**Universidade Federal de Alagoas – UFAL**
**Centro de Tecnologia – CTEC**
**Programa de Pós-Graduação em Engenharia Civil – PPGEC**

**Disciplina:** Mecânica Computacional das Estruturas (EES103)
**Professor:** Eduardo Toledo de Lima Junior
**Período Letivo:** 2024.1

# Guia e Relatório da aplicação *Truss2D_M*

**Data de Recolhimento:** 11/10/2024

**Discente:** Milton Mateus Guimarães dos Santos
**Matrícula:** 2024109379

Maceió, 10 de Outubro de 2024

**Discente:** Milton Mateus Guimarães dos Santos
**Matrícula:** 2024109379

# Guia e Relatório da aplicação *Truss2D_M*

Trabalho realizado na disciplina de Mecânica Computacional das Estruturas, do Programa de Pós-Graduação em Engenharia Civil (PPGEC-UFAL), sob supervisão do Professor Eduardo Toledo de Lima Junior.

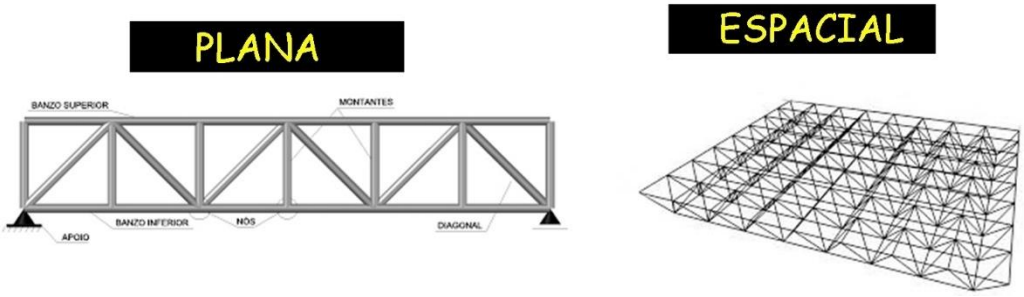
Maceió, 10 de Outubro de 2024

## SUMÁRIO

1 INTRODUÇÃO ..... 4
2 OBJETIVOS ..... 5
3 CONSIDERAÇÕES INICAIS ..... 5
4 ENTRADA DE DADOS ..... 5
5 PRÉ-ANÁLISE DA ESTRUTURA ..... 7
6 ANÁLISE ESTRUTURAL ..... 8
6.1 Matriz de Rigidez da Estrutura ..... 8
6.2 Determinação dos deslocamentos nodais ..... 10
6.3 Determinação dos esforços internos solicitantes e reações de apoio ..... 10
7 EXECUÇÃO DO PROGRAMA E SAÍDA DE DADOS ..... 10
8 VALIDAÇÃO DA APLICAÇÃO ..... 11

# 1 INTRODUÇÃO

As treliças são estruturas formadas pela associação de elementos de barra de eixo reto projetados para responder axialmente às solicitações impostas à estrutura por meio de carregamentos nodais aplicados à estrutura analisada. Neste tipo de estrutura, os elementos, como dito outrora, são projetodos para responder unicamente à solicitações axiais devido ao tipo de ligação imposta nos nós (rotulada) que interligam os elementos de barra, a qual é isenta de rigidez rotacional. Assim, devido a tal característica, a estrutura garante a estabilidade e resistência apenas pela resposta mecânica normal do elementos de barra, sem quaisquer contribuições de respostas à esforços cortantes e de flexão. As treliças podem ser comcebidas de forma plana ou espacial (Figura 1) a depender de sua funcionalidade, de modo que as treliças planas são amplamente usadas em estruturas metálicas de galpões e tesouras de telhados (Figura 2) enquanto as treliças espaciais são mais utilizadas em telhados especiais (Figura 3.a), torres de transmissão (Figura 3.b) e obras de arte (Figura 4).




( a ) ( b )

Figura 1: Exemplos de treliças plana (a) e espacial (b).

