UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS PROFESSORA MARIA DA PENHA
CENTRO DE CIENCIAS TECNOLOGIA E SAÚDE
CURSO DE ENGENHARIA CIVIL

SEBASTIÃO DE LIMA PIMENTA

ANÁLISE NÃO LINEAR GEOMÉTRICA DE FLECHAS EM TRELIÇAS METÁLICAS

ARARUNA - PB
2018

SEBASTIÃO DE LIMA PIMENTA

ANÁLISE NÃO LINEAR GEOMÉTRICA DE FLECHAS EM TRELIÇAS METÁLICAS

Trabalho de Conclusão de Curso apresentado ao Programa de Graduação em Engenharia Civil da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharel em Engenharia Civil.

Área de concentração: Estruturas.

Orientador: Prof. Me. Leonardo Medeiros da Costa.

ARARUNA - PB
2018

P644a Pimenta, Sebastião de Lima.
Análise não linear geométrica de flechas em treliças metálicas [manuscrito] : / Sebastiao de Lima Pimenta. - 2018.
69 p. : il. colorido.

Digitado.
Trabalho de Conclusão de Curso (Graduação em Engenharia Civil) - Universidade Estadual da Paraíba, Centro de Ciências, Tecnologia e Saúde , 2018.
"Orientação : Prof. Me. Leonardo Medeiros da Costa. , Coordenação do Curso de Engenharia Civil - CCTS."

1. Análise Estrutural. 2. Programa Computacional. 3. Engenharia estrutural.

21. ed. CDD 624.171

Elaborada por Tiago J . da S. Pereira - CRB - 15/450 BSC8/UEPB

SEBASTIÃO DE LIMA PIMENTA

ANÁLISE NÃO LINEAR GEOMÉTRICA DE FLECHAS EM TRELIÇAS METÁLICAS

Trabalho de Conclusão de Curso apresentado ao Programa de Graduação em Engenharia Civil da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharel em Engenharia Civil.

Área de concentração: Estruturas.

Aprovada em: 20 / 06 / 2018

BANCA EXAMINADORA

Prof. Me. Leonardo Medeiros da Costa (Orientador)
Universidade Estadual da Paraíba (UEPB)

Prof. Me. Eduardo Morais de Medeiros
Universidade Estadual da Paraíba (UEPB)

Doutorando Pedro Filipe de Luna Cunha
Universidade de Brasília (UNB)

A minha mãe, pelo apoio e o incentivo, DEDICO

AGRADECIMENTOS

Primeiramente à minha família que sempre me apoiou nos meus estudos.

Ao professor Leonardo Medeiros, pela orientação, por sempre estar à disposição para esclarecimento de duvidas e por ser um excelente profissional.

Ao professor Pedro Filipe, pela ajuda no entendimento do Método dos Elementos Finitos e por ser um ótimo professor.

Ao meu amigo Erveton Victor e ao professor Rafael de Brito, pelo incentivo à aprender programação.

Aos funcionários da UEPB, pela presteza e atendimento quando nos foi necessário.

Aos meus amigos Felipe Alves, Francisco Ramon e Roberto Paixão, componentes da BRAVOS, por sempre me incluírem nos grupos de trabalho e principalmente pela amizade.

Aos amigos que mudaram de universidade André Vieira e Inaldo Neto, que deixavam toda a turma mais alegre.

E por fim aos colegas de classe, por todos os momentos de descontração em forma de jogos, por todas as risadas e pelos melhores amigos que se pode encontrar.

RESUMO

A não linearidade geométrica consiste na consiste na determinação dos esforços e deslocamentos de uma estrutura levando em consideração o estado deformado da estrutura após aplicação de cargas, e portanto, as equações de equilíbrio devem ser reformuladas para cada uma dessas alterações. O presente trabalho, neste sentido, se propôs a elaborar um software capaz de analisar as flechas de uma estrutura do tipo treliça, implementando as mudanças de geometria a cada passo de carga aplicada. Para a realização dos cálculos foi utilizado o Método dos Elementos Finitos e o algoritmo foi escrito na linguagem de programação Python. Com os resultados obtidos no software, intitulado de SATE, e através do SAP2000 versão 15, constatou-se que a abordagem mais simplificada da não linearidade geométrica implementada no SATE, se aproxima bastante dos resultados do SAP2000 à medida que se diminui o incremento de carga. Os resultados mostram que o nó com maior divergência entre os softwares caiu de 5,61% para 1,89% quando se alterou o número de passos de carga, de dois para cem passos.

Palavras-Chave: Análise Não Linear. Análise Estrutural. Treliças Metálicas. Programa Computacional.

ABSTRACT

Geometric non-linearity consists of the analysis of a structure taking into account the deformed state of the structure after load application, and therefore, the equilibrium equations must be reformulated for each of these changes. The present work, in this sense, has proposed to elaborate a software capable of analyzing the arrows of a structure of the truss type, implementing the changes of geometry to each parcel of applied load. In order to perform the calculations, the Finite Element Method was used and the coding of the software was done using the Python programming language. With the results obtained in the software, titled SATE, and results obtained through SAP2000, it was found that the more simplified approach of geometric non-linearity implemented in SATE, closely approximates the results of SAP2000 as the number of applications increases of load. The results show that the node with the greatest divergence between the software dropped from 5.61% to 1.89% when the number of load applications was changed from two to one hundred parcels.

Keywords: Structural analysis. Metallic Trusses. Computational Program.

LISTA DE ILUSTRAÇÕES

Figura 3.1 - Treliça com dois graus de liberdade 14
Figura 3.2 - Equilíbrio do nó C 17
Figura 3.3 - Termos k11 e k21 da matriz de rigidez da treliça 19
Figura 3.4 - Termos 12 e k22 da matriz de rigidez da treliça 19
Figura 3.5 – (a) Forças no nó C para d1=1 e (b) Forças no nó C para d2=1 21
Figura 3.6 – Alongamento da barra tracionada 23
Figura 3.7 – Trajetória de equilíbrio 25
Figura 4.1 - Treliça Howe 28
Figura 4.2 - Fluxograma da Metodologia Utilizada 29
Figura 5.1- Interface gráfica do SATE 30
Figura 5.2 - Deformada da estrutura no SAP2000 para P = 3kN 31
Figura 5.3 - Deformada da estrutura no SAP2000 para P = 10kN 31
Figura 5.4 - Deformada da estrutura no SAP2000 para P = 100kN 31
Figura 5.5 - Modelagem a treliça Howe no SATE com carga de 3kN 32
Figura 5.6 - Modelagem a treliça Howe no SATE com carga de 10kN 32
Figura 5.7 - Modelagem a treliça Howe no SATE com carga de 100kN 32
Figura 5.8 - Deformada da estrutura pelo SATE para P = 3kN 33
Figura 5.9 - Deformada da estrutura pelo SATE para P = 10kN 33
Figura 5.10 - Deformada da estrutura pelo SATE para P = 100kN 33
Figura 5.11 - Deformada da estrutura pelo SATE com dois incrementos para P = 3kN 36
Figura 5.12 - Deformada da estrutura pelo SATE com dois incrementos para P = 10kN 37
Figura 5.13 - Deformada da estrutura pelo SATE com dois incrementos para P = 100kN 37
Figura 5.14 - Deformada da estrutura pelo SATE com dez incrementos para P = 3kN 38
Figura 5.15 - Deformada da estrutura pelo SATE com dez incrementos para P = 10kN 39
Figura 5.16 - Deformada da estrutura pelo SATE com dez incrementos para P = 100kN 39
Figura 5.17 - Deformada da estrutura pelo SATE com cem incrementos para P = 3kN 40
Figura 5.18 - Deformada da estrutura pelo SATE com cem incrementos para P = 10kN 41
Figura 5.19 - Deformada da estrutura pelo SATE com cem incrementos para P = 100kN 41

Figura 5.20 – Deformada da estrutura pela análise não linear do SAP2000 para P = 3kN.....42
Figura 5.21 - Deformada da estrutura pela análise não linear do SAP2000 para P = 10kN 43
Figura 5.22 - Deformada da estrutura pela análise não linear do SAP2000 para P = 100kN ..43

LISTA DE QUADROS

Quadro 5.1 - Diferença percentual entre as flechas dos softwares para P = 3kN.....34
Quadro 5.2 - Diferença percentual entre as flechas dos softwares para P = 10kN.....34
Quadro 5.3 - Diferença percentual entre as flechas dos softwares para P = 100kN.....35
Quadro 5.4 - Valores de flechas do SATE para dois incrementos 37
Quadro 5.5 - Valores de flechas do SATE para dez incrementos 39
Quadro 5.6 - Valores de flechas do SATE para cem incrementos 41
Quadro 5.7 - Valores de flechas não lineares do SAP2000.....43
Quadro 5.8 - Eficiência do SATE em relação ao SAP2000 para P = 3kN.....44
Quadro 5.9 - Eficiência do SATE em relação ao SAP2000 para P = 10kN.....45
Quadro 5.10 - Eficiência do SATE em relação ao SAP2000 para P = 100kN.....46
Quadro 5.11 - Comparação entre flechas lineares e não lineares do SATE para P = 3kN.....47
Quadro 5.12 - Comparação entre flechas lineares e não lineares do SATE para P = 10kN.....48
Quadro 5.13 - Comparação entre flechas lineares e não lineares do SATE para P = 100kN...48
Quadro 5.14 - Comparação entre flechas lineares e não lineares do SAP2000.....49

SUMÁRIO

1 INTRODUÇÃO 11
2 OBJETIVOS 12
2.1 OBJETIVO GERAL 12
2.2 OBJETIVOS ESPECÍFICOS 12
3 FUNDAMENTAÇÃO TEÓRICA 13
3.1 METODO DOS ELEMENTOS FINITOS 13
3.1.1 Método básico (Demonstração de Vaz) 14
3.1.2 Método clássico (Demonstração de Vaz) 18
3.1.3 Método da análise matricial 22
3.2 NÃO LINEARIDADE GEOMÉTRICA 24
4 METODOLOGIA 27
5 ANÁLISES E RESULTADOS 30
5.1 EFICIÊNCIA DA ANÁLISE LINEAR DO SATE 31
5.2 EFICIÊNCIA DA ANÁLISE NÃO LINEAR DO SATE 36
5.2.1 Aplicação de carga com dois incrementos 36
5.2.2 Aplicação de carga com dez incrementos 38
5.2.3 Aplicação de carga com cem incrementos 40
5.3 DIFERENÇA ENTRE FLECHAS LINEARES E NÃO LINEARES 47
6 CONCLUSÃO 51
REFERÊNCIAS 52
APÊNDICE A – CÓDIGO FONTE PRINCIPAL (SATE.PY) 53
APÊNDICE B – CÓDIGO FONTE AUXILIAR (ELEM_FINITOS.PY) 68

1 INTRODUÇÃO

As treliças metálicas são largamente utilizadas como elementos de cobertura devido à elevada relação resistência/peso. Deste modo é possível viabilizar construções leves que vencem grandes vãos.

De acordo com Lacerda (2014), treliças podem ser definidas como um arranjo estável de barras delgadas interligadas, onde as conexões são articuladas e as barras são conectadas por pinos sem atrito de forma que nenhum momento possa ser transmitido por essa conexão. Essa é a definição de uma treliça ideal, as estruturas reais fogem dessa premissa, porém os esforços oriundos da ligação com mais de um pino e/ou com atrito são pequenos, e não determinantes no dimensionamento. Desta forma, as treliças caracterizam-se por serem um arranjo de barras que somente transmitem forças axiais, contudo as treliças ainda podem estar sujeitas a esforços cortantes e momentos fletores se estiverem submetidas à forças transversais ao longo do elemento.

Dentro desse contexto a análise das flechas em treliças se faz necessária para garantir sua estabilidade e atender requisitos normativos quanto ao Estado Limite de Serviço, porém uma análise linear que é a comumente feita em projetos estruturais, pode não ser satisfatória uma vez que pode ocorrer a não linearidade física e/ou geométrica.

Como Cunha (2017) fala em seu trabalho devido a influência da rigidez do elemento estrutural no seu funcionamento a análise não linear tem maior relevância em estruturas esbeltas, estas que estão sendo utilizadas com maior frequência, tanto por fatores econômicos, quanto por arquitetônicos. Dessa forma, demanda-se maiores conhecimentos sobre a não linearidade física e geométrica por parte do projetista.

Os efeitos da não linearidade geométrica podem ser bastante nocivos principalmente para edifícios de múltiplos pavimentos, onde as ações horizontais são mais intensas, como a ação do vento, provocando maiores mudanças na geometria por meio dos deslocamentos, interferindo significativamente no equilíbrio mecânico. (SILVA, 2014)

A principal consequência de uma análise não linear é o aumento da complexidade do problema, fazendo com seja inviável realização de uma solução analítica, nesse contexto, o uso de ferramentas computacionais e aproximações numéricas torna-se praticamente obrigatório

2 OBJETIVOS

2.1 OBJETIVO GERAL

O presente trabalho tem como objetivo desenvolver um software capaz de calcular flechas de treliças levando em consideração a não linearidade geométrica. O desenvolvimento da programação realiza-se com a linguagem Python com base no método dos elementos finitos, formulado a partir do método dos deslocamentos.

2.2 OBJETIVOS ESPECÍFICOS

- Verificar se o Método dos Elementos Finitos MEF foi bem aplicado, através de comparação com a análise linear realizada pelo SAP2000;
- Verificar se a abordagem simplificada da não linearidade geométrica foi bem aplicada, através de comparação com a análise não linear geométrica realizada pelo SAP2000;
- Comparar os valores obtidos na análise linear com os valores obtidos na análise não linear.

3 FUNDAMENTAÇÃO TEÓRICA

Para a melhor compreensão deste trabalho se faz necessário a revisão bibliográfica do método dos elementos finitos, assim como da analise não linear geométrica.

3.1 METODO DOS ELEMENTOS FINITOS

De acordo com Vaz (2011) o MEF, diferente da Teoria da Elasticidade, faz uso dos conceitos de "discretização" do continuo e de "Matriz de interpolação" que fornece os deslocamentos em um ponto no interior do elemento em função de seus deslocamentos nodais, esses deslocamentos nos nós do modelo representam as incógnitas do método, além disso, o termo discretização utilizado refere-se a um modelo com um número finito de incógnitas para a análise de meios contínuos. Já a Teoria da Elasticidade realiza suas análises com um número infinito de variáveis que usam funções contínuas, ou seja, com infinitas incógnitas como solução.

O MEF utilizado neste trabalho, como dito anteriormente, é pertencente à família do Método dos Deslocamentos, os membros dessa família são caracterizados por possuírem, como equação fundamental, a equação de equilíbrio cujas incógnitas são deslocamentos generalizados. O conceito de generalizado aqui mencionado refere-se a deslocamentos generalizados, grandezas cinemáticas, tais como, deslocamentos lineares, rotações etc.

Vaz (2011) diz ainda que os membros dessa família do MEF passaram por mudanças ao longo do tempo formando uma árvore genealógica, com novos métodos herdando características dos métodos mais antigos e sendo gerados a partir deles. Os novos membros sofrem algumas alterações que só são bem sucedidas se forem bem adaptadas às condições existentes. Um exemplo que Vaz (2011) cita é a Análise Matricial de Estruturas (AME) e o MEF, que só tiveram larga aceitação quando os computadores atingiram grau de desenvolvimento suficiente para realizar as operações de forma viável, mesmo o MEF surgido antes dessa fase.

A análise matricial é um dos métodos que são aplicados na análise computacional de flechas em estruturas. Em seu livro Vaz (2011) demostra os métodos antecessores à análise matricial, esses métodos são o método básico e o método clássico. Antes de entrar no contexto

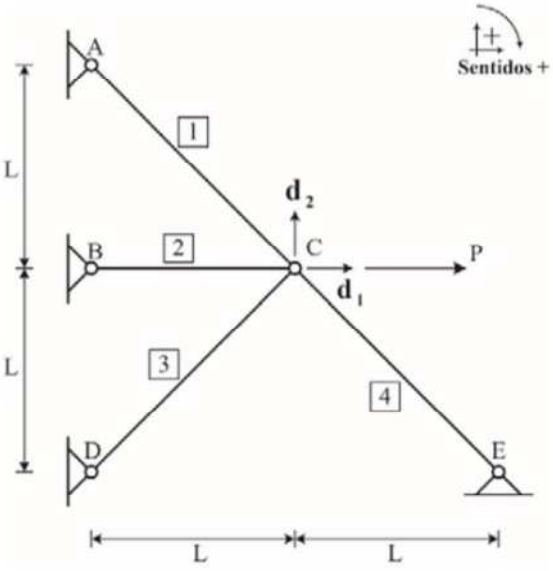
do método da análise matricial é importante explanar um pouco sobre esses dois métodos antecessores, tal explanação será feita através da demonstração realizada por Vaz (2011).

3.1.1 Método básico (Demonstração de Vaz)

O método básico mostra bem o conceito por trás do método dos elementos finitos, ele faz parte da família do método dos deslocamentos e seu processo consiste em manipular as três equações básicas da análise de estruturas (compatibilidade, de equilíbrio e constitutivas) de modo a colocar todas as informações disponíveis nas equações de equilíbrio com deslocamentos livres como incógnitas. O número de deslocamentos livres é chamado de grau de liberdade da estrutura.

