Universidade Federal de Alagoas
Unidade Acadêmica Centro de Tecnologia
Colegiado do Curso de Engenharia Civil

Tutorial do Programa Pórtico Plano AME

Lucas Henrique Correia da Rocha
lucas.rocha@ctec.ufal.br

Prof. João Carlos Cordeiro Barbirato
jccb@ctec.ufal.br

Maceió-AL, 26/05/2021

Sumário

1 Introdução

O presente documento detalha as diretrizes e o processo de elaboração do programa Pórtico Plano AME, desenvolvido com o objetivo de aplicar os conhecimentos abordados na disciplina de Análise Matricial de Estruturas, ministrada pelo professor João Carlos Cordeiro Barbirato no curso de graduação em Engenharia Civil da Universidade Federal de Alagoas.

A implementação foi feita em Python, por ser tratar de uma linguagem de alto nível, possuindo uma sintaxe simples, ampla aplicação dentro do meio acadêmico e facilmente acessível ao público, apesar de ser uma linguagem interpretada e relativamente lenta.

O programa utiliza o Método da Rigidez para o cálculo, já que ele é um método que favorece o uso computacional por possuir equações simples para o caso de estruturas formadas por barras reticuladas de seções transversais prismáticas.

A validação de resultados obtidos nas análises desenvolvidas será feita utilizando o software FTOOL desenvolvido por pesquisadores da PUC-Rio, comparando-se os resultados obtidos pelos dois softwares a partir de dois exemplos buscando cobrir as funcionalidades da aplicação.

2 Formulação empregada

Conforme citado anteriormente, o programa implementa o Método da Rigidez, também denominado como Método dos Deslocamentos, para análise das estruturas em questão, já que ele é aplicável para análise computacional, possuindo solução única, para qualquer estrutura, ainda que estaticamente ou geometricamente indeterminada, através de equações simples, porém numerosas, justificando a adoção de um computador.

Explicando sucintamente sobre a formulação, é preciso destacar inicialmente que, falando de estruturas de barras reticuladas, definimos os elementos como as próprias barras que compõem a estrutura e os nós, as ligações entre cada uma delas, são definidos dois sistemas de coordenadas para a estrutura: sistema global e local. (Figura 1)

O sistema global referencia as forças e os deslocamentos com relação a estrutura inteira, numerando-se as coordenadas de 1 a 3n, onde n é a quantidade de nós da estrutura. Adota-se, portanto, um sistema de coordenadas cartesiano no qual a estrutura é inserida, estabelecendo direções principais e convenção de sinais conveniente. Já o sistema de coordenadas local está associado a cada elemento e, desta maneira, pode ser definido conforme necessidade da aplicação.

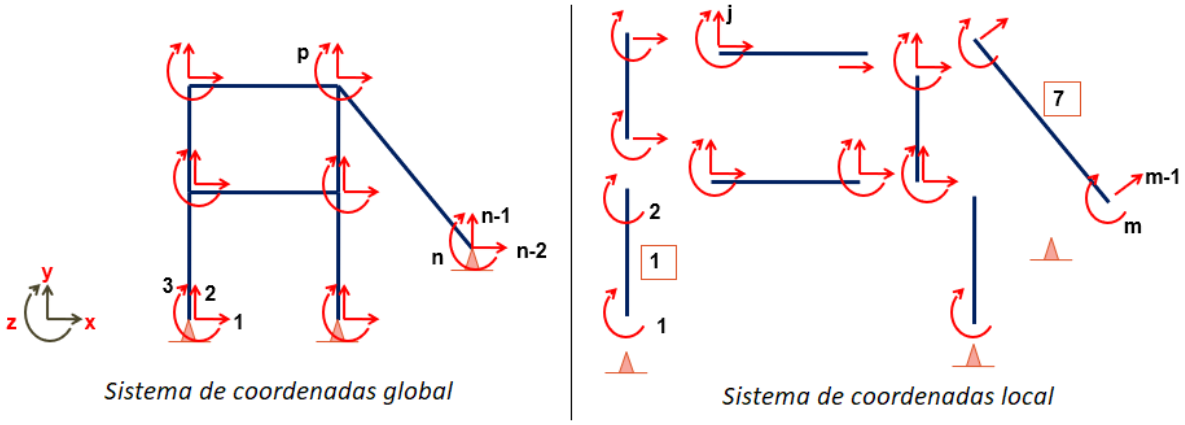

Figura 1: Sistemas de coordenadas global e local
Fonte: Adaptado do material de aula do professor

Para a implementação do programa é interessante padronizar o sistema de coordenadas locais para todos os elementos e, visando uma análise eficiente, este sistema de coordenadas local é adotado de modo a representar os esforços internos solicitantes nas extremidades da barra. (Figura 2)

Com o sistema de coordenadas local assim definido, o próximo passo é definir a matriz de rigidez para este elemento. Esta matriz é o objeto matemático que relaciona os estados de deslocamento com os vetores de forças externas que o geram. Isso é realizado aplicando estados

Figura 2: Sistemas de coordenadas local adotado
Fonte: Adaptado do material de aula do professor

de deslocamentos unitários para cada uma das coordenadas restringindo as demais e analisando as forças externas que atuam sobre o elemento para gerá-los.

Esta matriz pode ser interpretada fisicamente como modelo matemático da estrutura, à medida define como ela reage com as forças a partir dos deslocamentos aplicados. Assim, obtém-se para a matriz de rigidez do elemento escolhido:

$$[r_e]_i = \begin{pmatrix} \frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{-2EI}{L} \\ \frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & -\frac{2EI}{L} & 0 & \frac{6EI}{L^2} & \frac{4EI}{L} \end{pmatrix}$$

Contudo, quem sofre as solicitações é a estrutura como um todo, sendo necessário definir, não apenas a matriz de rigidez local, mas também a global. A partir daí, surge o conceito de matriz de incidência, objeto que modela as transformações das coordenadas locais para globais, que podem apresentar-se em duas formas: cinemática, relativa aos deslocamentos, e estática, aos carregamentos.

Para o processo dos deslocamentos, torna-se relevante definir a matriz de incidência cinemática. Para facilitar a aplicação e reduzir o custo computacional com multiplicação de matrizes, será realizada apenas a rotação das coordenadas via matriz de incidência, definindo-a como:

$$[\beta_e]_i = \begin{pmatrix} -\cos\theta & -\sin\theta & 0 & 0 & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\theta & \sin\theta & 0 \\ 0 & 0 & 0 & \sin\theta & -\cos\theta & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

onde θ corresponde ao ângulo de rotação da barra

a partir da direção horizontal no sentido anti-horário.