( a ) ( b )

Figura 2: Treliça plana em estruturas metálicas de galpões (a) e em tesouras de telhados (b).

( a ) ( b )

Figura 3: Treliça plana em estruturas metálicas de galpões (a) e em tesouras de telhados (b).

Figura 4: Torre Eiffel como exemplificação de treliças espaciais aplicadas em obras de arte.

# 2 OBJETIVOS

O trabalho em questão visa o desenvolvimento de um programa, desenvolvido em linguagem python, que permita a análise de treliças planas, tipo de estrutura exemplificada na Figura 1.a. A aplicação permite analisar treliças isostáticas e hiperestáticas com ligações por nós rolutados para obtenção do comportamento da estrutura analisada, tais como: reações de apoio, esforços normais e deslocamentos da estrutura. O programa em questão foi desenvolvido com base no conteúdo ministrado na disciplina "Mecânica Computacional das Estruturas" e nos livros Matrix Analysis of Structures (Aslam Kassimali, 2010) e Análise Matricil de Estruturas com Orientação à Objetos (Martha, 2019), de modo a ter um referencial teórico necessário para o desenvolvimento do aplicação *Truss2D_M*. Apesar deste documento apresentará brevemente o referencial teórico utilizado para produção da aplicação, este documento tem enfoque sobre a construção do arquivo de entrada de dados, interpretação dos arquivos *output* e validação dos resultados obtidos pela aplicação.

# 3 CONSIDERAÇÕES INICAIS

A aplicação desenvolvida teve como base a linguagem *python.* Além disso, algumas bibliotecas externas foram utilizadas para seu desenvolvimento, portanto para executar a aplicação *Truss2D_M* é necessário que, além da instalação da própria linguagem *python*, as seguintes bibliotecas sejam instaladas previamente: *numpy* e *matplotlib*. A aplicação não possui interface gráfica, portanto deve ser executada em linha de comando

# 4 ENTRADA DE DADOS

Para realizar a análise da estrutura pelo programa *Truss2D_M,* são necessários dois passos: construir o arquivo de entrada com os dados, informando sua geometria, propriedades físicas e geométricas, bem como seus vínculos (condições de contorno) e seus carregamentos, e a execução do programa vinculando o arquivo de dados.
Para construir a entrada de dados, deve-se criar um arquivo de extensão "*.json"* com as informações necessárias para definir a estrutura. O formato JSON (JavaScript Object Notation) é uma forma de notação de objetos *JavaScript*, que pode representados de forma comum a diversas linguagens com

a estruturação de dados em listas e dicionários. Dessa forma, os dados referentes à estruturas deverão ser definidos em um arquivo em formato JSON, os quais são divididos nas seguintes classes:

- *typeStructures*: Define o tipo de estrutura a ser analisada (neste caso "Truss2D");
- *nodes*: Define uma lista de dicionários referentes aos nós da estrutura com as seguintes chaves:
  - *id*: Identificador do nó (inserir valor inteiro);
  - *coordenadas*: Vetor de ordem 2 com as coordenadas x e y, respectivamente, em metros;
- *material*: Define uma lista de materiais a serem utilizados na estrutura com as seguintes chaves:
  - *id*: Identificador do material (inserir valor inteiro);
  - *E*: módulo de elasticidade de Young $[kN/m^2]$;
- *sectionProp*: Define uma lista de propiedades geométricas de uma seção a ser utilizada na estrutura com as seguintes chaves:
  - *id*: Identificador da seção transversal (inserir valor inteiro);
  - *A*: Área da seção transversal $[m^2]$;
- *elements*: Define uma lista de dicionários referentes aos elementos da estrutura com as seguintes chaves:
  - *id*: Identificador do elemento;
  - *NI*: Identificador *id* do nó inicial do elemento;
  - *NF*: Identificador *id* do nó final do elemento;
  - *material*: Identificador *id* do material do elemento;
  - *sectionProp*: Identificador *id* do elemento;
- *nodalLoads*: Define uma lista de ações nodais a serem aplicadas na estrutura, com as seguintes chaves:
  - *id*: Identificador *id* do nó ao qual será aplicado (inserir valor inteiro);
  - *forcas*: Vetor de ordem 2 de forças nodais a ser aplicado, nas direções x e y $[kN]$;
- *restrictions*: Define uma lista de dicionários referentes às restrições da estrutura com as seguintes chave:
  - *no*: Identificador *id* do nó a ser aplicado a restrição;
  - *restricoes*: Vetor binário de ordem 2, no qual define-se em qual grau de liberdade haverá restrição (zero para não aplicar restrições e um para aplicar restrições);