Para melhor demonstrar o método básico faz-se uso da treliça na figura abaixo:

Figura 3.1 - Treliça com dois graus de liberdade



Fonte: Vaz (2011)

As equações de compatibilidade são aquelas que relacionam os deslocamentos nodais livres d1 e d2 na direção horizontal e vertical com alongamentos/encurtamentos δi das barras i. Os deslocamentos são supostos positivos com os sentidos indicados na Figura 3.1. Os alongamentos serão considerados positivos e os encurtamentos negativos. As expressões para

os δi das quatro barras são obtidas projetando-se os deslocamentos nodais nas direções das barras, assim:

$$\begin{cases} \delta_1(d_1,d_2) = d_1\frac{\sqrt{2}}{2} - d_2\frac{\sqrt{2}}{2} \\ \delta_2(d_1,d_2) = d_1 \\ \delta_3(d_1,d_2) = d_1\frac{\sqrt{2}}{2} + d_2\frac{\sqrt{2}}{2} \\ \delta_4(d_1,d_2) = -d_1\frac{\sqrt{2}}{2} + d_2\frac{\sqrt{2}}{2} \end{cases} \tag{3.1}$$

A segunda equação de compatibilidade relaciona os alongamentos/encurtamentos das barras δi com as deformações longitudinais ε_i. Da resistência dos materiais:

$$\varepsilon_i = \frac{\delta_i}{L_i} \tag{3.2}$$

Como os comprimentos das barras são:

$$\begin{cases} L_1 = L\sqrt{2} \\ L_2 = L \\ L_3 = L\sqrt{2} \\ L_4 = L\sqrt{2} \end{cases} \tag{3.3}$$

Chega-se a:

$$\begin{cases} \varepsilon_1(d_1,d_2) = \dfrac{d_1\frac{\sqrt{2}}{2} - d_2\frac{\sqrt{2}}{2}}{L\sqrt{2}} = \dfrac{1}{2L}(d_1 - d_2) \\ \varepsilon_2(d_1,d_2) = \dfrac{d_1}{L} \\ \varepsilon_3(d_1,d_2) = \dfrac{d_1\frac{\sqrt{2}}{2} + d_2\frac{\sqrt{2}}{2}}{L\sqrt{2}} = \dfrac{1}{2L}(d_1 + d_2) \\ \varepsilon_4(d_1,d_2) = \dfrac{-d_1\frac{\sqrt{2}}{2} + d_2\frac{\sqrt{2}}{2}}{L\sqrt{2}} = \dfrac{1}{2L}(-d_1 + d_2) \end{cases} \tag{3.4}$$

Para efeito de simplificação, se utilizará a lei de Hooke, uma vez que ainda estamos tratando apenas do cálculo de uma flecha convencional. Assim, para cada barra, temos:

$$\sigma_i = E\ \varepsilon_i \tag{3.5}$$

Ou, em termos de esforços normais Ni,

$$\frac{N_i}{A} = E\frac{\delta_i}{L_i} \tag{3.6}$$

Onde:

- E = módulo de elasticidade do material;
- A = área de seção transversal;
- Ni = esforço normal;
- Li = comprimento da barra i.

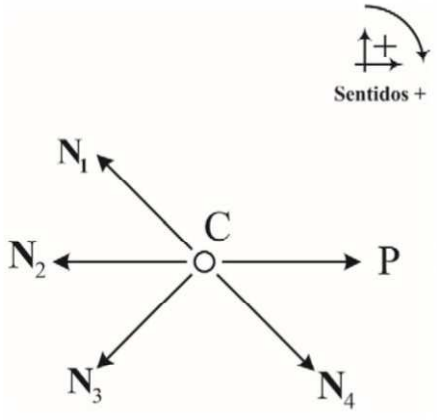
É importante observar que tanto o módulo de elasticidade do material quanto a área da seção transversal são duas grandezas supostas constantes para todas as barras.

Substituindo-se para cada barra i, δi dado em (3.1) na equação (3.6), obtém-se:

$$\begin{cases} N_1(d_1, d_2) = \dfrac{EA}{2L}(d_1 - d_2) \\ N_2(d_1, d_2) = \dfrac{EAd_1}{L} \\ N_3(d_1, d_2) = \dfrac{EA}{2L}(d_1 + d_2) \\ N_4(d_1, d_2) = \dfrac{EA}{2L}(-d_1 + d_2) \end{cases} \tag{3.7}$$

As equações de equilíbrio são obtidas para as direções horizontal e vertical no nó C. Os sentidos das forças axiais Ni que atuam nas barras i, são admitidos a princípio como de tração. Para se escrever as equações de equilíbrio, valem, no entanto, os sentidos indicados na Figura 3.2.

Figura 3.2 - Equilíbrio do nó C



Fonte: Vaz (2011)

As equações de equilíbrio são:

- Na direção horizontal:

$$\sum F_h = 0; \qquad -N_1\frac{\sqrt{2}}{2} - N_2 - N_3\frac{\sqrt{2}}{2} + N_4\frac{\sqrt{2}}{2} + P = 0; \tag{3.8}$$

- Na direção vertical:

$$\sum F_v = 0; \qquad N_1\frac{\sqrt{2}}{2} - N_3\frac{\sqrt{2}}{2} - N_4\frac{\sqrt{2}}{2} = 0; \tag{3.9}$$

Substituindo-se as expressões (3.7) em (3.9) e manipulando-as, obtém-se:

$$\begin{cases} \dfrac{2{,}061\,EA}{L}d_1 - \dfrac{0{,}354\,EA}{L}d_2 = P \\ \dfrac{-0{,}354\,EA}{L}d_1 + \dfrac{1{,}061\,EA}{L}d_2 = 0 \end{cases} \tag{3.10}$$

A expressão (3.10) é a equação fundamental do método dos deslocamentos para a análise da treliça plana da Figura 3.1. Matricialmente, ela pode ser reescrita como:

$$\frac{EA}{L}\begin{bmatrix} 2{,}061 & -0{,}354 \\ -0{,}354 & 1{,}061 \end{bmatrix}\begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} = \begin{Bmatrix} P \\ 0 \end{Bmatrix} \tag{3.11}$$

Cuja solução é:

$$\begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} = \frac{P\,L}{E\,A} \begin{Bmatrix} 0{,}515 \\ 0{,}171 \end{Bmatrix} \tag{3.12}$$

Com os deslocamentos d1 e d2 é possível obter agora todas as respostas da estrutura em termos de alongamento/encurtamento, na expressão (3.1), deformações, em (3.4), tensões, em (3.5), e esforços normais Ni, em (3.7). Tais valores estão indicados a seguir:

$$\begin{Bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{Bmatrix} = \frac{P\,L}{E\,A} \begin{Bmatrix} +0{,}243 \\ +0{,}515 \\ +1{,}778 \\ -0{,}243 \end{Bmatrix} \tag{3.13}$$

$$\begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \end{Bmatrix} = \frac{P}{E\,A} \begin{Bmatrix} +0{,}172 \\ +0{,}515 \\ +0{,}343 \\ -0{,}172 \end{Bmatrix} \tag{3.14}$$

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \end{Bmatrix} = \frac{P}{A} \begin{Bmatrix} +0{,}172 \\ +0{,}515 \\ +0{,}343 \\ -0{,}172 \end{Bmatrix} \tag{3.15}$$

$$\begin{Bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{Bmatrix} = P \begin{Bmatrix} +0{,}172 \\ +0{,}515 \\ +0{,}343 \\ -0{,}172 \end{Bmatrix} \tag{3.16}$$

3.1.2 Método clássico (Demonstração de Vaz)

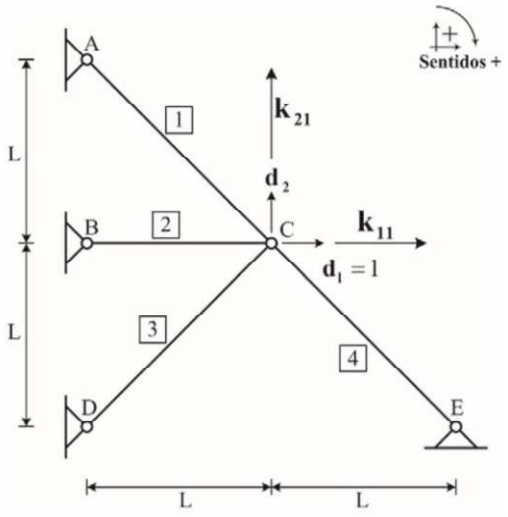
O método clássico é essencialmente o mesmo que o método básico. Sua contribuição foi no sentido de sistematizar, ou seja, organizar, ou ainda criar uma metodologia que possa ser aplicada da mesma forma a todas as estruturas.

O método utiliza conceitos de estados auxiliares e de superposição de efeitos. Onde se supõe um deslocamento de valor unitário na direção de cada um dos graus de liberdade. A força interna na direção i devido ao deslocamento unitário na direção do grau de liberdade dj é chamada de coeficiente de rigidez kij.

Aplicando o método para um melhor esclarecimento, tem-se.

- Estado auxiliar 1, d1 = 1.

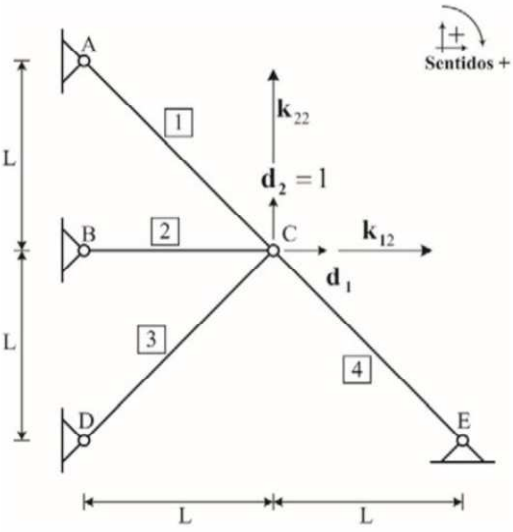
Figura 3.3 - Termos k11 e k21 da matriz de rigidez da treliça



Fonte: Vaz (2011)

- Estado auxiliar 2, d2 = 1.

Figura 3.4 - Termos 12 e k22 da matriz de rigidez da treliça



Fonte: Vaz (2011)

Para se obter os coeficientes kij (força interna resultante na direção i devida a um deslocamento unitário na direção j) procede-se da seguinte maneira: inicialmente, calculam-se os alongamentos/encurtamentos das barras dij (alongamento/encurtamento na barra i devido a

um deslocamento unitário na direção do grau de liberdade dj) de forma análoga ao que foi feito para se obter os alongamentos/encurtamentos no método básico.

- Para o estado auxiliar 1.

$$\begin{cases} \delta_{11} = \frac{\sqrt{2}}{2} \\ \delta_{21} = 1 \\ \delta_{31} = \frac{\sqrt{2}}{2} \\ \delta_{41} = \frac{-\sqrt{2}}{2} \end{cases} \tag{3.17}$$

- Para o estado auxiliar 2.

$$\begin{cases} \delta_{12} = \frac{-\sqrt{2}}{2} \\ \delta_{22} = 0 \\ \delta_{32} = \frac{\sqrt{2}}{2} \\ \delta_{42} = \frac{\sqrt{2}}{2} \end{cases} \tag{3.18}$$

Utilizando-se a relação constitutiva é possível calcular os esforços normais nas barras Nij (esforço normal na barra i devido a um deslocamento unitário na direção do grau de liberdade dj) com uma expressão análoga a (3.6).

$$N_{ij} = E\,A\frac{\delta_{ij}}{L_i} \tag{3.19}$$

Assim:

- Para o estado auxiliar 1.

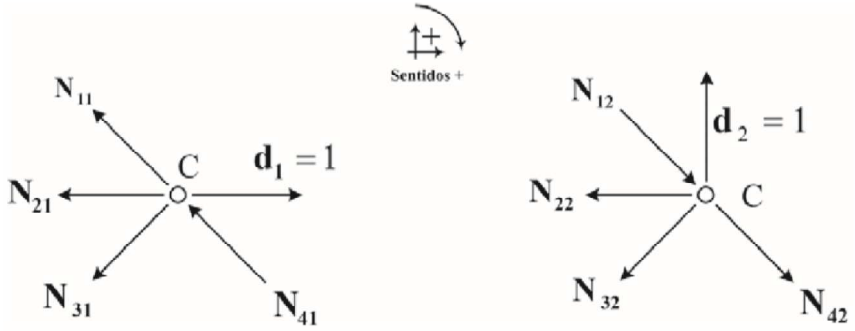
$$\begin{cases} N_{11} = \dfrac{\bar{E}\,\bar{A}}{2\,L} \\ N_{21} = \dfrac{E\,A}{L} \\ N_{31} = \dfrac{E\,A}{2\,L} \\ N_{41} = \dfrac{-E\,A}{2\,L} \end{cases} \tag{3.20}$$

- Para o estado auxiliar 2.

$$\begin{cases} N_{12} = \dfrac{-E\,A}{2\,L} \\ N_{22} = 0 \\ N_{32} = \dfrac{E\,A}{2\,L} \\ N_{42} = \dfrac{E\,A}{2\,L} \end{cases} \tag{3.21}$$

Os coeficientes de rigidez kij (esforço na direção i para um deslocamento unitário na direção j) são calculados utilizando-se as equações de equilíbrio no nó C. Assim, das equações de equilíbrio na direção horizontal e vertical da Figura 3.5, da correspondente a d1 = 1 obtém-se, respectivamente, os coeficientes k11 e k21.

Figura 3.5 – (a) Forças no nó C para d1=1 e (b) Forças no nó C para d2=1



Fonte: Vaz (2011)

- Para o estado auxiliar 1, Figura 3.5a.

$$\begin{cases} k_{11} = 2{,}061\dfrac{\bar{E}\,\bar{A}}{L} \\ k_{21} = -0{,}354\dfrac{E\,A}{L} \end{cases} \tag{3.22}$$

- Para o estado auxiliar 2, Figura 3.5b.

$$\begin{cases} k_{12} = -0{,}354\dfrac{E\,A}{L} \\ k_{22} = 1{,}061\dfrac{E\,A}{L} \end{cases} \tag{3.23}$$

- O estado auxiliar 0, fornece:

$$\begin{cases} f_1 = P \\ f_2 = 0 \end{cases} \tag{3.24}$$

A superposição de efeitos, que deve garantir o equilíbrio das forças resistentes e aplicadas, pode agora ser escrita como:

$$\begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} \tag{3.25}$$

ou com os valores da estrutura sendo analisada:

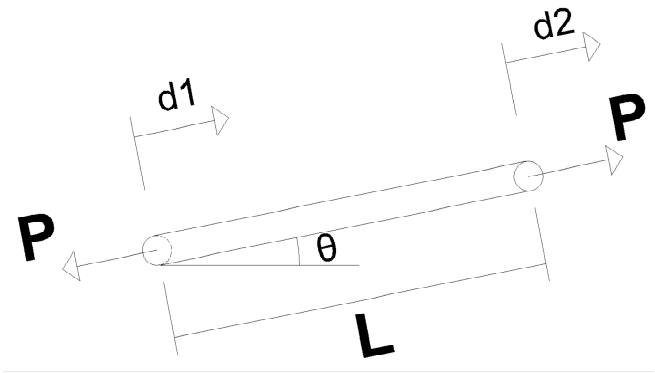
$$\frac{E\,A}{L}\begin{bmatrix} 2{,}061 & -0354 \\ -0{,}354 & 1{,}061 \end{bmatrix}\begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} = \begin{Bmatrix} P \\ 0 \end{Bmatrix}; \qquad \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} = \frac{P\,L}{E\,A}\begin{Bmatrix} 0{,}515 \\ 0{,}171 \end{Bmatrix} \tag{3.26}$$

A expressão (3.26) é idêntica à expressão (3.11), como não poderia deixar de ser. Desse modo, as respostas das estruturas obtidas pelo método básico, dadas pelas expressões de (3.12) a (3.16) serão as mesmas.

3.1.3 Método da análise matricial

Através da análise matricial foi possível sistematizar as operações matemáticas da análise de estruturas fazendo uso operações com vetores e matrizes. Toda essa sistematização se baseia na ideia de sistema local e sistema global de coordenadas.