Deste modo, a matriz de rigidez global do elemento é definida pelo triplo produto matricial $[r_g]_i = [\beta_e]_i^T [r_e]_i [\beta_e]_i$ e suas coordenadas são transferidas para sua respectiva posição na matriz de rigidez local $[R]$ diretamente por associação do valor na sua respectiva posição, de tal forma que:

$$[R]_i = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \\ 0 & 0 & 0 & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 0 & 0 & 0 & \\ 0 & 0 & 0 & \dots & r_{g11} & r_{g12} & r_{g13} & r_{g14} & r_{g15} & r_{g16} & \dots & 0 & 0 & 0 & \text{J1} \\ 0 & 0 & 0 & \dots & r_{g21} & r_{g22} & r_{g23} & r_{g24} & r_{g25} & r_{g26} & \dots & 0 & 0 & 0 & \text{J2} \\ 0 & 0 & 0 & \dots & r_{g31} & r_{g32} & r_{g33} & r_{g34} & r_{g35} & r_{g36} & \dots & 0 & 0 & 0 & \text{J3} \\ 0 & 0 & 0 & \dots & r_{g41} & r_{g42} & r_{g43} & r_{g44} & r_{g45} & r_{g46} & \dots & 0 & 0 & 0 & \text{K1} \\ 0 & 0 & 0 & \dots & r_{g51} & r_{g52} & r_{g53} & r_{g54} & r_{g55} & r_{g56} & \dots & 0 & 0 & 0 & \text{K2} \\ 0 & 0 & 0 & \dots & r_{g61} & r_{g62} & r_{g63} & r_{g64} & r_{g65} & r_{g66} & \dots & 0 & 0 & 0 & \text{K3} \\ 0 & 0 & 0 & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 0 & 0 & 0 & \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \\ & & & & \text{J1} & \text{J2} & \text{J3} & \text{K1} & \text{K2} & \text{K3} & & & & & \end{pmatrix}$$

onde J1, J2 e J3 são as coordenadas referentes ao nó j do elemento
e K1, K2 e K3, referentes ao nó k.

Assim, a matriz de rigidez é a soma das contribuições de cada elemento $[R]_i$. Daí então, parte-se para a definição dos vetores de ações equivalentes.

Como as coordenadas globais leem apenas as forças aplicadas nos nós, é preciso determinar cargas concentradas que gerem efeitos equivalentes aos carregamentos distribuídos. Isso é considerado aplicando-os ao elemento e transformando-os ações em coordenadas locais com os mesmos valores de reações obtidas, porém na direção oposta, ou seja, o simétrico do valor da reação, e transformando este vetor para as coordenadas locais com o inverso da matriz de incidência estática que, pelo Teorema da Contragradiência (ou de Chebsch) equivale a $[\beta_e]^T$.

$$\begin{Bmatrix} qxe \\ qye \end{Bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix} \begin{Bmatrix} qx \\ qy \end{Bmatrix} \rightarrow \begin{cases} qxe = qx\cos\theta + qy\sin\theta \\ qye = qx\sin\theta - qy\cos\theta \end{cases}$$

onde qxe e qye são os carregamentos calculados
perpendiular e paralelo ao eixo do elemento

$$P_{ne_e} = \left\{ \begin{array}{c} \frac{qxeL}{2} \\ \frac{qyeL}{2} \\ -\frac{qyeL^2}{12} \\ -\frac{qxeL}{2} \\ -\frac{qyeL}{2} \\ -\frac{qyeL^2}{12} \end{array} \right\} \rightarrow \{F\}^{ne} = [\beta_e]^T \{P\}^{ne} \rightarrow \boxed{\{F\} = \{F\}^n + \{F\}^{ne}}$$

Finalmente com o vetor de forças e a matriz de rigidez definidos, basta aplicar as condições de contorno para cada nó e resolver a equação fundamental do Método da Rigidez:

$$\{F\} = [R]\{u\} \rightarrow \{u\} = [R]^{-1}\{F\}$$

Para impor as condições, porém, existem basicamente três estratégias: Reordenação do sistema - trocando linhas e colunas, mas que toma um tempo computacional excessivo; Particionamento direto - criando um vetor que define o formato final do sistema particionado e monta a matriz de rigidez e demais vetores diretamente; e a Técnica do Pênalti - removendo momentaneamente as linhas e colunas da matriz de rigidez e a coordenada do vetor de forças.

Optou-se então por implementar a Técnica do Pênalti, multiplicando-se o elemento da diagonal principal da matriz de rigidez e a correspondente coordenada do vetor de forças por 10^6.

Por fim, com o vetor de deslocamentos globais armazenado, fez-se transformação deles para as coordenadas locais de cada elemento e obteve-se de volta o vetor de ações em coordenadas locais, que conforme definido no início do processo, representam os esforços internos solicitantes nas extremidades das barras e as reações de apoio.

3 Implementação do programa

Conforme especificado anteriormente, o programa foi implementado em Python. Para isso, foram utilizados como biblioteca externas:

- ***numpy***, para definição dos vetores, matrizes e devidas manipulações desses elementos como produto e resolução de sistemas lineares;
- ***pandas***, para pegar os dados a partir do arquivo de entrada em Excel e imprimir na tela os dados de resultados organizados;
- ***tkinter***, para criação e manipulação de interface gráfica e desenho da estrutura e dos diagramas.

Para fins de organização o programa foi subdividido em 6 módulos. São eles: main, EntDados, ConfirmEstrut, AnMatric, Resultados e MostDiag. Cada um deles possui uma função específica. As próximas subseções apresentam uma síntese de organização do código. Este, porém, é aberto e encontra-se comentado, informando de maneira resumida a função atribuída a cada bloco de comando.

3.1 Módulo main

Trata-se do módulo principal. Nele são determinados, através de uma pequena caixa de diálogo, o caminho do arquivo Excel de entrada de dados e a opção de exibir ou não os diagramas de esforços internos solicitantes e as reações de apoio.

Por definição, o programa varre todos os arquivos com extensão *.xlsx* e *.xls* existentes na pasta *ent* no mesmo diretório do programa. Entretanto, o usuário pode especificar outro arquivo, digitando ou colando o caminho completo na caixa de texto.

Além disso, ele é responsável por chamar todos os demais módulos na sua respectiva ordem e condição de chamada, verificando também a existência do arquivo no caminho especificado.

3.2 Módulo EntDados

Verificada a existência do arquivo, o primeiro módulo executado é o *EntDados.py*. Para o funcionamento adequado do programa, deve ser um arquivo Excel, com extensão .xlsx, criado a partir do arquivo *PlanilhaModelo.xlsx*

Recebendo o caminho do arquivo, este é responsável por abrí-lo, percorrê-lo pegando todos os dados necessários para execução e manipulação destes, extraindo-os com o auxílio da bilioteca *pandas* e armazenando-os em vetores definidos com *numpy*.

Após obtê-los, faz-se uma verificação da quantidade para verificar a consistência dos dados especificados, verificando se a quantidade de linhas passadas para os nós e elementos estão compatíveis com o que foi atribuído na primeira tabela. Caso isso não ocorra, uma mensagem de erro será exibida e o erro correspondente, mostrado no Terminal.

Além dos dados obtidos por meio do arquivo, são calculados por esse módulo os comprimentos e os valores de seno e cosseno de cada barra a partir das coordenadas dos pontos. E o programa segue adiante.