- *types*: Lista de ordem 2 para definir o tipo de restrição aplicada à cada grau de liberdade, as quais podem ser: "*fix*" para aplicar um apoio, "*prescrible*" para aplicar um deslocamento prescrito ao nó e *"flexible"* para aplicar um apoio flexível (mola) no grau de liberdade de interesse;
- *flexible*: esta chave condicional deve ser adicionada se houver apoios flexíveis (restrição *"flexible"*) no nó. Deve-se inserir um vetor com as constantes de mola, em kN/m, de cada vínculo flexível em um grau de liberdade, de forma sequencial;
- *prescrible*: esta chave condicional deve ser adicionada se houver prescrições de deslocamento (restrição *"prescrible"*) no nó. Deve-se inserir um vetor com as prescrições, em m, para cada grau de liberdade fixado com uma presrição, de forma sequencial;

A seguir, temos um exemplo de arquivo de entrada (Figura 5), em formato JSON, seguindo as diretrizes especificadas anteriormente.

```
{
    "typeStructure": "Truss2D",
    "nodes": [
        {"id": 1, "coordenadas": [0.0, 0.0]},
        {"id": 2, "coordenadas": [3.0, 0.0]},
        {"id": 3, "coordenadas": [1.5, 2.598076211353315940291169512258 8]},
        {"id": 4, "coordenadas": [4.5, 2.598076211353315940291169512258 8]}
    ],
    "elements": [
        {"id": 1, "NI": 1, "NF": 2, "material": 1, "sectionProp": 1},
        {"id": 2, "NI": 1, "NF": 3, "material": 1, "sectionProp": 2},
        {"id": 3, "NI": 2, "NF": 4, "material": 2, "sectionProp": 1},
        {"id": 4, "NI": 3, "NF": 2, "material": 2, "sectionProp": 2},
        {"id": 5, "NI": 4, "NF": 3, "material": 1, "sectionProp": 1}
    ],
    "material": [
        {"id": 1, "E": 205e6},
        {"id": 2, "E": 300e6}
    ],
    "sectionProp": [
        {"id": 1, "area": 6e-2},
        {"id": 2, "area": 10e-2}
    ],
    "restrictions": [
        {"no": 1, "restricoes": [1, 1], "types": ["flexible","flexible"], "flexible": [1.0e3, 1.0e2]},
        {"no": 4, "restricoes": [1, 1], "types": ["flexible","prescrible"], "flexible": [1.0e3], "prescrible": [-1.0e-4]}
    ],
    "nodalLoads": [
        {"no": 3, "forcas": [-5, -10]},
        {"no": 1, "forcas": [-5, -10]}
    ]
}
```

Figura 5: Exemplo de arquivo de entrada, visualizado no *Visual Studio Code*.

O mesmo arquivo de entrada pode ser visualizado utilizando o site "https://jsonviewer.stack.hu/" , o qual se trata simplesmente de um visualizador de JSON, que possibilita uma melhor compreensão dos dados presentes no arquivo de entrada.

# 5 PRÉ-ANÁLISE DA ESTRUTURA

A pré-análise da estrutura consiste na identificação dos graus de liberdade possíveis de um nó da

estrutura, logo para o caso de uma treliça plana, ocorre a existência de 2 graus de liberdades por nó. Assim, os $n$ nós da estrutura resultam em $2n$ graus de liberdade possíveis, os quais são identificados. Além disso, numera-se os graus de liberdade da estrutura que é definido por $2n - Nº\_Restrições$.