Figura 3.6 – Alongamento da barra tracionada



Fonte: Próprio

Para a barra estar em equilíbrio, temos:

$$P_1 = -A * \sigma \tag{3.27}$$

$$P_2 = A * \sigma \tag{3.28}$$

Utilizando (3.2) e (3.5) nas expressões (3.27) e (3.28), temos:

$$P_1 = -EA * \frac{d_2 - d_1}{L} \tag{3.29}$$

$$P_2 = E\,A * \frac{d_2 - d_1}{L} \tag{3.30}$$

Em forma matricial:

$$\frac{E\,A}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} = \begin{Bmatrix} P_1 \\ P_2 \end{Bmatrix} \tag{3.31}$$

Os deslocamentos e forças podem ser decompostos no sistema de coordenadas globais x e y utilizando as seguintes equações:

$$d = d_x \cos\theta + d_y \sin\theta \tag{3.32}$$

$$P_x = P\cos\theta \tag{3.33}$$

$$P_y = P\cos\theta \tag{3.34}$$

Na forma matricial tem-se:

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ 0 & 0 & \cos\theta & \sin\theta \end{bmatrix}\begin{Bmatrix} d_{x1} \\ d_{y1} \\ d_{x2} \\ d_{y2} \end{Bmatrix} = \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} \tag{3.35}$$

$$\begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & \cos\theta \\ 0 & \sin\theta \end{bmatrix}\begin{Bmatrix} P_1 \\ P_2 \end{Bmatrix} = \begin{Bmatrix} P_{x1} \\ P_{y1} \\ P_{x2} \\ P_{y2} \end{Bmatrix} \tag{3.36}$$

Substituindo (3.36) e (3.35) na equação (3.31), tem-se:

$$\begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & \cos\theta \\ 0 & \sin\theta \end{bmatrix} \frac{E A}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ 0 & 0 & \cos\theta & \sin\theta \end{bmatrix} \begin{Bmatrix} d_{x1} \\ d_{y1} \\ d_{x2} \\ d_{y2} \end{Bmatrix} = \begin{Bmatrix} P_{x1} \\ P_{y1} \\ P_{x2} \\ P_{y2} \end{Bmatrix} \quad (3.37)$$

$$\frac{E A}{L} \begin{bmatrix} \cos^2\theta & \cos\theta\sin\theta & -\cos^2\theta & -\cos\theta\sin\theta \\ \cos\theta\sin\theta & \sin^2\theta & -\cos\theta\sin\theta & -\sin^2\theta \\ -\cos^2\theta & -\cos\theta\sin\theta & \cos^2\theta & \cos\theta\sin\theta \\ -\cos\theta\sin\theta & -\sin^2\theta & \cos\theta\sin\theta & \sin^2\theta \end{bmatrix} \begin{Bmatrix} d_{x1} \\ d_{y1} \\ d_{x2} \\ d_{y2} \end{Bmatrix} = \begin{Bmatrix} P_{x1} \\ P_{y1} \\ P_{x2} \\ P_{y2} \end{Bmatrix} \quad (3.38)$$

A equação (3.38) é a que será implementada no sistema de cálculo de flechas através do método dos elementos finitos.

3.2 NÃO LINEARIDADE GEOMÉTRICA

Como Lacerda (2014) menciona em seu trabalho, a análise linear das estruturas supõe algumas condições utilizadas para simplificar o problema, essas condições também são chamadas de hipóteses simplificadoras. As principais hipóteses simplificadoras são:

- Os deslocamentos nodais são infinitesimalmente pequenos;
- O material é linearmente elástico;
- As condições de contorno não se modificam durante a aplicação das cargas.

Quando todas essas hipóteses são cumpridas diz-se que a estrutura tem comportamento linear, ou seja, existe uma proporcionalidade entre carga e deslocamento. Porém muitas vezes, antes mesmo do material perder sua linearidade física, os deslocamentos nodais aumentam até certo ponto onde não podem ser considerados tão insignificantes.

De acordo com Benjamin (1982), a não linearidade geométrica é relevante nos casos em que devido a grandeza dos deslocamentos surge a necessidade de se escreverem as equações de equilíbrio em relação a configuração deformada da estrutura.

A realização dessas análises não lineares podem ser mais importantes em determinadas ocasiões, como diz Pinto & Ramalho (2002), no projeto de edifícios altos deve-se atentar para o problema da não linearidade geométrica quando a estrutura é solicitada simultaneamente por carregamentos verticais e horizontais. Pois, o carregamento vertical agindo na estrutura deslocada pode ocasionar a manifestação de esforços adicionais capazes

de conduzi-la ao colapso, uma vez que a estrutura foi projetada levando em conta sua geometria inicial. Tais efeitos podem ser desprezados se a estrutura for rígida o suficiente, porém, em estruturas flexíveis, esses efeitos são bastante significativos devendo ser obrigatoriamente considerados. Desse modo, as estruturas podem ser classificadas em estruturas de nós móveis ou estruturas de nós fixos, conforme a importância dos efeitos de segunda ordem na análise.

O comportamento não linear é estudado muitas vezes fazendo uso da trajetória de equilíbrio ou caminho de equilíbrio, que consiste num gráfico que relaciona carga × deslocamento. Cada ponto ao longo dessa trajetória representa uma configuração de equilíbrio estático da estrutura, ou seja, ao ser solicitada por diferentes valores de carga, a estrutura obteve determinadas deformações e é deslocada antes de atingir o repouso, esses valores de carga e deslocamentos compõem a trajetória de equilíbrio. Se o gráfico resultante dessa relação for não linear, significa que a estrutura também tem comportamento não linear.

Figura 3.7 – Trajetória de equilíbrio

Fonte: Lacerda (2014)

Essa trajetória de equilíbrio descrita possui alguns pontos críticos que são interessantes de serem analisados. “Esses pontos são aqueles em que uma trajetória de equilíbrio atinge valores extremos ou aqueles em que diferentes caminhos de equilíbrio se encontram” (Souza 2015). De acordo com Lacerda (2014) existem dois tipos de pontos críticos, um desses tipos é o ponto de limite, que é um ponto de extremo (ponto de máximo ou mínimo) no gráfico, o outro tipo é o ponto de bifurcação que é o ponto no qual dois ou mais caminhos de equilíbrio

se cruzam. Lacerda (2014) diz ainda que quando a estrutura atinge um dos pontos críticos, ela pode se torna instável, por isso, a identificação deles é de grande importância para um projeto de engenharia.

No presente trabalho não utiliza-se a trajetória de equilíbrio, o conceito de não linearidade geométrica usado aqui será o mais básico e característico, que foi mencionado varias vezes, a análise da estrutura já deformada, para isso se utilizará a força aplicada em passos.

4 METODOLOGIA

Para se cumprir os objetivos propostos para esse trabalho foi necessário a utilização de dois softwares, a linguagem de programação Python e a ferramenta de análise estrutural SAP2000 versão 15.

O Python foi utilizado para desenvolver toda interface gráfica e lógica operacional do programa proposto nos objetivos, dando atenção especial a construção de uma interface gráfica simples e intuitiva para melhor adaptabilidade do usuário a plataforma, além de utilizar os conceitos e fundamentos do método dos elementos finitos para realizar todos os cálculos necessários na obtenção das flechas de treliças planas.

O software proposto foi batizado de "SATE" (Software de Análise de Treliças por Etapas). Vale salientar que o SATE se limitar apenas ao cálculo de flechas de treliças com cargas nodais, dessa forma o mesmo não leva em consideração o peso próprio ou qualquer outra carga fora dos nós.

Após o desenvolvimento do programa computacional, foram realizados os seguintes testes comparativos:

- Flechas calculadas utilizando o software desenvolvido e flechas calculadas através de análise linear utilizando o SAP2000;
- Flechas calculadas utilizando a análise não linear geométrica através do software desenvolvido e flechas calculadas através de análise não linear geométrica (P-Delta) utilizando o SAP2000.
- Treliças analisadas de forma linear e treliças analisadas com não linearidade geométrica;

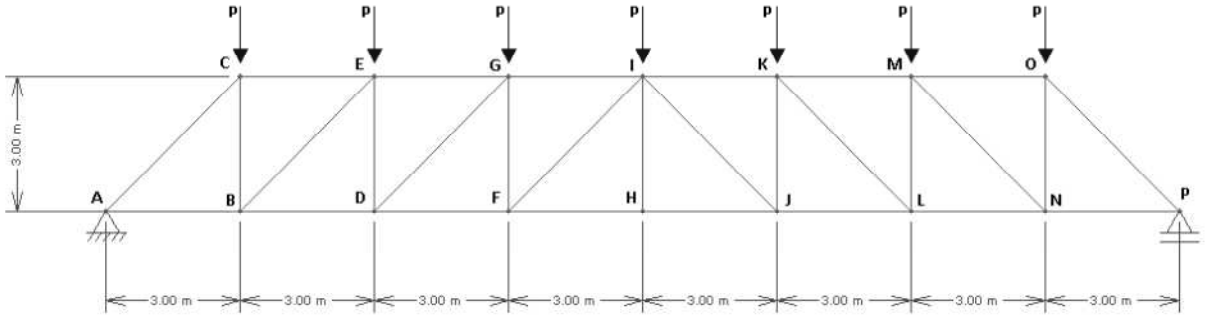
Foi necessária a adoção de uma treliça plana para servir como treliça base na realização das análises comparativas. As características e geometria da treliça adotada são:

- Treliça de aço com módulo de elasticidade igual a 200 GPa;
- Barras com seção de área igual a 1 cm^2 ou 0,0001 m^2;
- A geometria da treliça será a de uma treliça Howe.

Para cada uma das análises comparativas mencionadas foram utilizadas três forças P diferentes, uma de 3kN, uma de 10kN e outra de 100kN. A carga de 3kN foi escolhida pois a flecha causada por ela está dentro dos limites de L/350, estabelecidos na NBR 8800:2008,

essa carga representa um carregamento mais próximo de um carreamento real. Já as cargas de 10 e 100 kN foram escolhidas para analisar o comportamento teórico da treliça quando submetida à grandes carregamentos.

Figura 4.1 - Treliça Howe



Fonte: Próprio

Já definidas as características da treliça base, essa foi primeiramente analisada de forma linear no SAP2000 para obtenção de flechas, e submetida às rotinas de cálculos programadas no software desenvolvido, a fim de se realizar a comparação entre os dois softwares através do erro percentual entre os métodos.

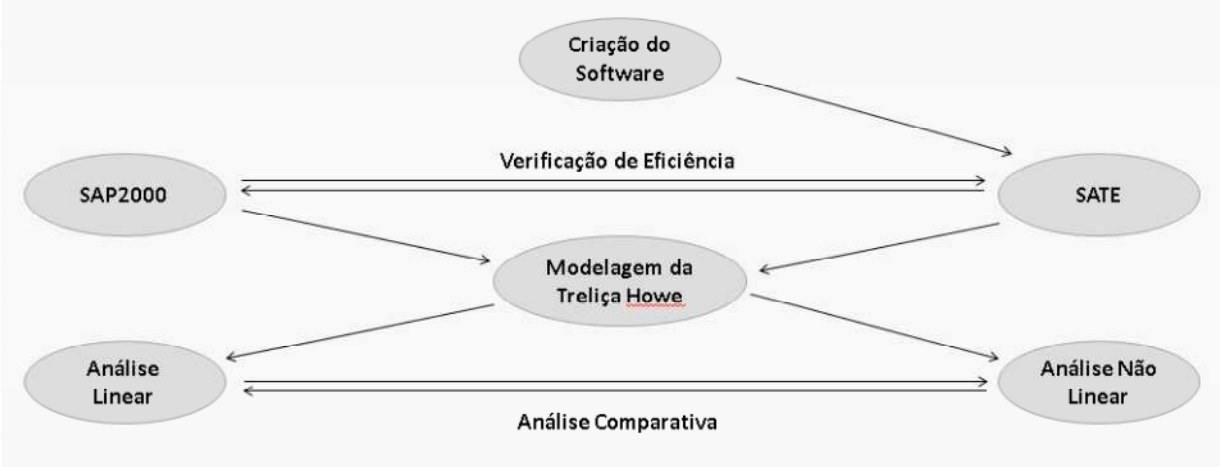
Para a realização da segunda análise comparativa primeiro foram expostos os valores obtidos no SATE, realizando a aplicação da força dividida em passos de 2, 10 e 100 vezes para cada uma das cargas. A variação no número de passos tem como finalidade visualizar o comportamento da estrutura com o aumento do número de aplicações. Em seguida foram exibidos os resultados obtidos com a análise não linear geométrica do SAP2000. Por fim foi feita a comparação entre os dois métodos para verificação da eficácia desse tipo análise.

A análise da treliça de forma não linear geométrica realizada no software proposto é realizada em incrementos de carga, ou seja, a força antes de ser aplicada será dividida em *n* passos, de forma que a cada aplicação de uma parcela da força a estrutura sofrerá alterações pelas deformações antes da aplicação da próxima parcela. Para que se possa definir o número de "etapas" em que a força será aplicada, foi colocado no programa desenvolvido um campo onde o usuário pode definir em quantas vezes a força será dividida.

A terceira análise comparativa é a análise entre as flechas obtidas de forma linear e as flechas obtidas de forma não linear geométrica, essa análise tem como objetivo verificar a o quanto a não linearidade geométrica pode influenciar nos valores das flechas. Nesta análise os

valores de cada software foram analisados de forma separada, ou seja, as flechas lineares do SAP2000 foram comparadas com as flechas não lineares também do SAP2000, da mesma forma para o software desenvolvido. O fluxograma abaixo resume os procedimentos da metodologia.

Figura 4.2 - Fluxograma da Metodologia Utilizada



Fonte: Próprio

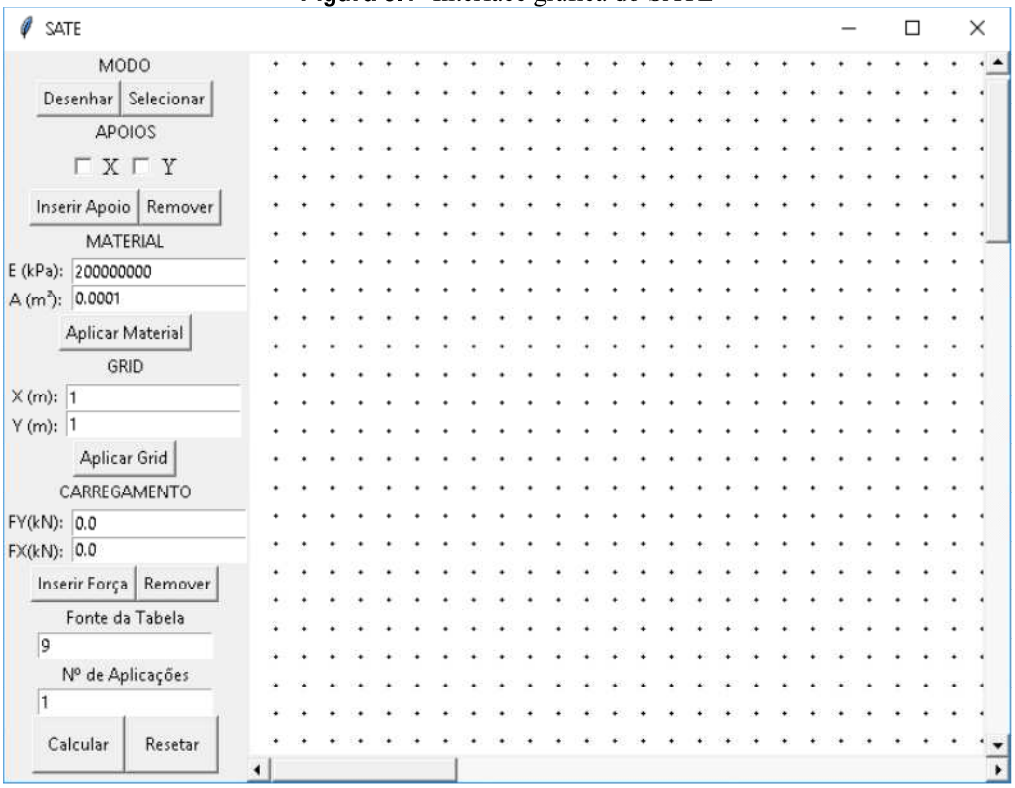
5 ANÁLISES E RESULTADOS

Seguindo os procedimentos mencionados na metodologia, o SATE foi desenvolvido através de dois códigos separados um deles, intitulado de “elem_finitos.py”, é responsável por todo o cálculo das flechas utilizando o método dos elementos finitos, já o outro código, intitulado de “SATE.py”, é a rotina principal responsável por toda a parte de interface gráfica, coleta dos dados, exibição dos resultados e integração com o algoritmo de cálculo já citado. Esses dois códigos são mostrados nos Apêndices A e B.

O SATE possui uma interface amigável e simplificada, com um grid que facilita o desenho da estrutura e o entendimento do usuário quanto à operação dessa ferramenta, além de ser esteticamente e funcionalmente bem similar a outros softwares desse tipo.

Após a modelagem da estrutura no SATE, o resultado é exibido em forma de gráfico e tabelas, onde para cada aplicação de incremento de força é traçado uma linha correspondente à deformação da estrutura. A figura abaixo mostra a interface do SATE.