3.3 Módulo ConfirmEstrut

De posse de todos os dados necessários, o programa chama o *ConfirmEstrut.py*.

Usando a biblioteca *tkinter*, este módulo cria uma janela com um elemento de *Canvas*, que é útil para criação de desenhos a partir de objetos como linhas, textos, imagens, elipses, arcos, entre outros.

Assim, com todos os vetores que a definem estabelecidos no módulo anterior, a estrutura é desenhada e a caixa de diálogo apresenta duas opções: *Continuar análise* ou *Redefinir Estrutura*.

Caso a opção escolhida seja de redefinir a estrutura, o programa será encerrado e o arquivo de entrada de dados deve ser convenientemente ajustado. Caso contrário, o programa dará início à análise.

3.4 Módulo AnMatric

O módulo inicia com a definição da matriz de rigidez global a partir do procedimento do Método da Rigidez Direta definindo a matriz local de um elemento básico e fazendo as devidas transformações para a matriz global conforme explicado anteriormente.

Depois, têm-se a determinação do vetor de forças considerando as ações equivalentes aos carregamentos lineares, transformando as forças distribuídas em ações locais no elemento e, assim como a matriz de rigidez, são feitas as transformações para as coordenadas globais.

Por fim, utilizando-se a Técnica do Pênalti para aplicar as restrições de deslocamentos indicadas pelas condições de contorno dos nós, resolve-se o sistema linear, obtendo o vetor de deslocamentos, finalizando o módulo em questão.

3.5 Módulo Resultados

Com o vetor de deslocamentos globais calculado, o módulo *Resultados.py* é responsável por calcular os vetores de ações locais de cada elemento que, conforme definido anteriormente,

representam os esforços internos solicitantes na sua respectiva convenção de sinais convenientemente adotada.

Além disso, com o vetor de forças em coordenadas locais, pode-se transformá-la para as coordenadas globais pelo Teorema da Contragradiência (ou de Clebsch), tendo portanto o valor das reações de apoio em cada ponto.

Com esses valores devidamente armazenados em *arrays* definidos pela biblioteca *numpy*, o programa organiza-os em *dataframes* (tipos de variáveis de tabelas definidos pela biblioteca *pandas*), exibe esses dados no terminal e os imprime num arquivo *txt* com o mesmo nome do arquivo de entrada de dados original na pasta *out* do programa.

3.6 Módulo MostDiag

Este, por fim, exibe na tela uma janela interativa onde são desenhados a estrutura e, conforme determinado pelo usuário, os esforços internos solicitantes e as reações de apoio de maneira similar ao que acontece no módulo *ConfirmEstrut.py*.

Vale salientar ainda que, a fim de evitar problemas com relação ao tamanho dos diagramas, foi criado um objeto deslizante que controla a escala do diagrama também de forma interativa para que facilitar a visualização dos resultados.

4 Tutorial

Nesta seção serão discutidas as diretrizes de execução para o funcionamento eficiente do programa. O fluxo de trabalho foi pensado para ser o mais simples possível e é baseado no fluxograma representado na Figura 3.

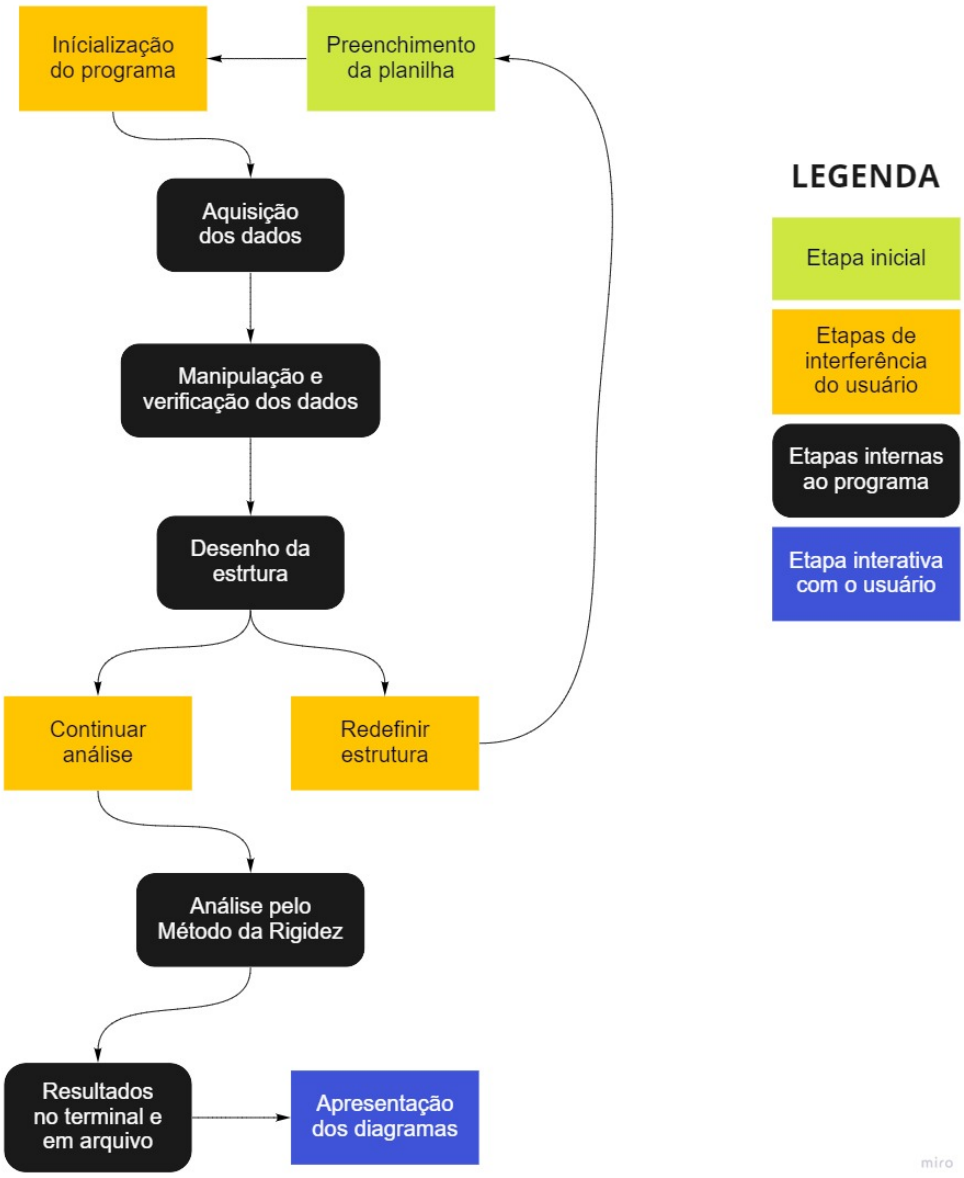

Figura 3: Fluxograma de trabalho do programa
Fonte: Autor

4.1 Preenchimento da planilha

Para facilitar o processo de utilização do programa, a entrada de dados é feita via arquivo do software Microsoft Excel. Assim, é disponibilizado duas planilhas denominadas ”PlanilhaModelo”, uma com extensão *.xlsx* e outra com *.xls*, para cobrir versões mais antigas do software,

que devem ser copiadas, renomeadas e alteradas conforme a necessidade do usuário.

