária que o software tenta descobrir `(u_sol).`

F `(L_form)`: É um vetor que condensa todas as forças atuando no sistema, como forças de corpo (gravidade), pressões superficiais (tração) e carregamentos concentrados .

A equação 4 é importante no método dos elementos finitos para problemas lineares (como a elasticidade linear do projeto). Quando se executa o comando `solver`, que é integrado DOLFIN (o motor de computação do FEniCS) realiza-se toda a álgebra linear pesada para transformar as equações diferenciais nesse sistema matricial (LOGG et al., 2012).

No contexto da mecânica dos sólidos, cada letra dessa equação representa uma matriz ou vetor global do sistema:

O DOLFIN não resolve apenas a equação final, ele automatiza três etapas matemáticas complexas que, antigamente, os engenheiros precisavam programar manualmente linha por linha:

Formulação Variacional ➔ 1. Montagem (Assembly)➔ 2. Aplicação de BCs➔ 3. Resolução Numérica

BCs (Boundary Conditions) - Condições de Contorno ou Condições de Fronteira

O DOLFIN pega as integrais definidas em a_form e L_form e as calcula elemento por elemento da malha (integração numérica via Quadratura Gaussiana). Ele gera matrizes locais pequenas para cada triângulo da malha e as "costura" de forma organizada para construir a enorme matriz global **K** e o vetor **F**.

Antes de resolver o sistema, o DOLFIN injeta as restrições que foram definidas em bcs (como o engaste fixo DirichletBC). Ele altera as linhas e colunas correspondentes na matriz **K** e no vetor **F** para garantir que os nós presos tenham deslocamento estritamente igual a zero (ou ao valor determinado), impedindo que a peça "voe" pelo espaço virtual quando a força for aplicada.

Como a malha possui milhares de nós, a matriz **K** se torna gigantesca, porém esparsa (cheia de elementos zero). O DOLFIN aciona bibliotecas de álgebra linear de alto desempenho (como PETSc, MUMPS ou Scipy) para resolver o sistema:

$$\mathbf{U} = \mathbf{K}^{-1} \cdot \mathbf{F} \qquad (5)$$

Ele faz isso utilizando métodos diretos ou iterativos avançados para entregar o resultado do vetor u em segundos, com alta precisão numérica.

06. Resolução Numérica via Método dos Elementos Finitos (MEF)

O cerne do processamento numérico é executado pelo solver do framework FEniCS, utilizando o módulo DOLFIN (Logg et al., 2012). O problema elástico linear resolve o campo de deslocamentos u através da formulação fraca do princípio dos trabalhos virtuais sobre o domínio Ω.

O material adotado possui comportamento mecânico elástico linear e isotrópico, configurado sob a hipótese de estado plano de tensões (EPT), cujas propriedades mecânicas de entrada (compatíveis com o aço estrutural) são:

- O Módulo de Young: ($E=210\times10^9$Pa), a

- Razão de Poisson: ($\nu = 0{,}3$)

-Tração linear aplicada na borda de $\sigma_0 = 100 \times 10^6$Pa.

Para a resolução do EPT, o FEniCS calcula internamente os Parâmetros de Lamé (μ e λ) convertidos pelas relações constitutivas clássicas (Timishenko e Goodier, 1970):

$$\mu = \frac{E}{2(1+\nu)} \qquad (6)$$

$$\lambda = \frac{E \cdot \nu}{1-\nu^2} \quad (7)$$

07. Pós-Cálculo e Formulação das Tensões de Von Mises

A partir do campo de deslocamentos primários u obtido na etapa anterior, o algoritmo calcula os tensores cinemáticos e estáticos derivados pelas equações constitutivas da mecânica dos sólidos contínuos (Shames e Cozzarelli, 1997):

Tensor de Deformações Lineares (ϵ): Representa a parte simétrica do gradiente de deslocamentos:

$$\epsilon(u)=\frac{1}{2}\left(\nabla u+(\nabla u)^T\right) \quad (8)$$

Tensor de Tensões Elásticas (σ): Obtido pela Lei de Hooke generalizada para materiais isotrópicos:

$$\sigma(u)=2\mu\epsilon(u)+\lambda tr(\epsilon(u))I_2 \quad (9)$$

onde $tr(\cdot)$ denota o traço do tensor e I_2 representa a matriz identidade de segunda ordem.