Figura 5.1- Interface gráfica do SATE

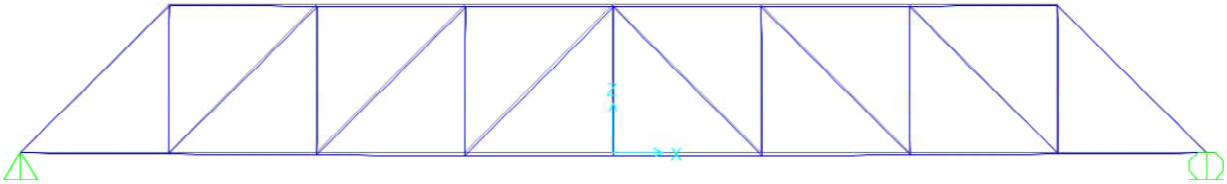


Fonte: Próprio

5.1 EFICIÊNCIA DA ANÁLISE LINEAR DO SATE

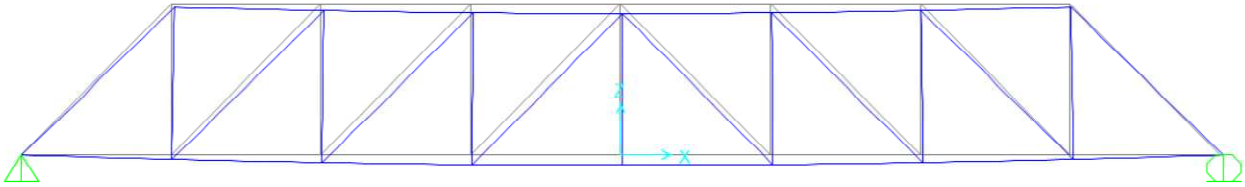
O início da análise de comparação da eficiência entre a ferramenta SAP2000 e o SATE deu-se com realização aa análise linear da treliça Howe no SAP2000 para as cargas de 3, 10 e 100 kN. Os gráficos das flechas resultantes para cada uma das cargas são exibidos nas figuras abaixo.

Figura 5.2 - Deformada da estrutura no SAP2000 para P = 3kN



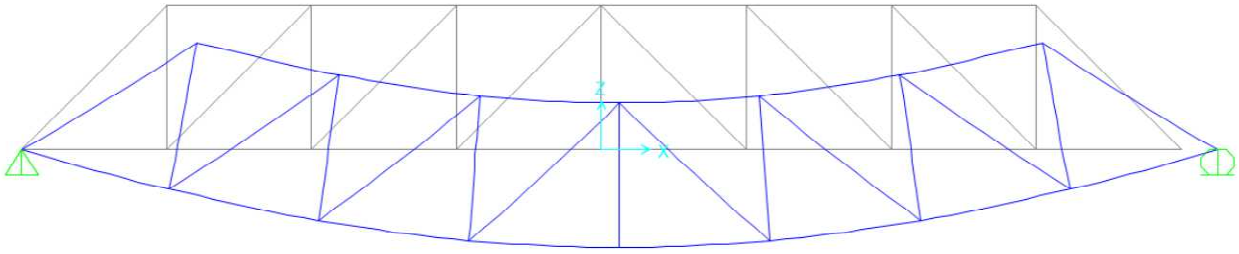
Fonte: SAP2000 V15

Figura 5.3 - Deformada da estrutura no SAP2000 para P = 10kN



Fonte: SAP2000 V15

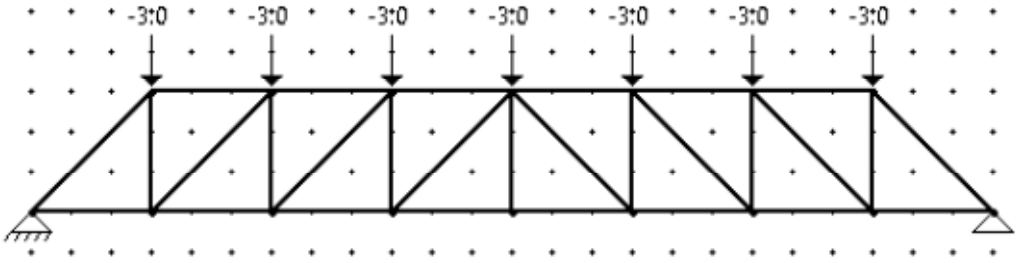
Figura 5.4 - Deformada da estrutura no SAP2000 para P = 100kN



Fonte: SAP2000 V15

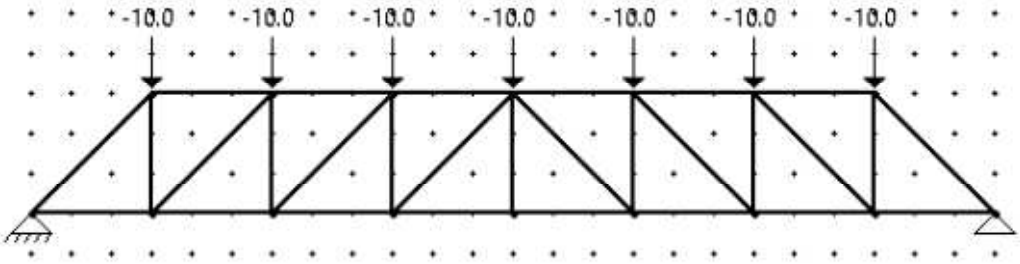
Finalizada a análise no SAP2000, foi feita a modelagem da treliça Howe no SATE, lembrando de que esta análise deve ser feita com a aplicação de toda carga de uma só vez, ou seja, sem dividir a força.

Figura 5.5 - Modelagem a treliça Howe no SATE com carga de 3kN



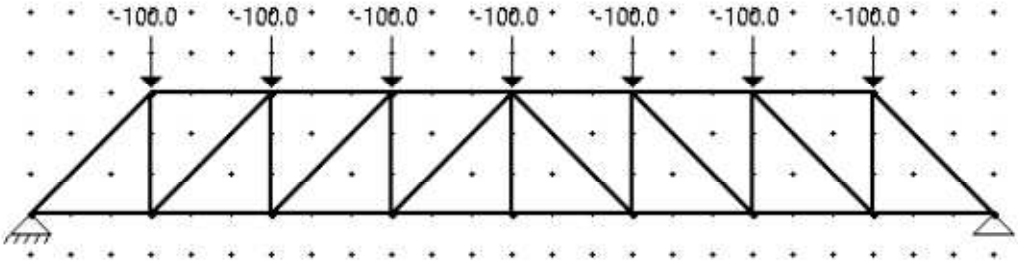
Fonte: Próprio

Figura 5.6 - Modelagem a treliça Howe no SATE com carga de 10kN



Fonte: Próprio

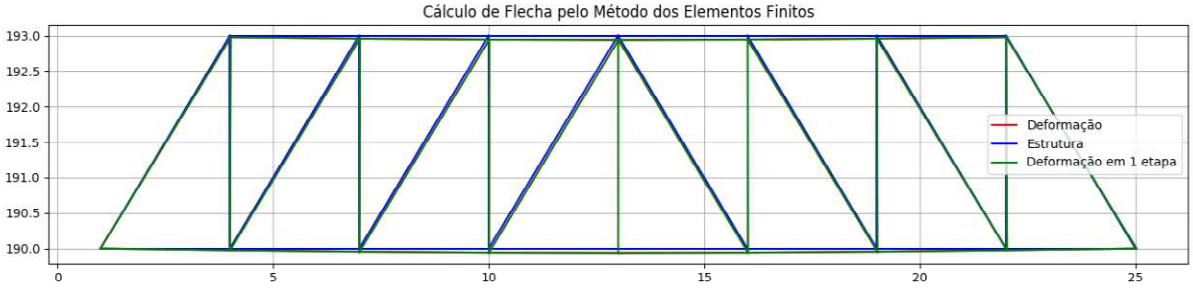
Figura 5.7 - Modelagem a treliça Howe no SATE com carga de 100kN



Fonte: Próprio

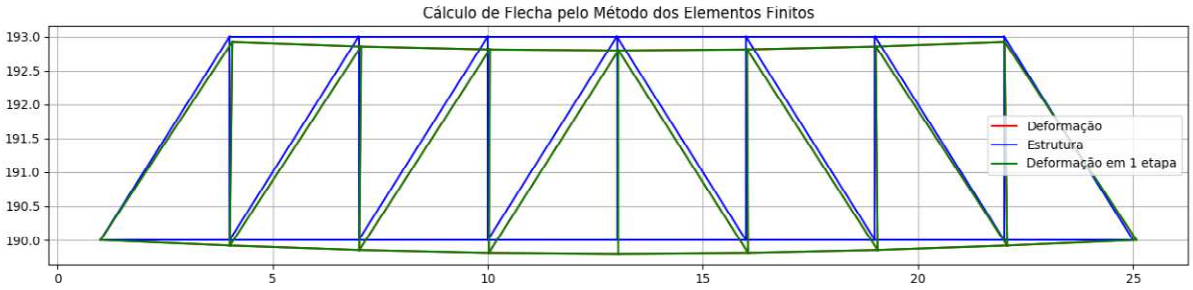
Modelada a treliça e definidas as características das barras, se fez a análise linear utilizando o SATE que resultou nas figuras a seguir.

Figura 5.8 - Deformada da estrutura pelo SATE para P = 3kN



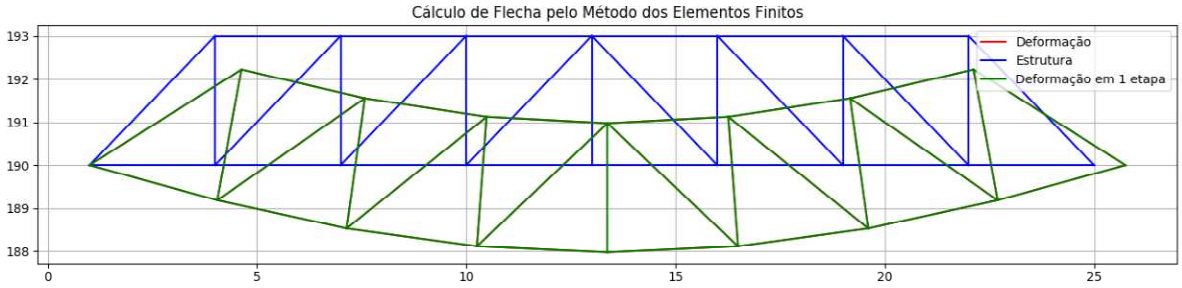
Fonte: Próprio

Figura 5.9 - Deformada da estrutura pelo SATE para P = 10kN



Fonte: Próprio

Figura 5.10 - Deformada da estrutura pelo SATE para P = 100kN



Fonte: Próprio

Os quadros a seguir exibem os valores de flechas de cada software e a diferença percentual entre os dois, para cada uma das cargas.

Quadro 5.1 - Diferença percentual entre as flechas dos softwares para P = 3kN

NÓS	FLECHA SAP2000 (m)		FLECHA SATE (m)		DIFERENÇA (%)	
	X	Y	X	Y	X	Y
Nó A	0,0000	0,0000	0,0000	0,0000	-	-
Nó B	0,0016	-0,0245	0,0016	-0,0245	0,00	0,00
Nó C	0,0189	-0,0234	0,0189	-0,0234	0,00	0,00
Nó D	0,0043	-0,0441	0,0043	-0,0441	0,00	0,00
Nó E	0,0173	-0,0434	0,0173	-0,0434	0,00	0,00
Nó F	0,0077	-0,0566	0,0076	-0,0566	-1,30	0,00
Nó G	0,0146	-0,0564	0,0146	-0,0563	0,00	-0,18
Nó H	0,0113	-0,0608	0,0113	-0,0608	0,00	0,00
Nó I	0,0113	-0,0608	0,0113	-0,0608	0,00	0,00
Nó J	0,0149	-0,0566	0,0148	-0,0566	-0,67	0,00
Nó K	0,0079	-0,0564	0,0079	-0,0563	0,00	-0,18
Nó L	0,0182	-0,0441	0,0182	-0,0441	0,00	0,00
Nó M	0,0052	-0,0434	0,0052	-0,0434	0,00	0,00
Nó N	0,0209	-0,0245	0,0209	-0,0245	0,00	0,00
Nó O	0,0036	-0,0234	0,0036	-0,0234	0,00	0,00
Nó P	0,0225	0,0000	0,0225	0,0000	0,00	-

Fonte: Próprio

Quadro 5.2 - Diferença percentual entre as flechas dos softwares para P = 10kN

NÓS	FLECHA SAP2000 (m)		FLECHA SATE (m)		DIFERENÇA (%)	
	X	Y	X	Y	X	Y
Nó A	0,0000	0,0000	0,0000	0,0000	-	-
Nó B	0,0053	-0,0816	0,0053	-0,0816	0,00	0,00
Nó C	0,0630	-0,0779	0,0630	-0,0778	0,00	-0,13
Nó D	0,0143	-0,1470	0,0142	-0,1470	-0,70	0,00
Nó E	0,0578	-0,1447	0,0578	-0,1447	0,00	0,00
Nó F	0,0255	-0,1886	0,0255	-0,1886	0,00	0,00
Nó G	0,0488	-0,1879	0,0488	-0,1878	0,00	-0,05

NÓS	FLECHA SAP2000 (m)		FLECHA SATE (m)		DIFERENÇA (%)	
	X	Y	X	Y	X	Y
Nó H	0,0375	-0,2027	0,0375	-0,2027	0,00	0,00
Nó I	0,0375	-0,2027	0,0375	-0,2027	0,00	0,00
Nó J	0,0495	-0,1886	0,0495	-0,1886	0,00	0,00
Nó K	0,0263	-0,1879	0,0263	-0,1878	0,00	-0,05
Nó L	0,0608	-0,1470	0,0607	-0,1470	-0,16	0,00
Nó M	0,0173	-0,1447	0,0173	-0,1447	0,00	0,00
Nó N	0,0689	-0,0816	0,0697	-0,0816	1,16	0,00
Nó O	0,0120	-0,0779	0,0120	-0,0778	0,00	-0,13
Nó P	0,0750	0,0000	0,0750	0,0000	0,00	-

Fonte: Próprio

Quadro 5.3 - Diferença percentual entre as flechas dos softwares para P = 100kN

NÓS	FLECHA SAP2000 (m)		FLECHA SATE (m)		DIFERENÇA (%)	
	X	Y	X	Y	X	Y
Nó A	0,0000	0,0000	0,0000	0,0000	-	-
Nó B	0,0525	-0,8162	0,0525	-0,8160	0,00	-0,02
Nó C	0,6302	-0,7787	0,6300	-0,7785	-0,03	-0,03
Nó D	0,1425	-1,4699	0,1425	-1,4696	0,00	-0,02
Nó E	0,5776	-1,4474	0,5775	-1,4471	-0,02	-0,02
Nó F	0,2551	-1,8862	0,2550	-1,8857	-0,04	-0,03
Nó G	0,4876	-1,8787	0,4875	-1,8782	-0,02	-0,03
Nó H	0,3751	-2,0274	0,3750	-2,0269	-0,03	-0,02
Nó I	0,3751	-2,0274	0,3750	-2,0269	-0,03	-0,02
Nó J	0,4951	-1,8862	0,4950	-1,8857	-0,02	-0,03
Nó K	0,2626	-1,8787	0,2625	-1,8782	-0,04	-0,03
Nó L	0,6077	-1,4699	0,6075	-1,4696	-0,03	-0,02
Nó M	0,1725	-1,4474	0,1725	-1,4471	0,00	-0,02
Nó N	0,6977	-0,8162	0,6975	-0,8160	-0,03	-0,02
Nó O	0,1200	-0,7787	0,1200	-0,7785	0,00	-0,03

NÓS	FLECHA SAP2000 (m)		FLECHA SATE (m)		DIFERENÇA (%)	
	X	Y	X	Y	X	Y
Nó P	0,7502	0,0000	0,7500	0,0000	-0,03	-

Fonte: Próprio

Com posse dos resultados de diferença percentual entre as flechas obtidas em cada software é possível ver que o SATE tem alguns valores com uma pequena diferença percentual em relação ao SAP2000, isso pode ser devido a erros de aproximação ou alguma diferença na aplicação do MEF, entretanto a diferença é muito pequena para ser considerada relevante.