O arquivo possui três planilhas. A primeira delas, denominada ”Qtdades” (Figura 4), recebem a quantidade de nós e de elementos da estrutura, passando o valor na célula abaixo da respectiva informação. Este dado é importante para verificação da consistência dos dados incluídos, definir o tamanho de vetores dentro do código.

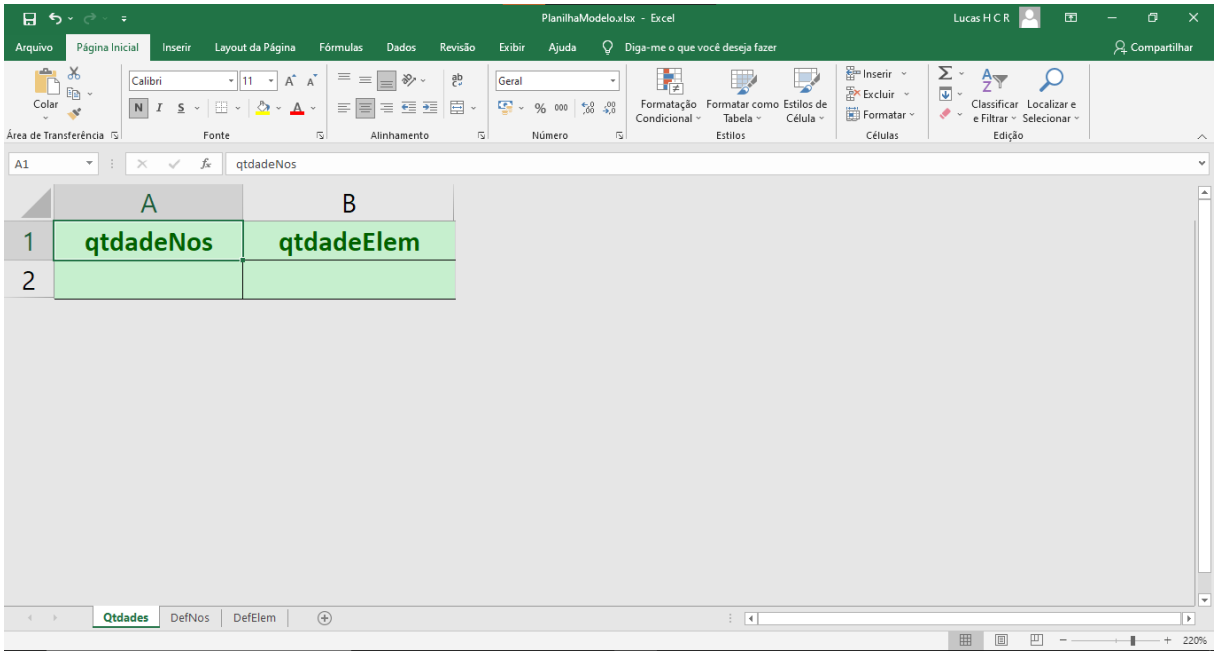

Figura 4: Planilha Qtdades
Fonte: Autor

Por conseguinte, tem-se a planilha DefNos (Figura 5) que define as informações relacionadas a cada nó da estrutura. De maneira análoga à anterior, a informação é dada nas células inferiores ao seu respectivo indicador, de tal forma que cada linha da tabela corresponderá a um nó.

Com relação as informações, têm-se:

- ***Posição do nó (x, y)*** **-** Recebe a posição horizontal (x) e vertical (y) de cada um dos nós com relação a um sistema de referência definido pelo usuário.

  Para o melhor aproveitamento dos módulos de desenho, é conveniente que o sistema de coordenadas seja definido de modo que a estrutura localize-se inteiramente no primeiro quadrante, ou seja, com a origem no seu limite inferior esquerdo, ou ainda, de modo que todos valores de x e y sejam positivos.

- ***Deslocamentos nodais (ux, uy, uz)*** **-** Recebem as informações com relação às condições de contorno dos deslocamentos horizontal (ux), vertical (uy) e rotacional (uz).

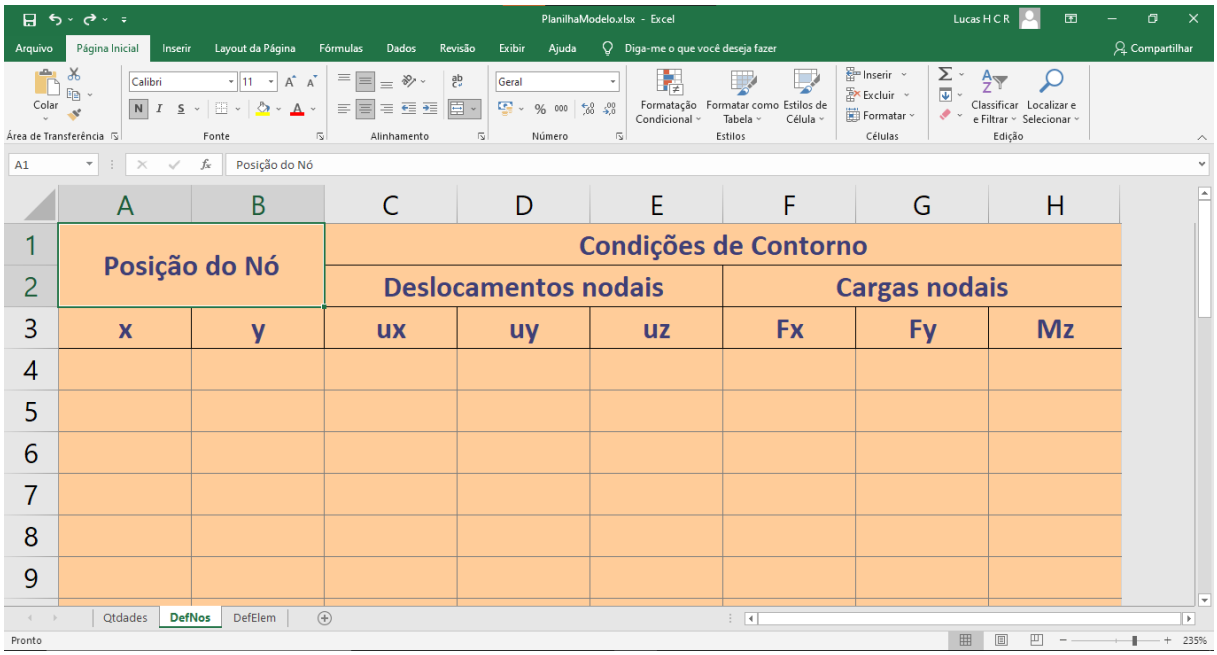

Figura 5: Planilha DefNos
Fonte: Autor

Este valor é definido a depender da restrição de deslocamento do ponto na direção correspondente. Para coordenadas cujo deslocamento seja restrito deve ser atribuído 1 a esse campo. Caso seja livre, atribui-se 0.