# 6 ANÁLISE ESTRUTURAL

## 6.1 Matriz de Rigidez da Estrutura

Para cada elemento da estrutura haverá uma matriz de rigidez interna $k_{int}$, a qual configura-se com base no sistema interno (ver Figura 6).

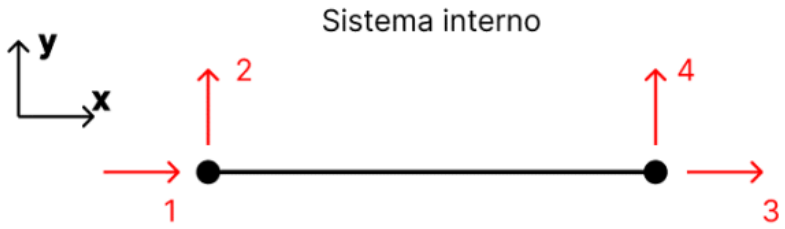



Figura 6: Elemento de barra genérico no sistema interno.

Ao experimentar deslocamentos em cada grau de liberdade, um vetor de forças reativas pode ser montado, o qual equivale à uma coluna da matriz de rigidez interna $k_I$. Contabilizando o a resposta mecânica à matriz de rigidez de interna expressa, podemos expressá-la por:

$$[k_I] = \begin{pmatrix} \frac{EA}{L} & 0 & -\frac{EA}{L} & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{EA}{L} & 0 & \frac{EA}{L} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Essa matriz deve ser transformada do sistema coordenado interno para o sistema local, como mostrado na Figura 7.

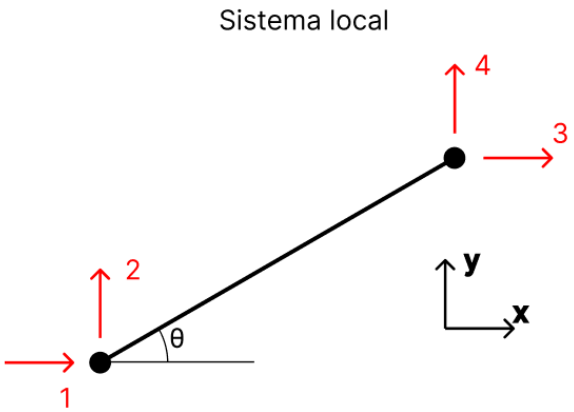



Figura 7: Elemento de barra genérico no sistema interno.

Para tal, utiliza-se a matriz de rotação $T$, apresentada a seguir,

$$[T] = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & \cos(\theta) & \sin(\theta) \\ 0 & 0 & -\sin(\theta) & \cos(\theta) \end{pmatrix}$$

na qual $\theta$ é o ângulo diretor do elemento discreto analisado. De posse transformação de rotação de coordenadas, transforma-se a matriz de rigidez interna para matriz de rigidez local $k_L$ através do triplo produto matricial

$$[k_L] = [T]^T \cdot [k] \cdot [T].$$

Por sua vez, para obter a matriz de rigidez global do elemento $[k_G]$, deve-se referenciar cada grau de liberdade local com os graus de liberdade global da estrutura por meio da Matriz de incidência Cinemática.

As Matrizes de Incidência Cinemática são montadas para cada elemento discreto, de ordem m por n, na qual m e n são os números de graus de liberdade do sistema local e global, respectivamente. A determinação dos elementos $\beta_{i,j}$ da matriz é realizada de modo que:

- $\beta_{i,j} = 0$ se os graus de liberdade local e global não possuem correspondência;
- $\beta_{i,j} = 1$ se os graus de liberdade local e global forem correspondentes e tiverem o mesmo sentido;
- $\beta_{i,j} = -1$ se os graus de liberdade local e global forem correspondentes mas tiverem sentidos opostos;

Com base na estratégia especificada neste passo, a Matriz de Incidência Cinemática $[\beta_p]$ de um elemento $p$ qualquer terá de ordem quatro por oito e será determinada verificando a correspondência dos graus de liberdade tal como exemplificado na .