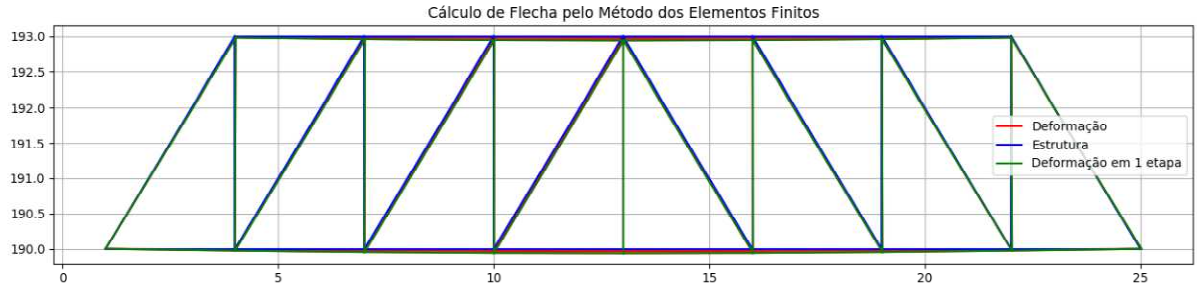
5.2 EFICIÊNCIA DA ANÁLISE NÃO LINEAR DO SATE

A segunda análise comparativa foi feita com a aplicação da carga dividida em 2, 10 e 100 vezes, para cada uma das cargas, dessa forma tem-se:

5.2.1 Aplicação de carga com dois incrementos

- Carga de 3 kN:

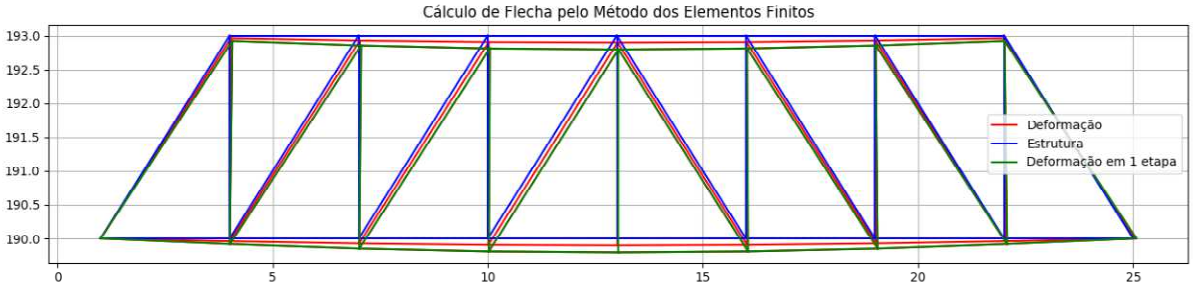
Figura 5.11 - Deformada da estrutura pelo SATE com dois incrementos para P = 3kN



Fonte: Próprio

- Carga de 10 kN:

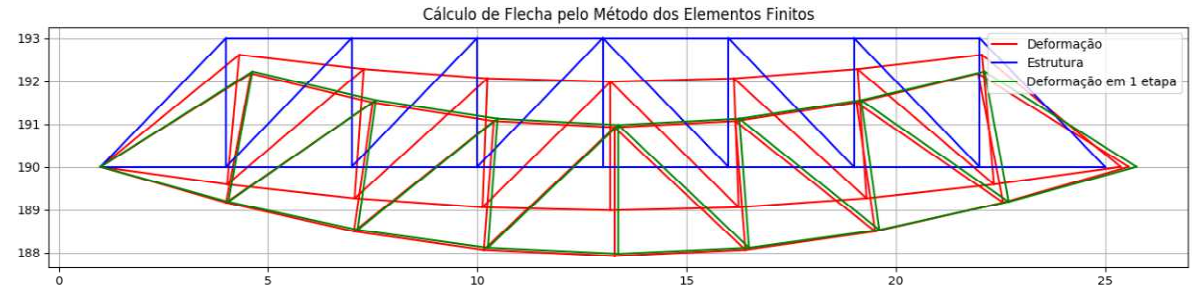
Figura 5.12 - Deformada da estrutura pelo SATE com dois incrementos para P = 10kN



Fonte: Próprio

- Carga de 100 kN:

Figura 5.13 - Deformada da estrutura pelo SATE com dois incrementos para P = 100kN



Fonte: Próprio

Quadro 5.4 - Valores de flechas do SATE para dois incrementos

NÓS	FLECHA PARA 3KN (m)		FLECHA PARA 10KN (m)		FLECHA PARA 100KN (m)	
	X	Y	X	Y	X	Y
Nó A	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
Nó B	0,0015	-0,0245	0,0047	-0,0818	-0,0011	-0,8338
Nó C	0,0189	-0,0234	0,0626	-0,0783	0,5944	-0,8259
Nó D	0,0042	-0,0441	0,0134	-0,1473	0,0565	-1,5053
Nó E	0,0173	-0,0435	0,0570	-0,1452	0,5029	-1,5008
Nó F	0,0076	-0,0566	0,0246	-0,1890	0,1589	-1,9343

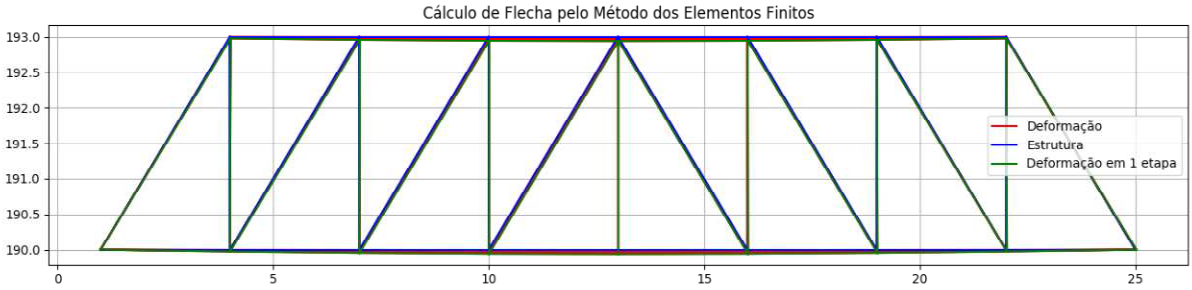
NÓS	FLECHA PARA 3KN (m)		FLECHA PARA 10KN (m)		FLECHA PARA 100KN (m)	
	X	Y	X	Y	X	Y
Nó G	0,0145	-0,0564	0,0479	-0,1883	0,3965	-1,9340
Nó H	0,0112	-0,0608	0,0366	-0,2032	0,2821	-2,0793
Nó I	0,0112	-0,0609	0,0366	-0,2032	0,2821	-2,0821
Nó J	0,0148	-0,0566	0,0486	-0,1890	0,4053	-1,9343
Nó K	0,0078	-0,0564	0,0253	-0,1883	0,1677	-1,9340
Nó L	0,0181	-0,0441	0,0598	-0,1473	0,5078	-1,5053
Nó M	0,0051	-0,0435	0,0162	-0,1452	0,0614	-1,5008
Nó N	0,0208	-0,0245	0,0685	-0,0818	0,5654	-0,8338
Nó O	0,0035	-0,0234	0,0105	-0,0783	-0,0301	-0,8259
Nó P	0,0223	0,0000	0,0732	0,0000	0,5642	0,0000

Fonte: Próprio

5.2.2 Aplicação de carga com dez incrementos

- Carga de 3 kN:

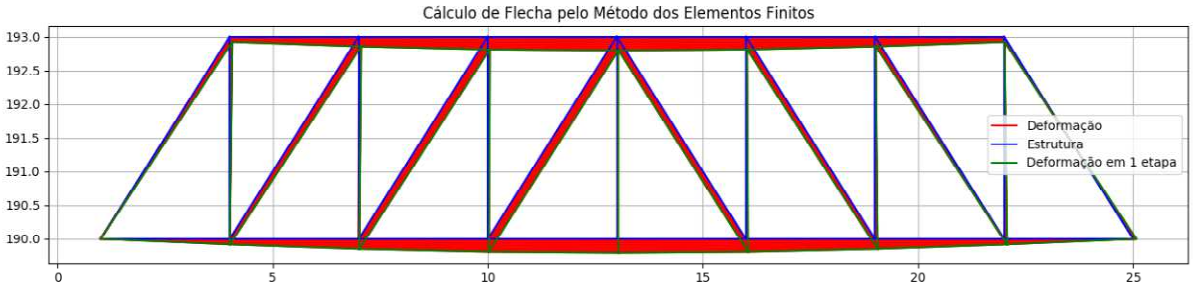
Figura 5.14 - Deformada da estrutura pelo SATE com dez incrementos para P = 3kN



Fonte: Próprio

- Carga de 10 kN:

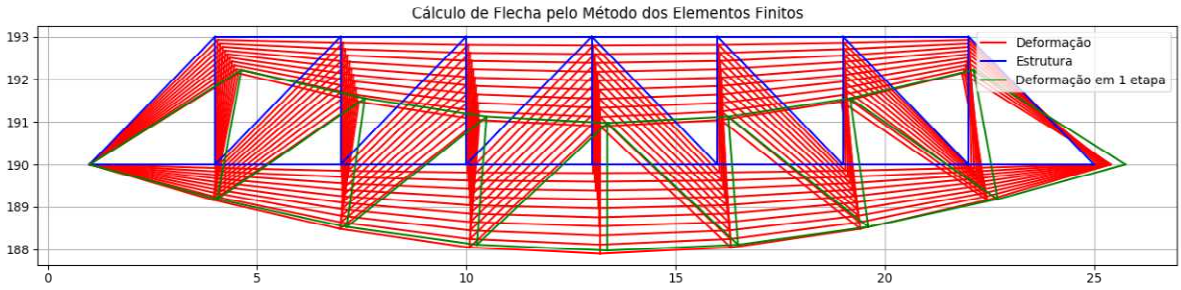
Figura 5.15 - Deformada da estrutura pelo SATE com dez incrementos para P = 10kN



Fonte: Próprio

- Carga de 100 kN:

Figura 5.16 - Deformada da estrutura pelo SATE com dez incrementos para P = 100kN



Fonte: Próprio

Quadro 5.5 - Valores de flechas do SATE para dez incrementos

NÓS	FLECHA PARA 3KN (m)		FLECHA PARA 10KN (m)		FLECHA PARA 100KN (m)	
	X	Y	X	Y	X	Y
Nó A	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
Nó B	0,0015	-0,0245	0,0043	-0,0819	-0,0464	-0,8357
Nó C	0,0188	-0,0234	0,0623	-0,0786	0,5573	-0,8537
Nó D	0,0041	-0,0441	0,0127	-0,1475	-0,0174	-1,5147
Nó E	0,0172	-0,0435	0,0564	-0,1456	0,4349	-1,5259

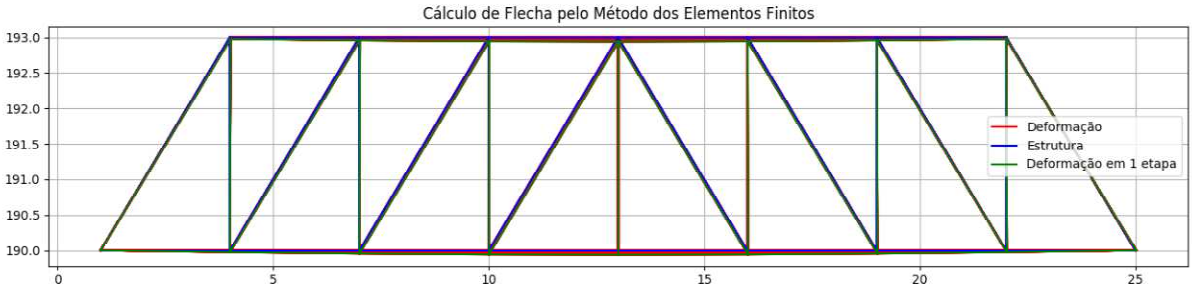
NÓS	FLECHA PARA 3KN (m)		FLECHA PARA 10KN (m)		FLECHA PARA 100KN (m)	
	X	Y	X	Y	X	Y
Nó F	0,0075	-0,0566	0,0238	-0,1893	0,0754	-1,9516
Nó G	0,0145	-0,0564	0,0471	-0,1887	0,3161	-1,9575
Nó H	0,0111	-0,0609	0,0358	-0,2035	0,2008	-2,0991
Nó I	0,0111	-0,0609	0,0358	-0,2036	0,2008	-2,1044
Nó J	0,0147	-0,0566	0,0479	-0,1893	0,3261	-1,9516
Nó K	0,0077	-0,0564	0,0246	-0,1887	0,0854	-1,9575
Nó L	0,0181	-0,0441	0,0590	-0,1475	0,4190	-1,5147
Nó M	0,0050	-0,0435	0,0153	-0,1456	-0,0333	-1,5259
Nó N	0,0207	-0,0245	0,0674	-0,0819	0,4480	-0,8357
Nó O	0,0034	-0,0234	0,0094	-0,0786	-0,1557	-0,8537
Nó P	0,0222	0,0000	0,0717	0,0000	0,4016	0,0000

Fonte: Próprio

5.2.3 Aplicação de carga com cem incrementos

- Carga de 3 kN:

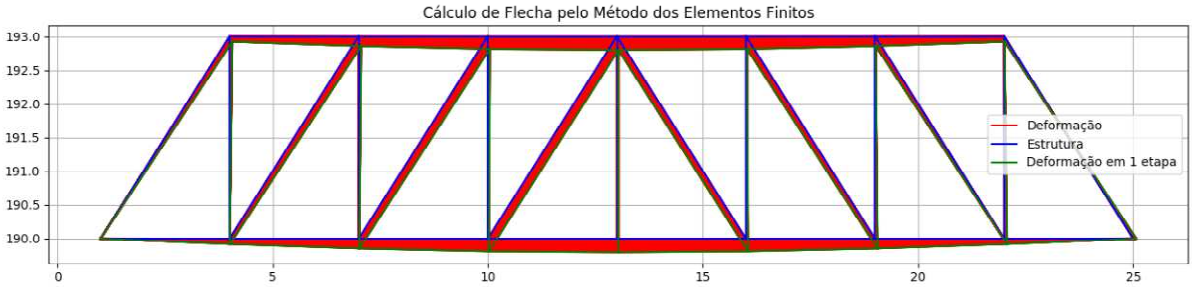
Figura 5.17 - Deformada da estrutura pelo SATE com cem incrementos para P = 3kN



Fonte: Próprio

- Carga de 10 kN:

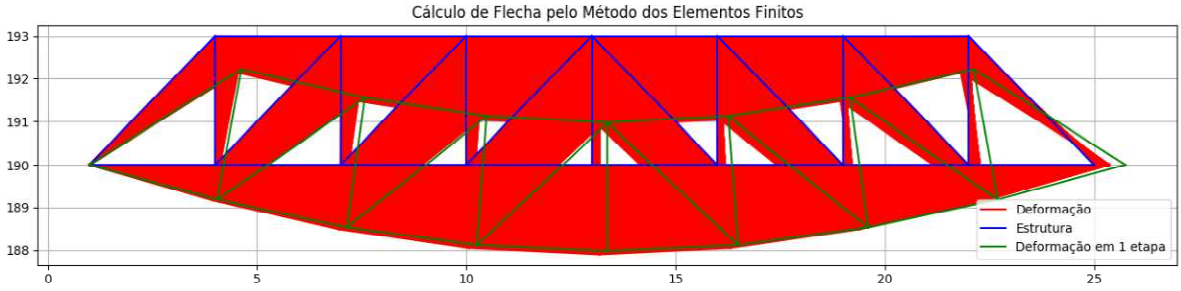
Figura 5.18 - Deformada da estrutura pelo SATE com cem incrementos para P = 10kN



Fonte: Próprio

- Carga de 100 kN:

Figura 5.19 - Deformada da estrutura pelo SATE com cem incrementos para P = 100kN



Fonte: Próprio

Quadro 5.6 - Valores de flechas do SATE para cem incrementos

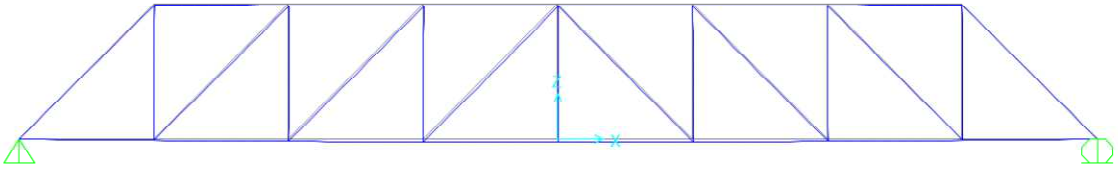
NÓS	FLECHA PARA 3KN (m)		FLECHA PARA 10KN (m)		FLECHA PARA 100KN (m)	
	X	Y	X	Y	X	Y
Nó A	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
Nó B	0,0015	-0,0245	0,0042	-0,0819	-0,0565	-0,8346
Nó C	0,0188	-0,0234	0,0623	-0,0787	0,5481	-0,8585
Nó D	0,0041	-0,0441	0,0126	-0,1476	-0,0341	-1,5143
Nó E	0,0172	-0,0435	0,0563	-0,1457	0,4190	-1,5290

NÓS	FLECHA PARA 3KN (m)		FLECHA PARA 10KN (m)		FLECHA PARA 100KN (m)	
	X	Y	X	Y	X	Y
Nó F	0,0075	-0,0566	0,0236	-0,1894	0,0564	-1,9526
Nó G	0,0145	-0,0564	0,0470	-0,1888	0,2976	-1,9599
Nó H	0,0111	-0,0609	0,0357	-0,2036	0,1821	-2,1006
Nó I	0,0111	-0,0609	0,0357	-0,2036	0,1821	-2,1065
Nó J	0,0147	-0,0566	0,0477	-0,1894	0,3079	-1,9526
Nó K	0,0077	-0,0564	0,0244	-0,1888	0,0666	-1,9599
Nó L	0,0180	-0,0441	0,0588	-0,1476	0,3984	-1,5143
Nó M	0,0050	-0,0435	0,0151	-0,1457	-0,0548	-1,5290
Nó N	0,0207	-0,0245	0,0672	-0,0819	0,4208	-0,8346
Nó O	0,0033	-0,0234	0,0091	-0,0787	-0,1838	-0,8585
Nó P	0,0222	0,0000	0,0714	0,0000	0,3643	0,0000

Fonte: Próprio

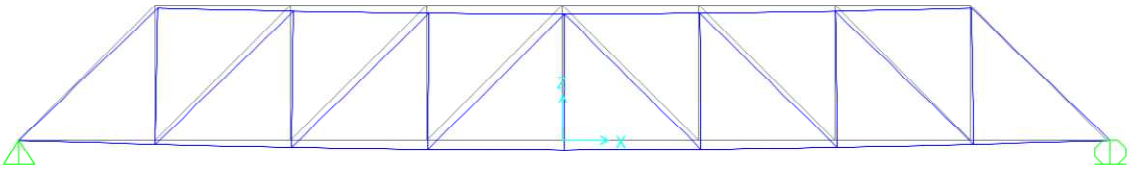
O SAP2000 conta com dois tipos de análises da não linearidade geométrica, a "P-Delta" e a "P-Delta plus Large Displacements", essa segunda é utilizada quando se tem grandes deslocamentos, que é o caso da flecha resultante da carga de 100kN. Dessa forma foi aplicada a análise "P-Delta" para as cargas de 3 e 10 kN e aplicada a análise "P-Delta plus Large Displacements" para a carga de 100kN.