- ***Cargas nodais (Fx, Fy, Mz)*** **-** Recebem as informações com relação às condições de contorno dos carregamentos concentrados horizontal (Fx), vertical (Fy) e rotacional (Mz).

  Ou seja, estes campos devem ser preenchidos com os eventuais valores de forças pontuais definidas na estrutura seguindo-se a convenção de sinal positivo de Fx para a direita, de Fy para cima e Mz no sentido anti-horário (Figura 6)

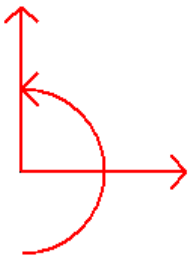

Figura 6: Convenção de sinais positiva para as cargas nodais
Fonte: Autor

Já a última planilha, definida como DefElem (Figura 7), é semelhante à anterior e serve para a atribuição dos parâmetros dos elementos de barras que compõem a estrutura. São eles:

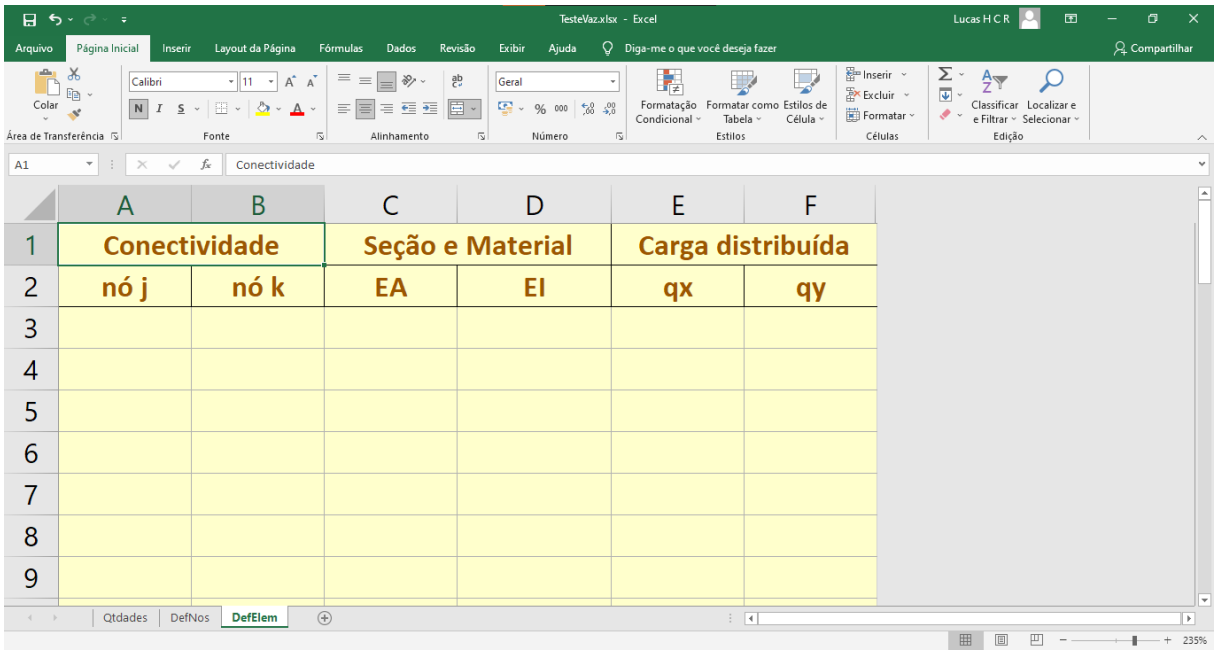

Figura 7: Planilha DefElem
Fonte: Autor

- ***Conectividade (j, k)*** **-** Recebe o índice dos nós anterior (j) e posterior (k) de cada barra, conectando-os e definindo o elemento. A indexação é realizada a partir da entrada dos nós conforme o item anterior, de modo que o primeiro nó recebe índice 1, o segundo, 2 e assim por diante.

  Por questões de direcionamento do elemento, é indicado usar sempre as definições das conectividade da esquerda para a direita e de baixo para cima, de modo que as definições de esquerda e direita sejam padronizadas, bem como as direções positivas e negativas do diagramas, ou seja, $j < k$.

- ***Seção e material (EA, EI)*** **-** Recebem os parâmetros geométricos e relativos ao material do elemento através do módulo de rigidez axial (EA) e à flexão (EI).

- ***Carga distribuída (qx, qy)*** **-** Recebem os valores dos carregamentos uniformemente distribuídos. Estes foram definidos como distribuídos ao longo do comprimento da barra e positivos para baixo e para a direita, conforme a orientação indicada na Figura 8. Quaisquer carregamentos diferente do especificado devem ser realizadas as devidas manipulações para utilização no programa.

OBS: Com relação às unidades de medidas, não há restrições desde que elas estejam padronizadas dentro de um mesmo sistema e não hajam números muito grandes a fim de evitar instabilidades numéricas

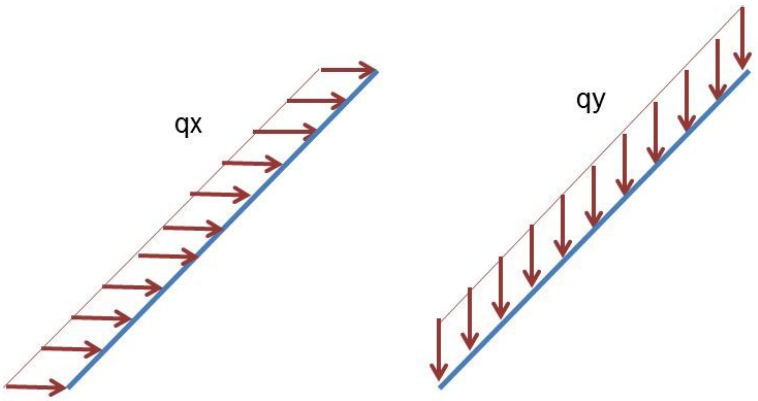

Figura 8: Convenção de sinais positiva para os carregamentos distribuídos
Fonte: Material de aula do professor

4.2 Inicialização do programa

Com o arquivo preenchido, o usuário pode inicializar o programa abrindo o módulo *main.py*. Para isso, é importante que a linguagem Python esteja devidamente instalada no dispositivo.

Ao fazê-lo, será exibida uma caixa de diálogo (Figura 9). Nela, o usuário deve marcar ou desmarcar a opção de gerar os diagramas ao final da análise e passar para o programa o arquivo de entrada de dados preenchido conforme explicado anteriormente.