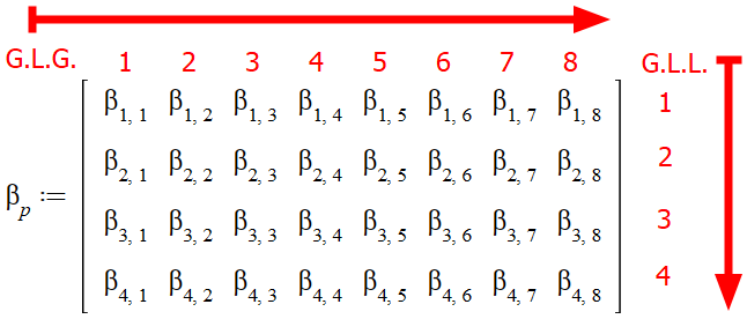



Figura 8: Correspondência de linhas e colunas da matriz $[\beta_p]$ com graus de liberdade local do elemento e global da estrutura, respectivamente.

A determinação da matriz de rigidez global do elemento $p$ ($k_{G,p}$) é dada pelo triplo produto matricial

$$[k_{G,p}] = [\beta_p]^T \cdot [k_{L,p}] \cdot [\beta_p].$$

Por fim, a Matriz Global da Estrutura é obtida somando as contribuições de rigidez de elemento, sendo expressa na equação:

$$[K_G] = \sum_{i=1}^{nEl} ([k_{G,i}]) = \sum_{i=1}^{nEl} ([\beta_i]^T \cdot [k_{L,i}] \cdot [\beta_i]).$$

## 6.2 Determinação dos deslocamentos nodais

Tendo em vista que a matriz de rigidez da estrutura $K_G$ é singular, as condições de contorno devem ser aplicadas para que se resolva o clássico problema estrutural

$$[K_G]\{u_G\} = \{F_N\},$$

no qual do vetor de forças nodais $F_N$. Aplicando as condições de contorno, um método de resolução de sistema de equações lineares deve ser utilizado para obtenção do vetor deslocamento global $u$ que representa os deslocamentos representados pelos graus de liberdade da estrutura.
Entretanto, deve ser utilizado o método do *Penalty* anteriormente a este último procedimento para considerar as restrições que os apoios dão aos graus de liberdade vinculados a eles. O método consiste em:

- Se o apoio é fixo no "GL" $n$: zerar a linha e coluna $n$ e tornar o elemento da $n,n$ da matriz de rigidez da estrutura $K_G$ unitário, além de zerar a linha $n$ do vetor de forças nodais $F_N$;
- Se o apoio é prescrito no "GL" $n$: zerar a linha $n$ e tornar o elemento da $n,n$ da matriz de rigidez da estrutura $K_G$ unitário, além de igualar a linha $n$ do vetor de forças nodais $F_N$ ao valor do deslocamento prescrito;
- Se o apoio é flexivel no "GL" $n$: soma-se ao elemento da $n,n$ da matriz de rigidez da estrutura $K_G$ a regidez da mola aplicada ao GL.

Seguindo estes passos, torna-se possível definir o vetor de deslocamentos globais da estrutura $\{u_G\}$.

## 6.3 Determinação dos esforços internos solicitantes e reações de apoio

Para obter os esforços internos solicitantes, basta obter o vetor de forças interno do elemento discreto $p$ ($\{F_{int,p}\}$) de interesse e avaliar o terceiro componente desse vetor (componente axial com convenção de acordo com a convenção padrão de tração e compressão). Por fim, o vetor de forças interno de um elemento $p$ é obtido pela equação

$$\{F_{int,p}\} = [K_{Int}] \cdot [T_p] \cdot [\beta_p] \cdot \{u_G\}.$$

Por sua vez, o vetor de reações de apoio pode ser obtido pela equação:

$$\{F_{react}\} = \sum_{i=1}^{nEl} \left( [\beta_i]^T \cdot [T_i]^T \cdot \{F_{int,i}\} \right) - \{F_N\},$$

Uma característica interessante do vetor $\{F_{react}\}$ é que cada coordenada representante de um GL sem condições de contorno é obrigatioriamente nula.