Tensor Deviador de Tensões (s): Isolamento da componente pura de distorção do material:

$$s(u)=\sigma(u)-\frac{1}{3}tr(\sigma(u))I_2 \quad (10)$$

Tensão Equivalente de Von Mises (σ_{vm}): Critério escalar utilizado para avaliar o início do escoamento e a concentração de tensões ao redor da descontinuidade geométrica do furo:

$$\sigma_{vm}=\sqrt{\frac{3}{2}(s:s)} \quad (11)$$

Onde $(s:s)$ representa o produto interno tensorial duplo do desvio de tensões.

08. Exportação dos Dados Numéricos

Após a convergência do solver e a resolução dos campos mecânicos, os dados numéricos de cada nó e elemento da malha são mapeados e gravados em disco. O script gera arquivos com extensões .pvd e .vtu através do comando nativo de saída do pacote DOLFIN. Este formato estruturado baseado em XML preserva a integridade das matrizes de resultados para ferramentas externas de análise (Ahrens et al., 2005).

09. Visualização e Análise Gráfica

Os arquivos .pvd exportados são importados no software de pós-processamento científico ParaView (Ahrens et al., 2005). Através de filtros visuais e paletas de cores, são renderizados mapas de contorno do campo escalar das Tensões de Von Mises, facilitando a análise qualitativa e quantitativa dos fatores de concentração de tensão elástica (K_t) adjacentes ao furo central da chapa.

Concluída a etapa de resolução numérica, os campos vetoriais (como deslocamentos e tensões) resultantes são salvos em disco na pasta de trabalho do Ubuntu. O FEniCS exporta estes dados estruturados em arquivos de extensão comprimida .pvd ou .vtu (padrão Visualization Toolkit - VTK). Esses arquivos são importados para o ParaView (Figura 6).

Na etapa final, por meio do Paraview, realiza-se o pós-processamento visual do projeto, permitindo aos estudantes aplicar filtros geométricos, renderizar mapas de contorno coloridos, plotar gráficos de linhas ao longo de seções transversais da peça e gerar superfícies deformadas texturizadas. Este fluxo visual completo corrobora para a validação física dos resultados gerados e consolida de forma empírica o aprendizado do método de elementos finitos, como mostrados nas Figura abaixo.

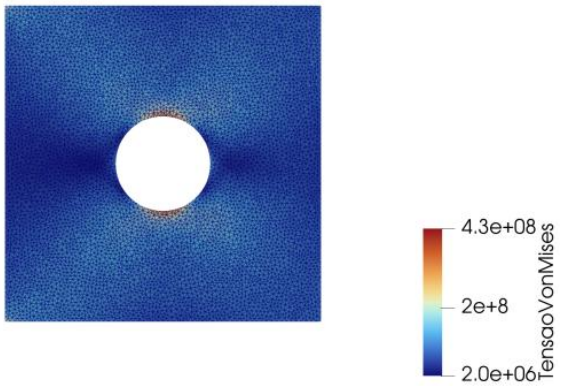

Figura 8. Chapa discretizada, apresentado as tensões. Fonte: Soraya Bandeira, 2026.

10. Documentação Técnica do Projeto

A última etapa consiste na compilação dos dados gerados, tabelas comparativas e gráficos extraídos. Esse conteúdo é estruturado e redigido na forma de artigo científico.

Por fim, o processo educacional e de pesquisa completa-se com a exportação de arquivos em formato VTK para o software livre ParaView, permitindo aos estudantes realizar o pós-processamento, a visualização gráfica tridimensional e o cálculo rigoroso de erros de convergência (como as normas de erro).

Após os comandos o terminal nano vai sumir e a tela limpa do terminal vai voltar. Digita-se o comando para rodar o script. O FEniCS vai ler esse arquivo, processar os cálculos do Método dos Elementos Finitos e gerar os arquivos de resultado (como o malha_{nome}.pvd que está no código do usuário) para abrir e visualizar em outro programa depois.