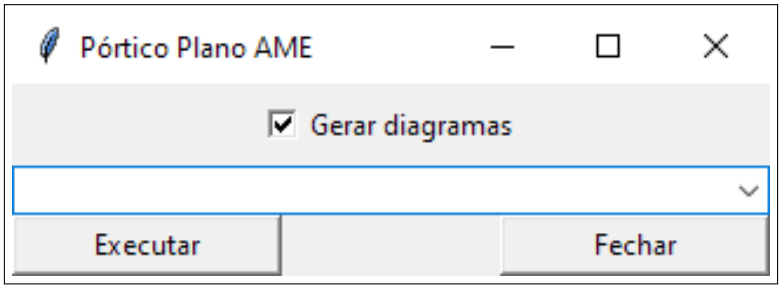

Figura 9: Janela de entrada de dados do programa
Fonte: Autor

Para isso, o usuário tem duas opções. A primeira é colocar o arquivo dentro da pasta *ent* no diretório do programa para que este arquivo seja lido e apareça na caixa de seleção. Além disso, o diretório completo do arquivo pode ser passado na caixa de seleção como um texto.

Por fim, clicando no botão *Executar*, o programa abre o arquivo e obtém os dados necessários. Caso haja algum problema durante a execução, será exibida uma mensagem de erro e o terminal pode indicar se o problema está na definição dos nós ou elementos.

4.3 Confirmando a estrutura

Com todos os dados definidos e armazenados, o programa abrirá uma janela onde a estrutura será desenhada e exibida conforme os dados inseridos. Caso a estrutura esteja de acordo com o

esperado, basta clicar no botão *Continuar Análise!* para prosseguir. Caso contrário, clicando em *Redefinir estrutura!*, o programa será encerrado e o usuário deve verificar o arquivo de entrada.

4.4 Resultado

Finalmente, ao fim do programa, os dados de esforços internos solicitantes, reações de apoio e deslocamentos são dispostos em forma de tabelas, exibidos no terminal e salvos em um arquivo *txt* com o mesmo nome do arquivo de entrada e, caso a opção tenha sido habilitada na janela inicial, o programa abrirá uma janela interativa onde o usuário poderá visualizar os diagramas gerados.

5 Exemplos de aplicação

A fim de demonstrar a aplicação em funcionamento e para validação dos resultados obtidos a partir dela, serão apresentados dois exemplos de pórticos planos, que serão disponibilizados juntos com a pasta do programa a fim de testar o programa e entender seu funcionamento.

5.1 Exemplo 1

O primeiro exemplo trata-se de um pórtico simples, hiperestático, formado por três nós e duas barras submetido a um carregamento distribuído e dois pontuais, sendo um de força vertical para baixo e outro de momento no sentido anti-horário, conforme Figura 10.

Adotou-se as dimensões de força em quiloNewton (kN) e as espaciais em metro (m) com módulo de rigidez à flexão (EI) igual a de 180kNm² e considerando a área infinita, tomando (EA) igual a 1×10^6.

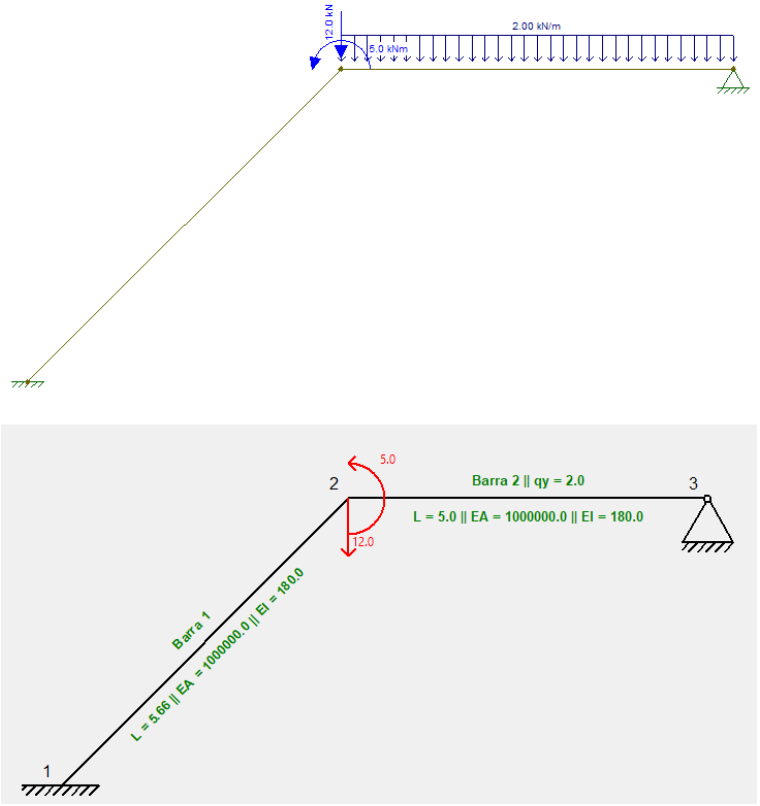

Figura 10: Comparativo entre as imagens da estrutura gerada pelo FTOOL (acima) e PPAME (abaixo)
Fonte: Autor

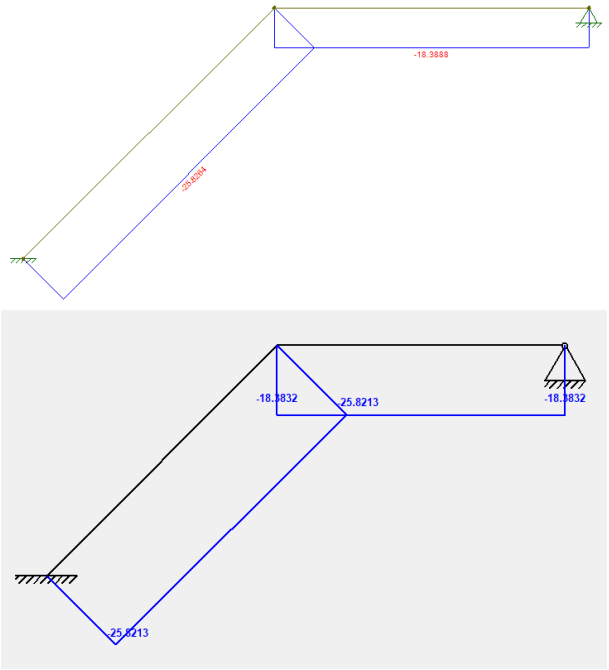

Figura 11: Comparativo entre os diagramas de esforço normal gerados pelo FTOOL (acima) e PPAME (abaixo)
Fonte: Autor

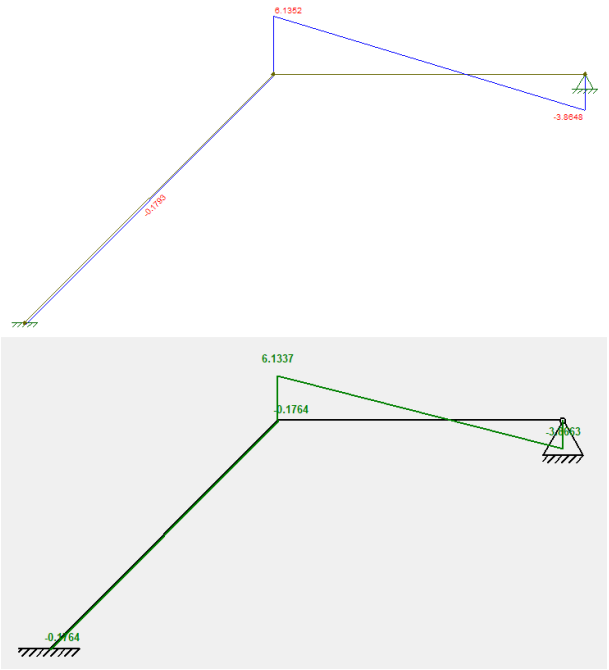

Figura 12: Comparativo entre os diagramas de esforço cortante gerados pelo FTOOL (acima) e PPAME (abaixo)
Fonte: Autor