# 7 EXECUÇÃO DO PROGRAMA E SAÍDA DE DADOS

Ao executar a aplicação no cmd na pasta raiz da aplicação (*Truss2D_M*), utilizando o comando "python main.py", este executará a aplicação e solicitará a o diretório de arquivo de entrada. Ao especificar e apertar "Enter", o processamento será realizado e ao fim da análise estrutural será plotado o gráfico da Estrutura.

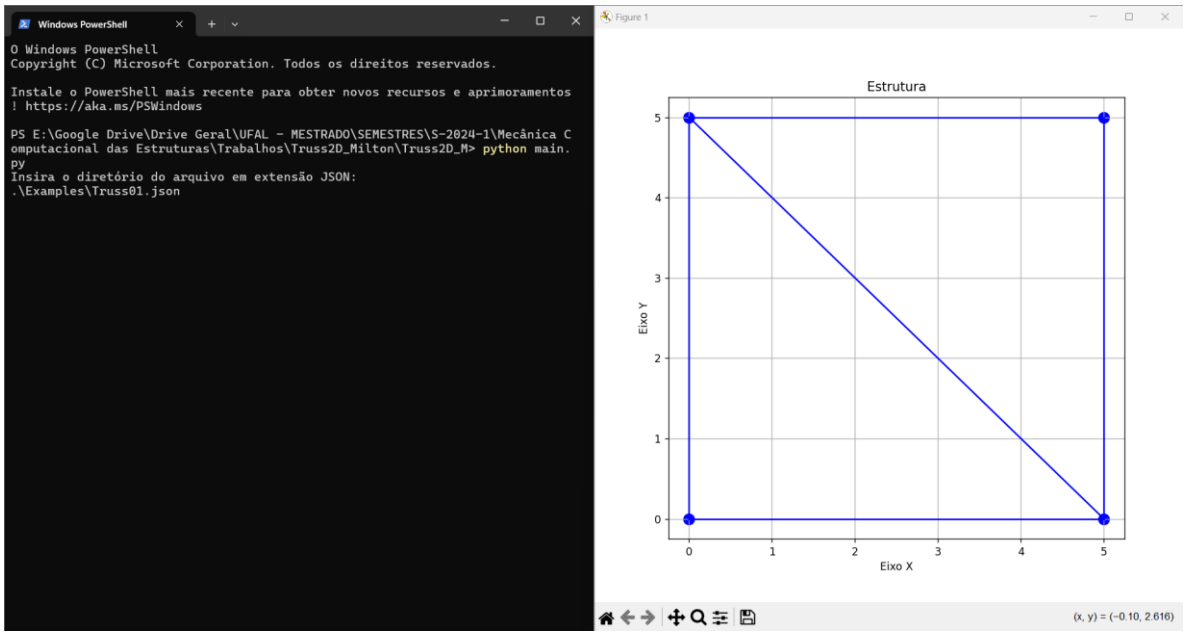



Figura 9: Execução da aplicação no cmd e plot da estrutura após a análise estrutural.

Além disso, uma pasta com o mesmo nome do arquivo é criada no diretório do arquivo de entrada. Exemplificando com o caso da Figura 9, a pasta com os resultados análise terá o nome “Truss01” e estará locada no diretório “.\Examples\”, contendo um plot da estrutura analisada e um arquivo JSON com os resultados, o qual recomenda-se fortemente a leitura via “https://jsonviewer.stack.hu/”.

# 8 VALIDAÇÃO DA APLICAÇÃO

Visando validar a aplicação, análise comparativa confrontando o *Ftool* é realizada. Definindo o modelo de comparação define a geometria, os carregamentos, as vinculações na Figura 9 e os valores das propriedades físicas e geométricas dos elementos e apoios como especificados na Tabela 1.