A grande vantagem dessa arquitetura é que o Python atua como uma interface acessível e flexível para os

estudantes, enquanto o FEniCS utiliza internamente a *Unified Form Language* (UFL) para traduzir a forma variacional fraca da matemática diretamente em código executável de alta velocidade (Alnæs et al., 2015). Em termos práticos e didáticos, essa sinergia permite que o aluno não precise programar exaustivamente matrizes de rigidez elementares ou loops de montagem global do zero; ele define a equação diferencial em sua linguagem matemática natural dentro do script Python e o ecossistema FEniCS/Ubuntu encarrega-se de compilar, discretizar e resolver o sistema linear de forma otimizada (Logg et al., 2012 e Langtangen 2016).

A tabela abaixo resume o fluxo de trabalho dos 10 passos adotados no desenvolvimento deste estudo.

Tabela 3. Os dez passos de MET no FEniCS.

Tópico	Atividade	Detalhamento Técnico
1 Ambiente Operacional	Infraestrutura de Computação e Sistema Operacional	Implementação utilizando Ubuntu 22.04 LTS integrada via WSL2 (Windows Subsystem for Linux).
2 Interface e Código	Ambiente de Desenvolvimento Integrado (IDE/CLI)	Utilização de interface de linha de comando (CLI) com o editor integrado GNU Nano para edição direta de código no terminal.
3 Automação e Lógica	Arquitetura Lógica e Scripting	Desenvolvimento do código-fonte em linguagem Python 3, utilizando bibliotecas científicas
4 Modelagem Geométrica	Definição do Domínio Geométrico	Construção da geometria bidimensional
5 Discretização	Discretização Espacial e Geração de Malha	Divisão do domínio contínuo em subdomínios discretos (elementos finitos triangulares de Lagrange).
6 Resolução Numérica	Formulação Variacional e Solução do Sistema	Formulação fraca do problema de Elasticidade Linear via Método de Galerkin utilizando a plataforma FEniCS. Montagem e resolução do sistema linear de equações diferenciais parciais ($K \cdot U=F$) pelo solver integrado DOLFIN.
7 Pós-Cálculo	Avaliação dos Campos Secundários de Tensão	Processamento algébrico dos resultados de deslocamento nodal (variável primária) para a extração dos componentes de deformação e cálculo do campo de Tensão Equivalente de Von Mises .
8 Exportação de Dados	Serialização e Armazenamento de Dados de Alta Densidade	Exportação das matrizes de resultados para formatos binários estruturados (.pvd e .vtu), compatíveis com visualizadores de dados científicos complexos, sem perda de precisão numérica.
9 Visualização	Análise Gráfica Avançada e Pós-Processamento	Importação dos dados no software open-source ParaView para geração de mapas de calor, linhas de corrente e análise de concentração de tensões.
10 Documentação	Síntese Metodológica e Relatório Técnico	Consolidação dos resultados obtidos e discussão .

CONCLUSÃO

O trabalho apresentou um fluxo didático para o uso do FEniCS no ensino do método dos elementos finitos, integrando ambiente Ubuntu/WSL2, linguagem Python, formulação variacional, geração de malha, resolução numérica e pós-processamento no ParaView. A aplicação à chapa com furo circular permitiu demonstrar, de forma encadeada, como um problema de elasticidade linear pode ser formulado, discretizado e resolvido computacionalmente, com visualização posterior do campo de tensões de Von Mises.

A principal contribuição do procedimento proposto está na explicitação das etapas que, em muitos softwares comerciais, tendem a permanecer ocultas ao usuário, como a definição da geometria, a imposição das condições de contorno, a montagem do sistema algébrico e a exportação dos resultados. Nesse sentido, o FEniCS mostrou-se adequado como recurso didático para aproximar a formulação matemática do MEF de sua implementação computacional, favorecendo a compreensão do papel de cada etapa do processo numérico.