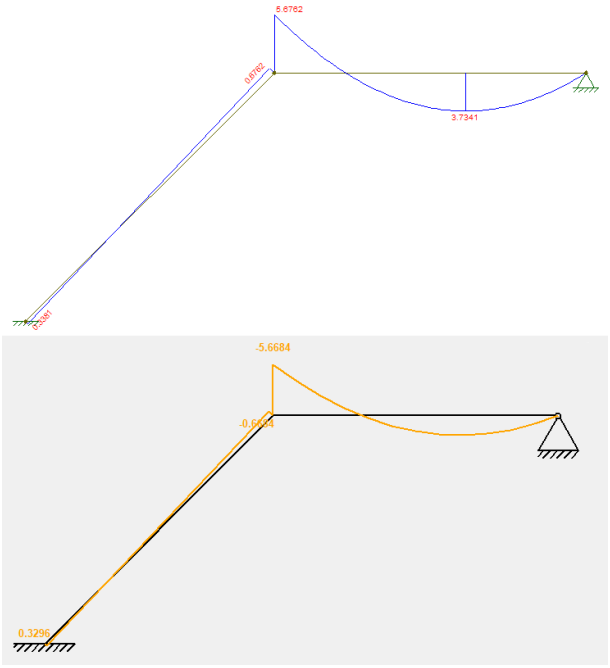

Figura 13: Comparativo entre os diagramas de momento fletor gerados pelo FTOOL (acima) e PPAME (abaixo)
Fonte: Autor

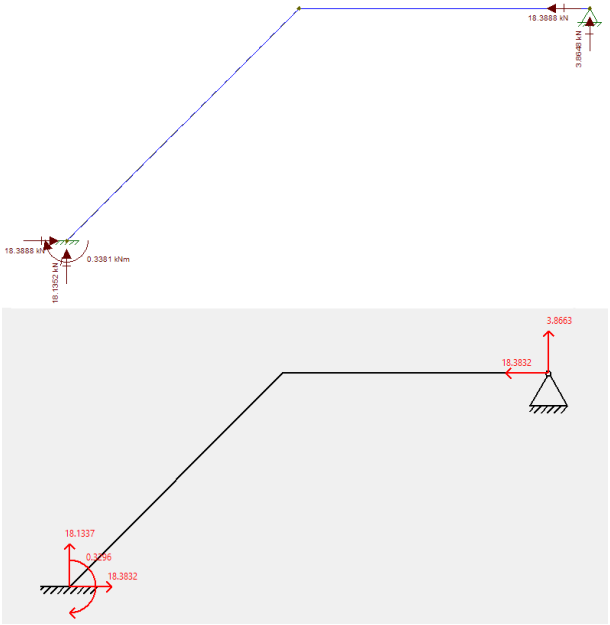

Figura 14: Comparativo entre as reações de apoio gerados pelo FTOOL (acima) e PPAME (abaixo)
Fonte: Autor

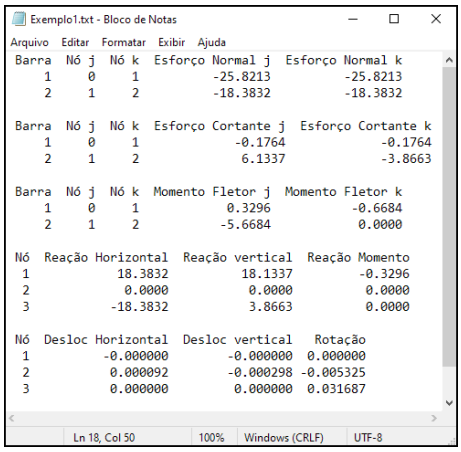

Figura 15: Arquivo *.txt* gerado pelo PPAME
Fonte: Autor

Comparando então os resultados obtidos com os dois programas, pode-se perceber a similaridade entre os resultados pelos diagramas e reações de apoio apresentados, exceto por uma pequena diferença a partir da terceira casa decimal, mostrando a consistência do programa.

5.2 Exemplo 2

O segundo exemplo, porém, modela um pórtico um pouco mais elaborado, também hiperestático, com uma célula de pórtico, formado por seis nós e seis barras submetido a dois carregamento distribuídos e um pontual horizontal para direita em um dos nós, conforme Figura 10.

Adotou-se novamente as dimensões de força em quiloNewton (kN) e as espaciais em metro (m) com um valor genérico para o módulo de rigidez à flexão (EI) igual a de 1kNm² e considerando a área infinita, tomando (EA) igual a 1×10^6

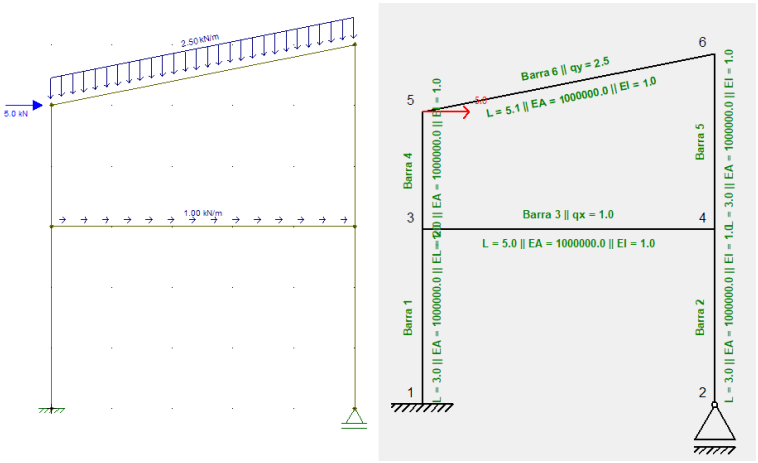

Figura 16: Comparativo entre as imagens da estrutura geradas pelo FTOOL (acima) e PPAME (abaixo)
Fonte: Autor

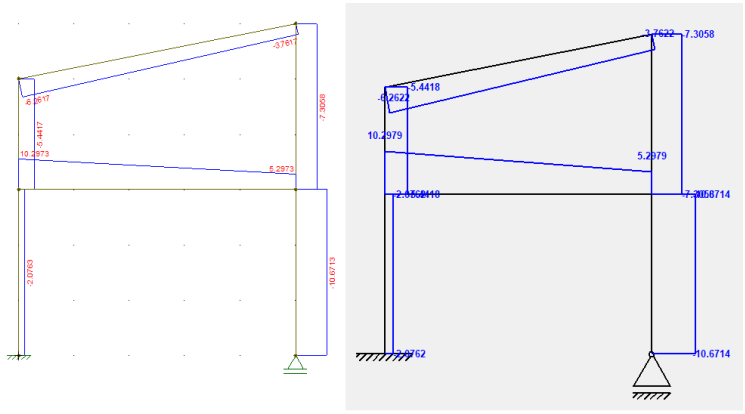

Figura 17: Comparativo entre os diagramas de esforço normal gerados pelo FTOOL (acima) e PPAME (abaixo)
Fonte: Autor