Figura 5.20 – Deformada da estrutura pela análise não linear do SAP2000 para P = 3kN



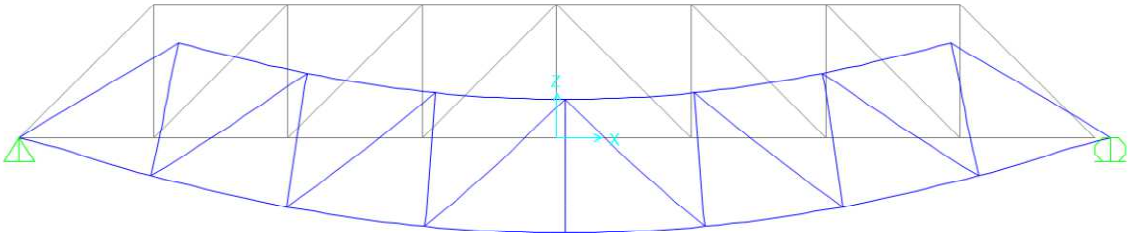
Fonte: SAP2000 V15

Figura 5.21 - Deformada da estrutura pela análise não linear do SAP2000 para P = 10kN



Fonte: SAP2000 V15

Figura 5.22 - Deformada da estrutura pela análise não linear do SAP2000 para P = 100kN



Fonte: SAP2000 V15

Quadro 5.7 - Valores de flechas não lineares do SAP2000

NÓS	FLECHA PARA 3KN (m)		FLECHA PARA 10KN (m)		FLECHA PARA 100KN (m)	
	X	Y	X	Y	X	Y
Nó A	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
Nó B	0,0016	-0,0245	0,0054	-0,0821	-0,0571	-0,8448
Nó C	0,0190	-0,0234	0,0636	-0,0784	0,5596	-0,8750
Nó D	0,0043	-0,0442	0,0144	-0,1478	-0,0364	-1,5318
Nó E	0,0174	-0,0435	0,0583	-0,1456	0,4265	-1,5526
Nó F	0,0077	-0,0567	0,0258	-0,1896	0,0526	-1,9716
Nó G	0,0147	-0,0565	0,0492	-0,1889	0,2998	-1,9855
Nó H	0,0113	-0,0609	0,0379	-0,2037	0,1770	-2,1132
Nó I	0,0113	-0,0609	0,0379	-0,2038	0,1770	-2,1247
Nó J	0,0149	-0,0567	0,0500	-0,1896	0,3014	-1,9716
Nó K	0,0079	-0,0565	0,0265	-0,1889	0,0543	-1,9855
Nó L	0,0183	-0,0442	0,0613	-0,1478	0,3904	-1,5318
Nó M	0,0052	-0,0435	0,0175	-0,1456	-0,0724	-1,5526

NÓS	FLECHA PARA 3KN (m)		FLECHA PARA 10KN (m)		FLECHA PARA 100KN (m)	
	X	Y	X	Y	X	Y
Nó N	0,0210	-0,0245	0,0704	-0,0821	0,4111	-0,8448
Nó O	0,0036	-0,0234	0,0122	-0,0784	-0,2056	-0,8750
Nó P	0,0226	0,0000	0,0758	0,0000	0,3540	0,0000

Fonte: Próprio

Para verificação da eficácia dessa abordagem simplificada da não linearidade geométrica, foi feita a diferença percentual de cada uma das etapas de carga em relação aos resultados obtidos no SAP2000.

Quadro 5.8 - Eficiência do SATE em relação ao SAP2000 para P = 3kN

NÓS	DIFERENÇA PARA DOIS INCREMENTOS DE CARGA (%)		DIFERENÇA PARA DEZ INCREMENTOS DE CARGA (%)		DIFERENÇA PARA CEM INCREMENTOS DE CARGA (%)	
	X	Y	X	Y	X	Y
Nó A	-	-	-	-	-	-
Nó B	-6,25	0,00	-6,25	0,00	-6,25	0,00
Nó C	-0,53	0,00	-1,05	0,00	-1,05	0,00
Nó D	-2,33	-0,23	-4,65	-0,23	-4,65	-0,23
Nó E	-0,57	0,00	-1,15	0,00	-1,15	0,00
Nó F	-1,30	-0,18	-2,60	-0,18	-2,60	-0,18
Nó G	-1,36	-0,18	-1,36	-0,18	-1,36	-0,18
Nó H	-0,88	-0,16	-1,77	0,00	-1,77	0,00
Nó I	-0,88	0,00	-1,77	0,00	-1,77	0,00
Nó J	-0,67	-0,18	-1,34	-0,18	-1,34	-0,18
Nó K	-1,27	-0,18	-2,53	-0,18	-2,53	-0,18
Nó L	-1,09	-0,23	-1,09	-0,23	-1,64	-0,23
Nó M	-1,92	0,00	-3,85	0,00	-3,85	0,00
Nó N	-0,95	0,00	-1,43	0,00	-1,43	0,00

NÓS	DIFERENÇA PARA DOIS INCREMENTOS DE CARGA (%)		DIFERENÇA PARA DEZ INCREMENTOS DE CARGA (%)		DIFERENÇA PARA CEM INCREMENTOS DE CARGA (%)	
	X	Y	X	Y	X	Y
Nó O	-2,78	0,00	-5,56	0,00	-8,33	0,00
Nó P	0,3540	0,0000	-1,77	-	-1,77	-

Fonte: Próprio

Quadro 5.9 - Eficiência do SATE em relação ao SAP2000 para P = 10kN

NÓS	DIFERENÇA COM CARGA APLICADA EM DUAS VEZES (%)		DIFERENÇA COM CARGA APLICADA EM DEZ VEZES (%)		DIFERENÇA COM CARGA APLICADA EM CEM VEZES (%)	
	X	Y	X	Y	X	Y
Nó A	-	-	-	-	-	-
Nó B	-12,96	-0,37	-20,37	-0,24	-22,22	-0,24
Nó C	-1,57	-0,13	-2,04	0,26	-2,04	0,38
Nó D	-6,94	-0,34	-11,81	-0,20	-12,50	-0,14
Nó E	-2,23	-0,27	-3,26	0,00	-3,43	0,07
Nó F	-4,65	-0,32	-7,75	-0,16	-8,53	-0,11
Nó G	-2,64	-0,32	-4,27	-0,11	-4,47	-0,05
Nó H	-3,43	-0,25	-5,54	-0,10	-5,80	-0,05
Nó I	-3,43	-0,29	-5,54	-0,10	-5,80	-0,10
Nó J	-2,80	-0,32	-4,20	-0,16	-4,60	-0,11
Nó K	-4,53	-0,32	-7,17	-0,11	-7,92	-0,05
Nó L	-2,45	-0,34	-3,75	-0,20	-4,08	-0,14
Nó M	-7,43	-0,27	-12,57	0,00	-13,71	0,07
Nó N	-2,70	-0,37	-4,26	-0,24	-4,55	-0,24
Nó O	-13,93	-0,13	-22,95	0,26	-25,41	0,38
Nó P	-3,43	-	-5,41	-	-5,80	-

Fonte: Próprio

Quadro 5.10 - Eficiência do SATE em relação ao SAP2000 para P = 100kN

NÓS	DIFERENÇA COM CARGA APLICADA EM DUAS VEZES (%)		DIFERENÇA COM CARGA APLICADA EM DEZ VEZES (%)		DIFERENÇA COM CARGA APLICADA EM CEM VEZES (%)	
	X	Y	X	Y	X	Y
Nó A	-	-	-	-	-	-
Nó B	-98,07	-1,30	-18,74	-1,08	-1,05	-1,21
Nó C	6,22	-5,61	-0,41	-2,43	-2,06	-1,89
Nó D	-255,22	-1,73	-52,20	-1,12	-6,32	-1,14
Nó E	17,91	-3,34	1,97	-1,72	-1,76	-1,52
Nó F	202,09	-1,89	43,35	-1,01	7,22	-0,96
Nó G	32,25	-2,59	5,44	-1,41	-0,73	-1,29
Nó H	59,38	-1,60	13,45	-0,67	2,88	-0,60
Nó I	59,38	-2,00	13,45	-0,96	2,88	-0,86
Nó J	34,47	-1,89	8,20	-1,01	2,16	-0,96
Nó K	208,84	-2,59	57,27	-1,41	22,65	-1,29
Nó L	30,07	-1,73	7,33	-1,12	2,05	-1,14
Nó M	-184,81	-3,34	-54,01	-1,72	-24,31	-1,52
Nó N	37,53	-1,30	8,98	-1,08	2,36	-1,21
Nó O	-85,36	-5,61	-24,27	-2,43	-10,60	-1,89
Nó P	59,38	-	13,45	-	2,91	-

Fonte: Próprio

Os valores negativos do quadro de eficiência do SATE expressam que os valores de flechas do SATE foram menores em relação ao SAP2000. Após a exibição desses quadros algumas observações podem ser feitas, uma delas é o fato da diferença percentual ser menor para menores cargas, além disso, podemos ver também que quanto maior o número de etapas de aplicação da carga menor a diferença percentual, em especial no eixo Y. Em geral, o eixo X é onde se tem a maior diferença percentual isso se deve também ao fato de que esse eixo é o que tem menores deslocamentos em valores absolutos, fazendo com que mesmo deslocamentos pequenos gerem uma grande diferença percentual.

5.3 DIFERENÇA ENTRE FLECHAS LINEARES E NÃO LINEARES

Agora para a terceira análise comparativa iremos avaliar a diferença percentual entre as flechas calculadas de forma linear e as calculadas de forma não linear geométrica. Essa análise foi feita primeiro para o SATE.

Quadro 5.11 - Comparação entre flechas lineares e não lineares do SATE para P = 3kN

NÓS	DIFERENÇA COM CARGA APLICADA EM DUAS VEZES (%)		DIFERENÇA COM CARGA APLICADA EM DEZ VEZES (%)		DIFERENÇA COM CARGA APLICADA EM CEM VEZES (%)	
	X	Y	X	Y	X	Y
Nó A	-	-	-	-	-	-
Nó B	-6,25	0,00	-6,25	0,00	-6,25	0,00
Nó C	0,00	0,00	-0,53	0,00	-0,53	0,00
Nó D	-2,33	0,00	-4,65	0,00	-4,65	0,00
Nó E	0,00	0,23	-0,58	0,23	-0,58	0,23
Nó F	0,00	0,00	-1,32	0,00	-1,32	0,00
Nó G	-0,68	0,18	-0,68	0,18	-0,68	0,18
Nó H	-0,88	0,00	-1,77	0,16	-1,77	0,16
Nó I	-0,88	0,16	-1,77	0,16	-1,77	0,16
Nó J	0,00	0,00	-0,68	0,00	-0,68	0,00
Nó K	-1,27	0,18	-2,53	0,18	-2,53	0,18
Nó L	-0,55	0,00	-0,55	0,00	-1,10	0,00
Nó M	-1,92	0,23	-3,85	0,23	-3,85	0,23
Nó N	-0,48	0,00	-0,96	0,00	-0,96	0,00
Nó O	-2,78	0,00	-5,56	0,00	-8,33	0,00
Nó P	-0,89	-	-1,33	-	-1,33	-

Fonte: Próprio

Quadro 5.12 - Comparação entre flechas lineares e não lineares do SATE para P = 10kN

NÓS	DIFERENÇA COM CARGA APLICADA EM DUAS VEZES (%)		DIFERENÇA COM CARGA APLICADA EM DEZ VEZES (%)		DIFERENÇA COM CARGA APLICADA EM CEM VEZES (%)	
	X	Y	X	Y	X	Y
Nó A	-	-	-	-	-	-
Nó B	-11,32	0,25	-18,87	0,37	-20,75	0,37
Nó C	-0,63	0,64	-1,11	1,03	-1,11	1,16
Nó D	-5,63	0,20	-10,56	0,34	-11,27	0,41
Nó E	-1,38	0,35	-2,42	0,62	-2,60	0,69
Nó F	-3,53	0,21	-6,67	0,37	-7,45	0,42
Nó G	-1,84	0,27	-3,48	0,48	-3,69	0,53
Nó H	-2,40	0,25	-4,53	0,39	-4,80	0,44
Nó I	-2,40	0,25	-4,53	0,44	-4,80	0,44
Nó J	-1,82	0,21	-3,23	0,37	-3,64	0,42
Nó K	-3,80	0,27	-6,46	0,48	-7,22	0,53
Nó L	-1,48	0,20	-2,80	0,34	-3,13	0,41
Nó M	-6,36	0,35	-11,56	0,62	-12,72	0,69
Nó N	-1,72	0,25	-3,30	0,37	-3,59	0,37
Nó O	-12,50	0,64	-21,67	1,03	-24,17	1,16
Nó P	-2,40	-	-4,40	-	-4,80	-

Fonte: Próprio

Quadro 5.13 - Comparação entre flechas lineares e não lineares do SATE para P = 100kN

NÓS	DIFERENÇA COM CARGA APLICADA EM DUAS VEZES (%)		DIFERENÇA COM CARGA APLICADA EM DEZ VEZES (%)		DIFERENÇA COM CARGA APLICADA EM CEM VEZES (%)	
	X	Y	X	Y	X	Y
Nó A	-	-	-	-	-	-
Nó B	-102,10	2,18	-188,38	2,41	-207,62	2,28
Nó C	-5,65	6,09	-11,54	9,66	-13,00	10,28

NÓS	DIFERENÇA COM CARGA APLICADA EM DUAS VEZES (%)		DIFERENÇA COM CARGA APLICADA EM DEZ VEZES (%)		DIFERENÇA COM CARGA APLICADA EM CEM VEZES (%)	
	X	Y	X	Y	X	Y
Nó D	-60,35	2,43	-112,21	3,07	-123,93	3,04
Nó E	-12,92	3,71	-24,69	5,45	-27,45	5,66
Nó F	-37,69	2,58	-70,43	3,49	-77,88	3,55
Nó G	-18,67	2,97	-35,16	4,22	-38,95	4,35
Nó H	-24,77	2,59	-46,45	3,56	-51,44	3,64
Nó I	-24,77	2,72	-46,45	3,82	-51,44	3,93
Nó J	-18,12	2,58	-34,12	3,49	-37,80	3,55
Nó K	-36,11	2,97	-67,47	4,22	-74,63	4,35
Nó L	-16,41	2,43	-31,03	3,07	-34,42	3,04
Nó M	-64,41	3,71	-119,30	5,45	-131,77	5,66
Nó N	-18,94	2,18	-35,77	2,41	-39,67	2,28
Nó O	-125,08	6,09	-229,75	9,66	-253,17	10,28
Nó P	-24,77	-	-46,45	-	-51,43	-

Fonte: Próprio

Por fim se fez essa mesma análise com os resultados das flechas do SAP2000, o quadro 5.14 mostra a diferença percentual entre os valores de flechas obtidos com análise linear e não linear geométrica.

Quadro 5.14 - Comparação entre flechas lineares e não lineares do SAP2000

NÓS	CARGA DE 3KN (%)		CARGA DE 10KN (%)		CARGA DE 100KN (%)	
	X	Y	X	Y	X	Y
Nó A	-	-	-	-	-	-
Nó B	0,00	0,00	1,89	0,61	-208,76	3,50
Nó C	0,53	0,00	0,95	0,64	-11,20	12,37
Nó D	0,00	0,23	0,70	0,54	-125,54	4,21
Nó E	0,58	0,23	0,87	0,62	-26,16	7,27

NÓS	CARGA DE 3KN (%)		CARGA DE 10KN (%)		CARGA DE 100KN (%)	
	X	Y	X	Y	X	Y
Nó F	0,00	0,18	1,18	0,53	-79,38	4,53
Nó G	0,68	0,18	0,82	0,53	-38,52	5,68
Nó H	0,00	0,16	1,07	0,49	-52,81	4,23
Nó I	0,00	0,16	1,07	0,54	-52,81	4,80
Nó J	0,00	0,18	1,01	0,53	-39,12	4,53
Nó K	0,00	0,18	0,76	0,53	-79,32	5,68
Nó L	0,55	0,23	0,82	0,54	-35,76	4,21
Nó M	0,00	0,23	1,16	0,62	-141,97	7,27
Nó N	0,48	0,00	2,18	0,61	-41,08	3,50
Nó O	0,00	0,00	1,67	0,64	-271,33	12,37
Nó P	0,44	-	1,07	-	-52,81	-

Fonte: Próprio

Percebe-se que a coordenada X é muito variável, em especial para o carregamento de 100kN, isso acontece porque quando a carga é aplicada de uma vez (análise linear) a maioria dos nós é deslocada no sentido positivo do eixo X, já quando é levado em conta a não linearidade geométrica, essa deformação abrupta não ocorre, pelo contrário, a estrutura tem mais tempo de se deformar de modo uniforme, fazendo alguns nós serem deslocados no sentido negativo do eixo X devido a inclinação mais acomodada da estrutura provocada pela rotação, esse fenômeno pode ser facilmente visível observando os nós de qualquer um dos gráficos com não linearidade geométrica apresentados. Além disso, a deformação excessiva do eixo Y, no caso da carga de 100kN, puxa todos os pontos um pouco mais para o sentido negativo do eixo x.