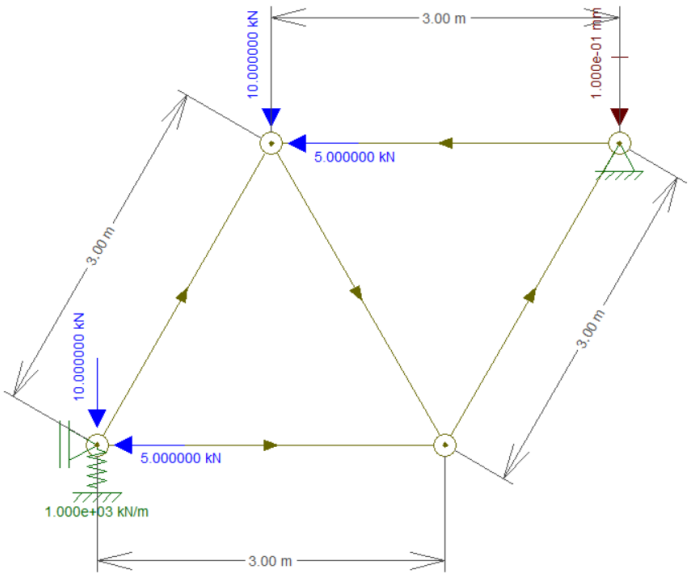



Figura 10: Problema proposto para validação de modelo (obtido no *Ftool*).

Tabela 1: Propriedades físicas e geométricas dos elementos e apoios.

| Parâmetro | Símbolo | Valor |
|---|---|---|
| Comprimento do elementos | $L$ | $3{,}00\ m$ |
| Módulo de Elasticidade de Young | $E$ | $205 \cdot 10^6\ kPa$ |
| Área da seção transversal | $A$ | $0{,}06\ m^2$ |
| Resistência da mola no apoio | $k$ | $1000\ kN/m$ |
| Deslocamento prescrito no apoio | $\delta v$ | $-0{,}1\ mm$ |