Como o estudo teve caráter demonstrativo, não se pode afirmar, a partir dos resultados apresentados, que houve validação pedagógica formal ou comprovação quantitativa da aprendizagem dos estudantes. Ainda assim, o fluxo descrito constitui uma base reprodutível para atividades de ensino, podendo ser ampliado em trabalhos futuros por meio de estudos de convergência de malha, comparação com soluções analíticas ou resultados de softwares comerciais e avaliação sistemática do desempenho discente no uso do FEniCS.

AGRADECIMENTOS

S.O.B. agradece ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ) pelo apoio financeiro concedido por meio de bolsa, processo nº 130848/2026-3.

REFERÊNCIAS

AHRENS, J.; GEVECI, B.; LAW, C. ParaView: An End-User Tool for Large Data Visualization. The Visualization Handbook, Elsevier, 2005.

ALNÆS, M. S.; BLECHTA, J.; HAKE, J.; JOHANSSEN, A.; KEHLET, B.; LOGG, A.; RICHARDSON, C.; RIGNAL, J. N.; WELLS, G. N. The FEniCS Project Version 1.5. Archive of Numerical Software, v. 3, n. 100, p. 9-23, 2015.

BATHE, K. J. Finite element procedures (2nd ed.). Klaus-Jürgen Bathe, 2014.

COOK, R. D.; MALKUS, D. S.; PLESHA, M. E.; WITT, R. J. Concepts and Applications of Finite Element Analysis. 4. ed. John Wiley & Sons, Nova York, 2001.

FISH, J.; BELYTSCHKO, T. A First course in finite elements, 1st ed. Chichester: John Wiley & Sons, 2007.

KIRBY, R. C.; LOGG, A. A Compiler for Variational Forms. ACM Transactions on Mathematical Software, v. 32, n. 3, p. 417–444, 2006.

LANGTANGEN, H. P.; LOGG, A. Solving PDEs in python: the FEniCS tutorial I, 1st ed. Oslo: Springer Nature, 2016.

LANGTANGEN, H. P.; LOGG, A. Solving PDEs in Python: The FEniCS Tutorial Volume I. Springer Nature, Oslo, 2017.

LIU, W. K.; LI, S.; PARK, H. Eighty years of the finite element method: birth, evolution, and future. Archives of Computational Methods in Engineering, 2022.

LOGAN, D. L. A first course in the finite element method (6th ed.). Cengage Learning, 2023.

LOGG, A.; MARDAL, K. A.; WELLS, G. N. Automated Solution of Differential Equations by the Finite Element Method. 1. ed. Berlim: Springer Science & Business Media, 2012.

NIIRANEN, J. Technical note: Open-source finite element software – from the early to the present.

TIMOSHENKO, S. P.; GOODIER, J. N. Theory of Elasticity. 3. ed. New York: McGraw-Hill, 1970.

Rakenteiden Mekaniikka, v. 54, n. 4, p. 172-173, 2021.

ODEN, J. T.; DEMKOWICZ, L. Applied Functional Analysis. 2. ed. CRC Press, Boca Raton, 2010.

ÖLWEGÅRD, M.; JANSSON, N.; DEGERLOOD, F.; HOFFMAN, J. High-performance computing framework for automated finite element analysis. International Journal for Numerical Methods in Engineering, v. 121, n. 14, p. 3145-3168, 2020.

REDDY, J. N. An Introduction to the Finite Element Method. 4. ed. Nova York: McGraw-Hill Education, 2019.

RODRIGUEZ, L. OLIVEIRA. Um código Python de elementos finitos com comportamento elastoplástico. 2023.

SHAMES, I. H.; COZZARELLI, F. A. Elastic and Inelastic Stress Analysis. Washington: Taylor & Francis, 1997.

STRANG, G.; FIX, G. J. An Analysis of the Finite Element Method. 2. ed. Wellesley-Cambridge Press, Wellesley, 2008.

ZIENKIEWICZ, O. C.; TAYLOR, R. L. The Finite Element Method: The Basis. 5. ed. Oxford: Butterworth-Heinemann, 2000.

ZIENKIEWICZ, O. C.; TAYLOR, R. L.; GOVINDJEE, S. The finite element method: Its basis and fundamentals (8th ed.). Elsevier, 2024.