Analisando os quadros pode-se notar também que a diferença percentual de flecha no eixo Y, é maior quanto maior for o número de etapas aplicadas, além disso, essa diferença percentual também aumenta com o aumento da carga aplicada, fato que pode ser observado comparando os resultados para os carregamentos de 3, 10 e 100 kN.

6 CONCLUSÃO

Após a exposição de todos os resultados é possível notar que a não linearidade geométrica não teve um impacto tão relevante para o carregamento de 3kN, onde vemos que a maior diferença percentual do eixo Y foi de 0,23%, isso acontece principalmente devido ao fato da carga e deslocamento serem baixos, nesse tipo de caso a não linearidade pode ser desprezada.

Também vimos que a maior diferença percentual, no eixo Y, entre a análise não linear do SATE e do SAP2000 foi para a carga de 100kN no valor de 1,89%. Já era esperado que houvesse diferença entre os valores do SATE e do SAP2000, uma vez que o SAP2000 usa métodos contínuos enquanto que o SATE usa uma abordagem mais simplificada, apenas dividindo a carga e aplicando após cada deformação da estrutura. Entretanto, talvez esse valor de diferença percentual pudesse ser reduzido ao aumentar o número de divisões em que a carga seria aplicada.

Para grandes deslocamentos a análise da estrutura levando em conta a não linearidade geométrica torna-se praticamente obrigatória, como podemos perceber nas análises não lineares com carga de 100kN, onde a diferença percentual entre os métodos lineares e não lineares chegam à 10,28% no SATE e 12,37% no SAP2000.

Em suma o conceito de análise não linear geométrica é bem aplicado quando utiliza-se a “análise por etapas” que foi aqui explicada e exemplificada. Esse tipo de análise por etapas pode ser muito bem implementada se levarmos em conta as fases de carregamento que acontecem em uma obra, onde existem os carregamentos de peso próprio, de construção, projeto e etc. Sabe-se que para uma estrutura se deformar ela precisa de tempo, então em uma situação real, de nada adiantaria aplicar a carga por etapa, porém com intervalo de tempo muito curto entre as aplicações de cada passo, dessa forma, utilizar as etapas de carregamento prevista para a obra pode otimizar ainda mais a análise por etapas proposta neste trabalho.

REFERÊNCIAS

BENJAMIN, Adilson Carvalho. **Análise não-linear geométrica de pórticos tridimensionais pelo método dos elementos finitos**. Rio de Janeiro, RJ, 1982.

CUNHA, Pedro Filipe de Luna. **Rotinas computacionais para análise não linear geométrica de estruturas reticuladas**. Recife, PE, 2017.

LACERDA, Estéfane George Macedo. **Análise não linear de treliças pelo método dos elementos finitos posicional**. Natal, RN, 2014.

OLIVEIRA, Ricardo Lopes De. **Elaboração de algoritmo com formulação não linear geométrica para o cálculo de treliças tridimensionais**. Campo Mourão, PR, 2015.

PINTO, Rivelli da Silva; RAMALHO, Marcio Antonio. **Não-linearidade física e geométrica no projeto de edifícios usuais de concreto armado**. São Carlos, SP, 2002.

SILVA, Wagner Queiroz. **Sobre análise não linear geométrica de edifícios considerando o empenamento dos núcleos estruturais e a interação solo-estrutura**. São Carlos, SP, 2014.

SOUZA, Luiz Antonio Farani De. **Análise não linear geométrica de treliças planas por meio do método dos elementos finitos**. Maringá, PR, 2015.

VAZ, Luiz Eloy. **Método dos elementos finitos em análise de estruturas**. Rio de Janeiro: Elsevier, 2011.

APÊNDICE A – CÓDIGO FONTE PRINCIPAL (SATE.py)