A seguir, apresenta-se os resultados obtidos pelo *software Ftool*

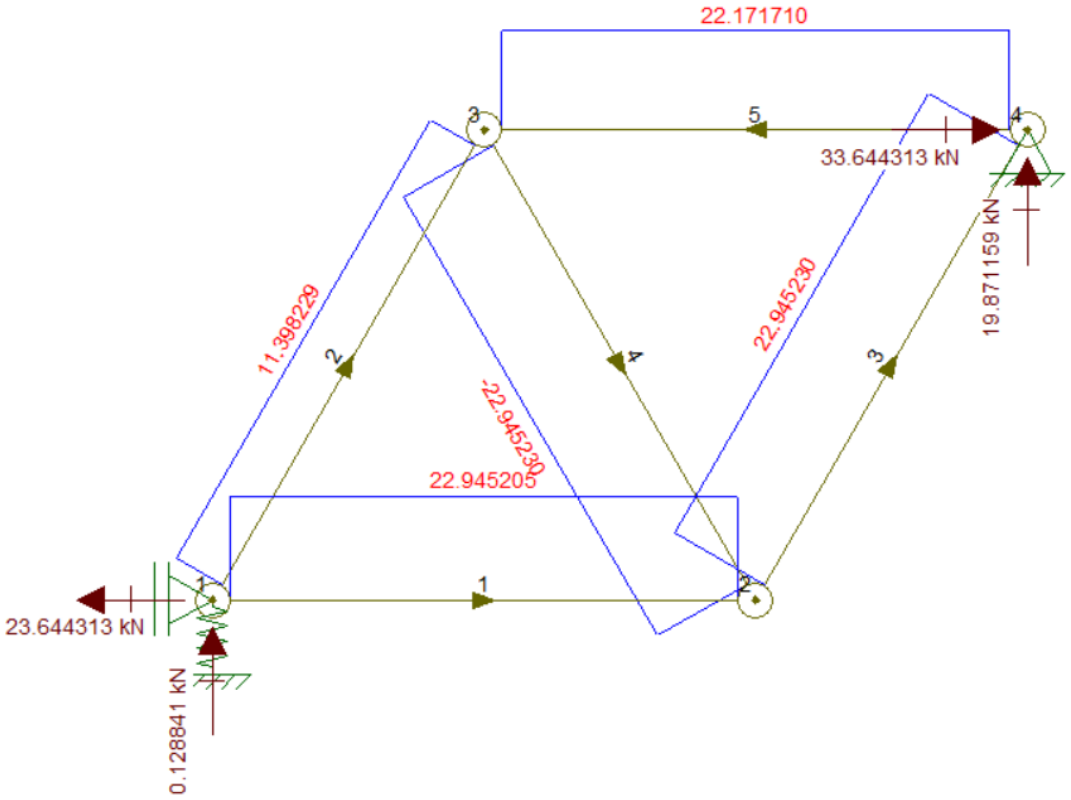



Figura 11: Resultados do proposto (obtido no *Ftool*).

Os resultados obtidos pela aplicação *Truss2D_M* e pelo *Ftool* estão presentes nas .

Tabela 2: Esforços normais nos elementos.

| Elemento | Esforço Normal [kN] | | Erro (%) |
|---|---|---|---|
| | ***Truss2D_M*** | ***Ftool*** | |
| 1 | 22,945238 | 22,945205 | $1{,}438 \cdot 10^{-4}$ |
| 2 | 11,398232 | 11,398229 | $2{,}632 \cdot 10^{-5}$ |
| 3 | 22,945238 | 22,945230 | $3{,}486 \cdot 10^{-5}$ |
| 4 | −22,945238 | −22,945230 | $3{,}486 \cdot 10^{-5}$ |
| 5 | 22,171735 | 22,171710 | $1{,}127 \cdot 10^{-4}$ |

Tabela 3: Deslocamentos nodais.

| **Nó** | **Direção** | **Deslocamentos (mm)** | | **Erro (%)** |
|---|---|---|---|---|
| | | ***Truss2D_M*** | ***Ftool*** | |
| 1 | y | $-1{,}2884 \cdot 10^{-1}$ | $-1{,}2884 \cdot 10^{-1}$ | 0,000 |
| 2 | x | $5{,}5963 \cdot 10^{-3}$ | $5{,}5964 \cdot 10^{-3}$ | $1{,}787 \cdot 10^{-3}$ |
| 2 | y | $-1{,}0969 \cdot 10^{-1}$ | $-1{,}0969 \cdot 10^{-1}$ | 0,000 |
| 3 | x | $-5{,}4077 \cdot 10^{-3}$ | $-5{,}4077 \cdot 10^{-3}$ | 0,000 |
| 3 | y | $-1{,}2251 \cdot 10^{-1}$ | $-1{,}2251 \cdot 10^{-1}$ | 0,000 |
| 4 | y | $-1{,}0000 \cdot 10^{-1}$ | $-1{,}0000 \cdot 10^{-1}$ | 0,000 |

Tabela 4: Reações de apoio.

| **Nó** | **Direção** | **Reações de apoio [kN]** | | **Erro (%)** |
|---|---|---|---|---|
| | | ***Truss2D_M*** | ***Ftool*** | |
| 1 | x | $-23{,}644354$ | $-23{,}644313$ | $1{,}734 \cdot 10^{-4}$ |
| 1 | y | 0,128841 | 0,128841 | 0,000 |
| 4 | x | 33,644354 | 33,644313 | $1{,}218 \cdot 10^{-4}$ |
| 4 | y | 19,871159 | 19,871159 | 0,000 |

Observando que os erros relativos, de forma geral, estão abaixo de $1{,}787 \cdot 10^{-5}$, é possível concluir que a aplicação está validada, possuindo resultados verossímios aos obtidos pelo *Ftool*.