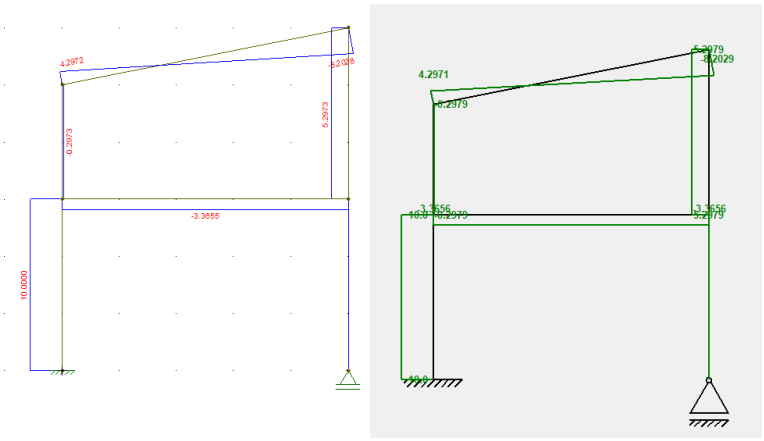

Figura 18: Comparativo entre os diagramas de esforço cortante gerados pelo FTOOL (acima) e PPAME (abaixo)
Fonte: Autor

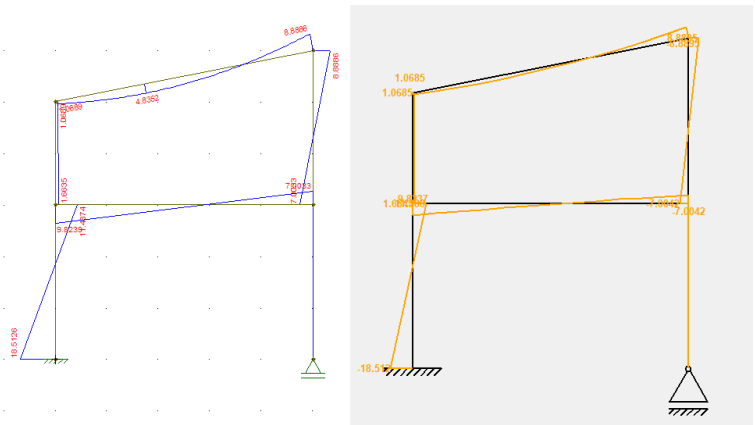

Figura 19: Comparativo entre os diagramas de momento fletor gerados pelo FTOOL (acima) e PPAME (abaixo)
Fonte: Autor

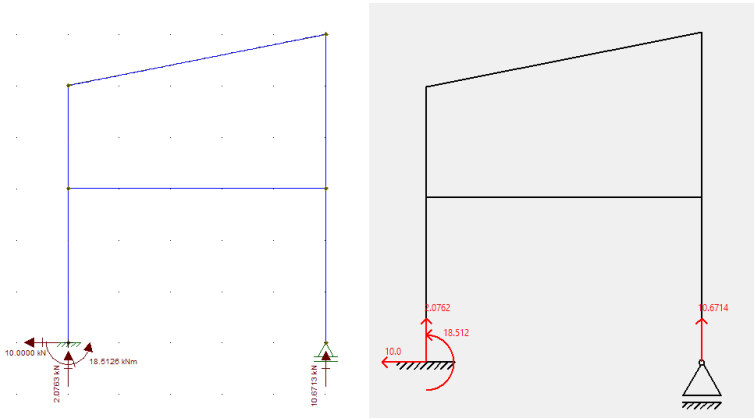

Figura 20: Comparativo entre as reações de apoio gerados pelo FTOOL (acima) e PPAME (abaixo)
Fonte: Autor

Exemplo2.txt - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

Barra	Nó j	Nó k	Esforço Normal j	Esforço Normal k
1	1	3	-2.0762	-2.0762
2	2	4	-10.6714	-10.6714
3	3	4	10.2979	5.2979
4	3	5	-5.4418	-5.4418
5	4	6	-7.3058	-7.3058
6	5	6	-6.2622	-3.7622

Barra	Nó j	Nó k	Esforço Cortante j	Esforço Cortante k
1	1	3	10.0000	10.0000
2	2	4	-0.0000	-0.0000
3	3	4	-3.3656	-3.3656
4	3	5	-0.2979	-0.2979
5	4	6	5.2979	5.2979
6	5	6	4.2971	-8.2029

Barra	Nó j	Nó k	Momento Fletor j	Momento Fletor k
1	1	3	-18.5120	11.4880
2	2	4	0.0000	-0.0000
3	3	4	9.8237	-7.0042
4	3	5	1.6643	1.0685
5	4	6	-7.0042	8.8895
6	5	6	1.0685	-8.8895

Nó	Reação Horizontal	Reação vertical	Reação Momento
1	-10.0	2.0762	18.512
2	0.0	10.6714	0.000
3	0.0	0.0000	-0.000
4	-0.0	-0.0000	-0.000
5	0.0	0.0000	-0.000
6	-0.0	-0.0000	-0.000

Nó	Desloc Horizontal	Desloc vertical	Rotação
1	0.000000	-0.000000	-0.000000
2	27.842036	-0.000000	-3.487322
3	38.303964	-0.000006	-10.535976
4	38.304003	-0.000032	-3.487322
5	56.444512	-0.000017	-7.803174
6	56.444493	-0.000054	-0.659483

Ln 1, Col 1 100% Windows (CRLF) UTF-8

Figura 21: Arquivo *.txt* gerado pelo PPAME
Fonte: Autor

Mais uma vez, analisando os resultados obtidos, pode-se perceber que os resultados apresentados são equivalentes, exceto por flutuações a partir da terceira casa decimal, tal como o exemplo anterior.

6 Conclusão

Este artigo, portanto, permitiu contemplar a aplicação dos conhecimentos adquiridos pelo aluno na disciplina de Análise Matricial de Estruturas a partir da elaboração do programa, tendo-se um entendimento do funcionamento de um grande parcela de software de análise estrutural.

Isso tem sua importância à medida que cabe ao engenheiro não apenas saber utilizar um programa, mas compreender a tecnologia a fundo com o intuito de ter uma visão crítica com relação aos resultados disponibilizados por ele e, para isso, é necessário conhecer como estes resultados são gerados.

Além disso, a trabalho foi muito eficaz para compreender o procedimento de cálculo do Método da Rigidez Direta, que é amplamente utilizado na análise estrutural, em especial na implementação computacional.