O código principal é o responsável por capturar todos os dados fornecidos pelo usuário através de uma interface amigável e utilizar o código auxiliar para o cálculo das flechas pelo MEF, também é no código principal onde a força é aplicada em passos, a fim de se implementar a não linearidade geométrica.

```
1  ########## Importanto Bibliotecas Necessárias ##########
2  import tkinter
3  import elem_finitos
4  from pylab import *
5
6  ########## Definindo Variáveis e Listas #########
7  MODO=None
8  anterior=False
9  criar_linha=False
10  pos_click=[]
11  linhas=[]
12  E=200000000
13  A=0.0001
14  Grid_x=1
15  Grid_y=1
16  Points=[]
17  Points_ativos=[]
18  Points_selects=[]
19  Points_travados=[]
20  Valor_forca={}
21  Labels_forcas={}
22  Points_forcas=[]
23  Linhas_seletcs=[]
24  Point_Exist=False
25
26  ########## Função Gráfica Responsável pela Interação do Grid ao Movimento
do Mouse ##########
27  def Movimento(event):
28      event.x+=hbar.get()[0]*2000
29      event.y+=vbar.get()[0]*2000
30      global anterior
31      x = int(round(event.x/(Grid_x*20),0))
32      y = int(round(event.y/(Grid_y*20),0))
33      tag = "grid%i,%i"%(x,y)
34      p='point%i,%i'%(int(round(event.x/20,0)),int(round(event.y/20,0)))
35      borda=1
36      if p in Points_ativos:
37          tag=p
38          borda=3
39      canvas.itemconfigure(tag,fill='#ff0000',outline='#ff0000',width=borda)
40      if anterior and anterior!=tag:
```

```
41          if anterior in Points_selects:
42              canvas.itemconfigure(anterior,fill='#ff0000',outline='#ff0000',width=borda)
43          else:
44              canvas.itemconfigure(anterior,fill='#000000',outline='#000000',width=1)
45      anterior = tag
46      if criar_linha:
47          canvas.delete(linhas[-1])
48          if p in Points_ativos:
49              canvas.create_line(pos_Click[0],pos_Click[1],int(round(event.x/20,0))*20,int(
round(event.y/20,0))*20,tags=linhas[-1],width=2)
50          else:
51              canvas.create_line(pos_Click[0],pos_Click[1],x*Grid_x*20,y*Grid_y*20,tags=li
nhas[-1],width=2)
52
53  ########## Função para Seleção e Destaque das Linhas Clicadas
##########
54  def select2(event):
55      global w,p,x1,y1,e
56      event.x+=hbar.get()[0]*2000
57      event.y+=vbar.get()[0]*2000
58      x=int(round(event.x/20,0))
59      y=int(round(event.y/20,0))
60      e=event
61      tag=event.widget.find_closest(canvas.canvasx(event.x),canvas.canvasy(event.y))[0]
62      if MODO=='select':
63          if canvas.itemcget(tag,'fill')=='red':
64              canvas.itemconfigure(tag,fill='black')
65              Linhas_seletcs.remove(canvas.gettags(tag)[0])
66          elif canvas.itemcget(tag,'fill')=='black':
67              canvas.itemconfigure(tag,fill='red')
68              Linhas_seletcs.append(canvas.gettags(tag)[0])
69
70  ########## Função para Seleção dos Pontos Clicados ##########
71  def select(event):
72      global w,p,x1,y1
73      event.x+=hbar.get()[0]*2000
74      event.y+=vbar.get()[0]*2000
75      x1=event.x
76      y1=event.y
77      p='point%i,%i'%(int(round(event.x/20,0)),int(round(event.y/20,0)))
78      if MODO=='select':
79          if p in Points_ativos:
80              if p in Points_selects:
81                  canvas.itemconfigure(p,fill='#000000',outline='#000000',width=1)
82                  Points_selects.remove(p)
83              else:
84                  canvas.itemconfigure(p,fill='red',outline='red', stipple='',width=3)
85                  Points_selects.append(p)
86
87  ########## Função Responsável por Desenhar Linhas no Grid ##########
88  def PointClick(event):
89      event.x+=hbar.get()[0]*2000
90      event.y+=vbar.get()[0]*2000
```

```
 91      global criar_linha, pos_Click,Point_Exist
 92      x = int(round(event.x/(Grid_x*20),0))
 93      y = int(round(event.y/(Grid_y*20),0))
 94      tag_grid = "grid%i,%i"%(x,y)
 95      x1=event.x
 96      y1=event.y
 97      p='point%i,%i'%(int(x*Grid_x),int(y*Grid_y))
 98      if MODO=='draw':
 99          if p in Points_ativos:
100              Point_Exist=True
101              if criar_linha:
102                  criar_linha=False
103                  canvas.delete(linhas[-1])
104                  tag='linha%i,%i,%i,%i'%(int(pos_Click[0]/20),int(pos_Click[1]/20),int(x*
Grid_x),int(y*Grid_y))
105                  linhas.remove(linhas[-1])
106                  linhas.append(tag)
107                  canvas.create_line(pos_Click[0],pos_Click[1],int(x*Grid_x)*20,int(y*Grid_y
)*20,tags=tag,width=2)
108                  canvas.tag_bind(tag, '<ButtonPress-1>', select2)
109                  canvas.tag_lower(tag)
110              else:
111                  criar_linha=True
112                  tag='linha%i,%i,%i,%i'%(int(x*Grid_x),int(y*Grid_y),int(x*Grid_x),int(y*Gr
id_y))
113                  linhas.append(tag)
114                  canvas.create_line(int(x*Grid_x)*20,int(y*Grid_y)*20,int(x*Grid_x)*20,int
(y*Grid_y)*20,tags=tag,width=2)
115                  canvas.tag_bind(tag, '<ButtonPress-1>', select2)
116                  pos_Click=[int(x*Grid_x)*20,int(y*Grid_y)*20]
117                  canvas.tag_lower(tag)
118          else:
119              Point_Exist=False
120              if criar_linha:
121                  criar_linha=False
122                  canvas.delete(linhas[-1])
123                  tag='linha%i,%i,%i,%i'%(int(pos_Click[0]/20),int(pos_Click[1]/20),int(x*
Grid_x),int(y*Grid_y))
124                  linhas.remove(linhas[-1])
125                  linhas.append(tag)
126                  canvas.create_line(pos_Click[0],pos_Click[1],x*Grid_x*20,y*Grid_y*20,tag
s=tag,width=2)
127                  canvas.tag_bind(tag, '<ButtonPress-1>', select2)
128                  canvas.create_oval(x*Grid_x*20-1.5,y*Grid_y*20-
1.5,x*Grid_x*20+1.5,y*Grid_y*20+1.5,fill='#000000',tags=p)
129                  canvas.tag_lower(tag)
130                  Points_ativos.append(p)
131              else:
132                  criar_linha=True
133                  tag='linha%i,%i,%i,%i'%(int(x*Grid_x),int(y*Grid_y),int(x*Grid_x),int(y*Gr
id_y))
134                  linhas.append(tag)
135                  canvas.create_line(x*Grid_x*20,y*Grid_y*20,x*Grid_x*20,y*Grid_y*20,ta
```

```
gs=tag,width=2)
136                 canvas.tag_bind(tag, '<ButtonPress-1>', select2)
137                 canvas.create_oval(x*Grid_x*20-1.5,y*Grid_y*20-
1.5,x*Grid_x*20+1.5,y*Grid_y*20+1.5,fill='#000000',tags=p)
138                 canvas.tag_lower(tag)
139                 Points_ativos.append(p)
140                 pos_Click=[x*Grid_x*20,y*Grid_y*20]
141
142     ########## Função que Define o Espaçamento do Grid ##########
143     def aplicar_grid():
144         global Grid_x,Grid_y
145         Grid_x=var3.get()
146         Grid_y=var4.get()
147         CreateGrid()
148
149     ########## Função que Define as Caracteristicas da Treliça ##########
150     def aplicar_material():
151         global E,A
152         E=var1.get()
153         A=var2.get()
154
155     ########## Função que Aplica o Espaçamento do Grid Definido
##########
156     def CreateGrid():
157         global Points
158         for i in Points:
159             canvas.delete(i)
160         Points=[]
161         for i in range(100):
162             for j in range(100):
163                 tag="grid%i,%i"%(i,j)
164                 canvas.create_oval(i*Grid_x*20-1,j*Grid_y*20-
1,i*Grid_x*20+1,j*Grid_y*20+1,fill='#000000',tags=tag)
165                 Points.append(tag)
166
167     ########## Função que Exclui o Grid ##########
168     def RemoverGrid():
169         global Points
170         for i in Points:
171             canvas.delete(i)
172         Points=[]
173
174     ########## Função que Ativa o Modo de Seleção ##########
175     def selecionar():
176         global MODO
177         MODO='select'
178         bt_select.configure(background='lightgreen')
179         bt_draw.configure(background='SystemButtonFace')
180         canvas.bind('<Button-1>',select)
181
182     ########## Função que Ativa o Modo de Desenho ##########
183     def desenhar():
184         global MODO
```

```
185         MODO='draw'
186         bt_draw.configure(background='lightgreen')
187         bt_select.configure(background='SystemButtonFace')
188         canvas.bind('<Button-1>',PointClick)
189 
190     ########## Função que Cancela a Criação de uma Linha (associado a tecla
"ESC") ##########
191     def Cancelar(event):
192         global criar_linha
193         if criar_linha:
194             criar_linha=False
195             canvas.delete(linhas[-1])
196             linhas.remove(linhas[-1])
197             if not Point_Exist:
198                 canvas.delete(Points_ativos[-1])
199                 Points_ativos.remove(Points_ativos[-1])
200         rem=list(Points_selects)
201         for p in rem:
202             canvas.itemconfigure(p,fill='#000000',outline='#000000')
203             Points_selects.remove(p)
204         rem=list(Linhas_seletcs)
205         for p in rem:
206             canvas.itemconfigure(p,fill='black')
207             Linhas_seletcs.remove(p)
208 
209     ########## Função que Aplica as Forças nos Nós da Treliça Selecionados
##########
210     def carregamento():
211         try:
212             f=var6.get()
213             if f!=0:
214                 for i in Points_selects:
215                     l = [int(j) for j in i[5:].split(',')]
216                     fx='ForçaX%i,%i'%(l[0],l[1])
217                     lx='LabelX%i,%i'%(l[0],l[1])
218                     if fx in Points_forcas:
219                         rem_carregamento(fx,lx)
220                     if f<0:
221                         canvas.create_polygon(l[0]*20+3,l[1]*20,l[0]*20+3+5,l[1]*20-
5,l[0]*20+3+5,l[1]*20,l[0]*20+25+3,l[1]*20,l[0]*20+5+3,l[1]*20,l[0]*20+5+3,l[1
]*20+5,l[0]*20+3,l[1]*20,tags=fx,fill='#000000',outline='#000000')
222                         canvas.create_text(l[0]*20+30+3+len(str(f))*3,l[1]*20,text=str(f),ta
g=lx)
223                         Valor_forca[fx]=f
224                         Points_forcas.append(fx)
225                     elif f>0:
226                         canvas.create_polygon(l[0]*20-3,l[1]*20,l[0]*20-5-3,l[1]*20-
5,l[0]*20-5-3,l[1]*20,l[0]*20-25-3,l[1]*20,l[0]*20-5-3,l[1]*20,l[0]*20-5-
3,l[1]*20+5,l[0]*20-3,l[1]*20,tags=fx,fill='#000000',outline='#000000')
227                         canvas.create_text(l[0]*20-30-3-
len(str(f))*3,l[1]*20,text=str(f),tag=lx)
228                         Valor_forca[fx]=f
229                         Points_forcas.append(fx)
```

```
230         else:
231             for i in Points_selects:
232                 l = [int(j) for j in i[5:].split(',')]
233                 fx='ForçaX%i,%i'%(l[0],l[1])
234                 lx='LabelX%i,%i'%(l[0],l[1])
235                 if fx in Points_forcas:
236                     rem_carregamento(fx,lx)
237     except:None
238     try:
239         f=var5.get()
240         if f!=0:
241             for i in Points_selects:
242                 l = [int(j) for j in i[5:].split(',')]
243                 fy='ForçaY%i,%i'%(l[0],l[1])
244                 ly='LabelY%i,%i'%(l[0],l[1])
245                 if fy in Points_forcas:
246                     rem_carregamento(fy,ly)
247                 if f<0:
248                     canvas.create_polygon(l[0]*20,l[1]*20-3,l[0]*20-5,l[1]*20-5-
3,l[0]*20,l[1]*20-5-3,l[0]*20,l[1]*20-25-3,l[0]*20,l[1]*20-5-3,l[0]*20+5,l[1]*20-
5-3,l[0]*20,l[1]*20-3,tags=fy,fill='#000000',outline='#000000')
249                     canvas.create_text(l[0]*20-len(str(f))/2,l[1]*20-35-
3,text=str(f),tag=ly)
250                     Valor_forca[fy]=f
251                     Points_forcas.append(fy)
252                 elif f>0:
253                     canvas.create_polygon(l[0]*20,l[1]*20-25-3,l[0]*20-5,l[1]*20-20-
3,l[0]*20,l[1]*20-20-3,l[0]*20,l[1]*20-3,l[0]*20,l[1]*20-20-3,l[0]*20+5,l[1]*20-
20-3,l[0]*20,l[1]*20-25-3,tags=fy,fill='#000000',outline='#000000')
254                     canvas.create_text(l[0]*20-len(str(f))/2,l[1]*20-35-
3,text=str(f),tag=ly)
255                     Valor_forca[fy]=f
256                     Points_forcas.append(fy)
257         else:
258             for i in Points_selects:
259                 l = [int(j) for j in i[5:].split(',')]
260                 fy='ForçaY%i,%i'%(l[0],l[1])
261                 ly='LabelY%i,%i'%(l[0],l[1])
262                 if fy in Points_forcas:
263                     rem_carregamento(fy,ly)
264     except:None
265
266  ########## Função que Remove as Forças nos Nós da Treliça Selecionados
##########
267  def remover_carregamento():
268     for i in Points_selects:
269         l = [int(j) for j in i[5:].split(',')]
270         fx='ForçaX%i,%i'%(l[0],l[1])
271         fy='ForçaY%i,%i'%(l[0],l[1])
272         lx='LabelX%i,%i'%(l[0],l[1])
273         ly='LabelY%i,%i'%(l[0],l[1])
274         rem_carregamento(fx,lx)
275         rem_carregamento(fy,ly)
```

```
276
277  ########## Complemento da Função "remover_carregamento"
##########
278  def rem_carregamento(f,l):
279      try:
280          canvas.delete(f)
281          canvas.delete(l)
282          Valor_forca.pop(f)
283      except:None
284
285  ########## Função que Insere Apoios nos Pontos Selecionados
##########
286  def apoioP():
287      x=var_x.get()
288      y=var_y.get()
289      pontos=[j[0] for j in Points_travados]
290      genero=[j[1] for j in Points_travados]
291      if x and y:
292          for i in Points_selects:
293              l = [int(j) for j in i[5:].split(',')]
294              if i in pontos:
295                  canvas.delete('Apoio%i,%i'%(l[0],l[1]))
296                  Points_travados.remove([i,genero[pontos.index(i)]])
297              canvas.create_polygon(l[0]*20,l[1]*20,l[0]*20-10,l[1]*20+10,l[0]*20-
13,l[1]*20+14,l[0]*20-10,l[1]*20+10,l[0]*20-5,l[1]*20+10,l[0]*20-
8,l[1]*20+14,l[0]*20-5,l[1]*20+10,l[0]*20,l[1]*20+10,l[0]*20-
3,l[1]*20+14,l[0]*20,l[1]*20+10,l[0]*20+5,l[1]*20+10,l[0]*20+2,l[1]*20+14,l[0]
*20+5,l[1]*20+10,l[0]*20+10,l[1]*20+10,l[0]*20+7,l[1]*20+14,l[0]*20+10,l[1]*2
0+10,tags='Apoio%i,%i'%(l[0],l[1]),fill='#ffffff',outline='#000000')
298              Points_travados.append([i,'xy'])
299              canvas.tag_lower('Apoio%i,%i'%(l[0],l[1]))
300      elif y:
301          for i in Points_selects:
302              l = [int(j) for j in i[5:].split(',')]
303              if i in pontos:
304                  canvas.delete('Apoio%i,%i'%(l[0],l[1]))
305                  Points_travados.remove([i,genero[pontos.index(i)]])
306              canvas.create_polygon(l[0]*20,l[1]*20,l[0]*20-
10,l[1]*20+10,l[0]*20+10,l[1]*20+10,tags='Apoio%i,%i'%(l[0],l[1]),fill='#ffffff',outlin
e='#000000')
307              Points_travados.append([i,'y'])
308              canvas.tag_lower('Apoio%i,%i'%(l[0],l[1]))
309      elif x:
310          for i in Points_selects:
311              l = [int(j) for j in i[5:].split(',')]
312              if i in pontos:
313                  canvas.delete('Apoio%i,%i'%(l[0],l[1]))
314                  Points_travados.remove([i,genero[pontos.index(i)]])
315              canvas.create_polygon(l[0]*20,l[1]*20,l[0]*20+10,l[1]*20-
10,l[0]*20+10,l[1]*20+10,tags='Apoio%i,%i'%(l[0],l[1]),fill='#ffffff',outline='#000000')
316              Points_travados.append([i,'x'])
317              canvas.tag_lower('Apoio%i,%i'%(l[0],l[1]))
318
```

```
319   ########## Função Principal, Responsável por Extrair os Dados Necessários da
Interface e Repassar para o Algoritmo Secundário "elem_finitos", que irá Aplicar o MEF.
Os Gráficos de Deformação Também são Gerados Aqui  ##########
320   def Calcular():
321       tamanho_Fonte=var_Fonte.get()
322       Nome_Colunas=["X","Y","X","Y","X","Y"]
323       Cor_Colunas=["blue","blue","red","red","green","green"]
324       Dados=[[],[],[],[],[],[]]
325       modulo=lambda no1,no2: ((no2[0]-no1[0])**2+(no1[1]-no2[1])**2)**(1/2)
326       global flexa,coord,conect,travados,Forcas,E,A
327       Num_etapas=var_etapas.get()
328       coord=[]
329       for i in Points_ativos:
330           pos=canvas.coords(i)
331           coord.append([int(round((pos[0]+pos[2])/(2*20),0)),200-
int(round((pos[1]+pos[3])/(2*20),0))])
332       coord.sort()
333       conect=[]
334       for i in linhas:
335           pos=canvas.coords(i)
336           conec=[coord.index([int(round(pos[0]/(20),0)),200-
int(round(pos[1]/(20),0))])+1,coord.index([int(round(pos[2]/(20),0)),200-
int(round(pos[3]/(20),0))])+1]
337           conec.sort()
338           conect.append(conec)
339       travados=[]
340       for i in Points_travados:
341           pos=canvas.coords(i[0])
342           ind=coord.index([int(round((pos[0]+pos[2])/(2*20),0)),200-
int(round((pos[1]+pos[3])/(2*20),0))])
343           if i[1]=='xy':
344               travados.append(ind*2)
345               travados.append(ind*2+1)
346           elif i[1]=='x':
347               travados.append(ind*2)
348           elif i[1]=='y':
349               travados.append(ind*2+1)
350       Forcas=[0]*len(coord)*2
351       Forcas2=[0]*len(coord)*2
352       for i in coord:
353           fx='ForçaX%i,%i'%(i[0],(200*Grid_x-i[1]))
354           fy='ForçaY%i,%i'%(i[0],(200*Grid_y-i[1]))
355           try:
356               Forcas[coord.index(i)*2]=Valor_forca[fx]/Num_etapas
357               Forcas2[coord.index(i)*2]=Valor_forca[fx]
358           except:
359               Forcas[coord.index(i)*2]=0
360               Forcas2[coord.index(i)*2]=0
361           try:
362               Forcas[coord.index(i)*2+1]=Valor_forca[fy]/Num_etapas
363               Forcas2[coord.index(i)*2+1]=Valor_forca[fy]
364           except:
365               Forcas[coord.index(i)*2+1]=0
```

```
366             Forcas2[coord.index(i)*2+1]=0
367         deformou=False
368         for i in coord:
369             Dados[0].append(i[0])
370             Dados[1].append(i[1])
371         list_A=[A]*len(conect)
372         for i in range(Num_etapas):
373             list_L=[]
374             for i in conect:
375                 no1=coord[i[0]-1]
376                 no2=coord[i[1]-1]
377                 list_L.append(modulo(no1,no2))
378             flexa=elem_finitos.principal(coord,conect,E,list_A,travados,Forcas)
379             cont=0
380             for i in range(len(coord)):
381                 for j in range(len(coord[0])):
382                     if 2*i+j not in travados:
383                         coord[i][j]-=flexa[cont]
384                         cont+=1
385             list_A2=[]
386             for i in conect:
387                 no1=coord[i[0]-1]
388                 no2=coord[i[1]-1]
389                 L=modulo(no1,no2)
390                 list_A2.append(list_A[conect.index(i)]*list_L[conect.index(i)]/L)
391             list_A=list(list_A2)
392             x=[[coord[i[0]-1][0],coord[i[1]-1][0]] for i in conect]
393             y=[[coord[i[0]-1][1],coord[i[1]-1][1]] for i in conect]
394             if deformou==False:
395                 plot(x[0], y[0],label='Deformação',color='red')
396                 for i in range(1,len(x)):
397                     plot(x[i], y[i],color='red')
398                 deformou=True
399             else:
400                 for i in range(len(x)):
401                     plot(x[i], y[i],color='red')
402         for i in coord:
403             Dados[2].append(i[0])
404             Dados[3].append(i[1])
405         coord=[]
406         for i in Points_ativos:
407             pos=canvas.coords(i)
408             coord.append([int(round((pos[0]+pos[2])/(2*20),0)),200-
int(round((pos[1]+pos[3])/(2*20),0))])
409         coord.sort()
410         list_A=[A]*len(conect)
411         x=[[coord[i[0]-1][0],coord[i[1]-1][0]] for i in conect]
412         y=[[coord[i[0]-1][1],coord[i[1]-1][1]] for i in conect]
413         plot(x[0], y[0],label='Estrutura',color='blue')
414         for i in range(1,len(x)):
415             plot(x[i], y[i],color='blue')
416         flexa2=elem_finitos.principal(coord,conect,E,list_A,travados,Forcas2)
417         cont=0
```

```
418     for i in range(len(coord)):
419         for j in range(len(coord[0])):
420             if 2*i+j not in travados:
421                 coord[i][j]-=flexa2[cont]
422                 cont+=1
423     x=[[coord[i[0]-1][0],coord[i[1]-1][0]] for i in conect]
424     y=[[coord[i[0]-1][1],coord[i[1]-1][1]] for i in conect]
425     plot(x[0], y[0],label='Deformação em 1 etapa',color='green')
426     for i in range(1,len(x)):
427         plot(x[i], y[i],color='green')
428     for i in coord:
429         Dados[4].append(i[0])
430         Dados[5].append(i[1])
431     Celulas=[]
432     for i in range(len(Dados[0])):
433         Celulas.append([])
434         for j in range(len(Dados)):
435             Celulas[i].append(Dados[j][i])
436     tabela=table(cellText=Celulas,rowLabels=['Nó
'+str(i+1) for i in range(len(Celulas))],colColours=Cor_Colunas,colLabels=Nome_Colunas,loc
='bottom')
437     tabela.auto_set_font_size(False)
438     tabela.set_fontsize(tamanho_Fonte)
439     tabela.scale(1,2)
440     subplots_adjust(left=0.05,right=0.99, bottom=0.60,top=0.96)
441     legend()
442     title('Cálculo de Flecha pelo Método dos Elementos Finitos')
443     grid(True)
444     show()
445
446  ########## Função que Remove Apoios nos Pontos Selecionados
##########
447  def Remover_apoio():
448     pontos=[j[0] for j in Points_travados]
449     genero=[j[1] for j in Points_travados]
450     for i in Points_selects:
451         l = [int(j) for j in i[5:].split(',')]
452         tag='Apoio%i,%i'%(l[0],l[1])
453         if i in pontos:
454             canvas.delete(tag)
455             Points_travados.remove([i,genero[pontos.index(i)]])
456
457  ########## Função que Remove Pontos, Apoios e Linhas (associado a tecla
"DELETE" ou "DEL" ##########
458  def Deletar(event):
459     print(Points_selects,Points_ativos)
460     def point_alone(p):
461         pontos=[j[0] for j in Points_travados]
462         genero=[j[1] for j in Points_travados]
463         l = [int(j) for j in p.split(',')]
464         fx='ForçaX%i,%i'%(l[0],l[1])
465         fy='ForçaY%i,%i'%(l[0],l[1])
466         lx='LabelX%i,%i'%(l[0],l[1])
```

```
467         ly='LabelY%i,%i'%(l[0],l[1])
468         rem_carregamento(fx,lx)
469         rem_carregamento(fy,ly)
470         tag='Apoio%i,%i'%(l[0],l[1])
471         if 'point'+p in pontos:
472             canvas.delete(tag)
473             Points_travados.remove(['point'+p,genero[pontos.index('point'+p)]])
474         count=0
475         for i in linhas:
476             if p ==i[11:] or p==i[5:10]:
477                 count+=1
478         if count==0:
479             canvas.delete('point'+p)
480             Points_ativos.remove('point'+p)
481     global linhas
482     rem=[]
483     rem_p=[]
484     for i in Linhas_seletcs:
485         rem.append(i)
486     remover_carregamento()
487     Remover_apoio()
488     for i in Points_selects:
489         for j in linhas:
490             if i[5:]==j[11:] or i[5:]==j[5:10]:
491                 rem.append(j)
492         canvas.delete(i)
493         rem_p.append(i)
494     for i in rem_p:
495         Points_selects.remove(i)
496         Points_ativos.remove(i)
497     for k in set(rem):
498         linhas.remove(k)
499         canvas.delete(k)
500         if k in Linhas_seletcs:
501             Linhas_seletcs.remove(k)
502         try:point_alone(k[11:])
503         except:None
504         try:point_alone(k[5:10])
505         except:None
506
507 ########## Função que Reseta o Grid, Voltando a Configuração Original
##########
508 def Resetar():
509     global E,A,Grid_x,Grid_y,Point_Exist,MODO,anterior,criar_linha,Points_selects
510     for i in linhas:
511         canvas.delete(i)
512     for i in Points_ativos:
513         canvas.delete(i)
514     Points_selects=list(Points_ativos)
515     remover_carregamento()
516     Remover_apoio()
517     E=200000000
518     A=0.0001
```

```
519         Grid_x=1
520         Grid_y=1
521         Point_Exist=False
522         linhas.clear()
523         Points_ativos.clear()
524         Points_selects.clear()
525         Points_travados.clear()
526         MODO=None
527         anterior=False
528         criar_linha=False
529         bt_draw.configure(background='SystemButtonFace')
530         bt_select.configure(background='SystemButtonFace')
531         var1.set(E)
532         var2.set(A)
533         var3.set(Grid_x)
534         var4.set(Grid_y)
535         var5.set(0.0)
536         var6.set(0.0)
537         var_x.set(False)
538         var_y.set(False)
539         var_etapas.set(1)
540         var_Fonte.set(9)
541 
542     ########## Declaração de Toda Interface Gráfica, como Botões, Textos,
Caixas de Entrada de Texto, Menu, e etc. ##########
543     tk = tkinter.Tk()
544     tk.title('SATE')
545     f=tkinter.Frame(tk)
546     f2=tkinter.Frame(f)
547     tools=tkinter.Frame(f,width=400)
548     propriedades_material=tkinter.Frame(tools,width=400)
549     propriedades_grid=tkinter.Frame(tools,width=400)
550     propriedades_carregamento=tkinter.Frame(tools,width=400)
551     labels_material=tkinter.Frame(propriedades_material,width=400)
552     labels_grid=tkinter.Frame(propriedades_grid,width=400)
553     labels_carregamento=tkinter.Frame(propriedades_carregamento,width=400)
554     entrys_material=tkinter.Frame(propriedades_material,width=400)
555     entrys_grid=tkinter.Frame(propriedades_grid,width=400)
556     entrys_carregamento=tkinter.Frame(propriedades_carregamento,width=400)
557     bts_carregamento=tkinter.Frame(tools,width=400)
558     fonte_Frame=tkinter.Frame(tools,width=400)
559     fonte_Texto=tkinter.Label(fonte_Frame,text='Fonte da Tabela')
560     lb_modo=tkinter.Label(tools,text='MODO')
561     bts_modo=tkinter.Frame(tools,width=400)
562     lb_material=tkinter.Label(tools,text='MATERIAL')
563     lb_grid=tkinter.Label(tools,text='GRID')
564     lb_carregamento=tkinter.Label(tools,text='CARREGAMENTO')
565     lb_apoio=tkinter.Label(tools,text='APOIOS')
566     lb_executar=tkinter.Label(tools,text='Nº de Aplicações')
567     label1=tkinter.Label(labels_material,text='E (kPa): ')
568     label2=tkinter.Label(labels_material,text='A (m²): ')
569     label3=tkinter.Label(labels_grid,text='X (m): ')
570     label4=tkinter.Label(labels_grid,text='Y (m): ')
```

```
label5=tkinter.Label(labels_carregamento,text='FY(kN): ')
label6=tkinter.Label(labels_carregamento,text='FX(kN): ')
var1=tkinter.DoubleVar(tk)
var1.set(E)
var2=tkinter.DoubleVar(tk)
var2.set(A)
var3=tkinter.DoubleVar(tk)
var3.set(Grid_x)
var4=tkinter.DoubleVar(tk)
var4.set(Grid_y)
var5=tkinter.DoubleVar(tk)
var6=tkinter.DoubleVar(tk)
var_Fonte=tkinter.DoubleVar(tk)
var_Fonte.set(9)
caixa_Fonte=tkinter.Entry(fonte_Frame,textvar=var_Fonte)
Caixa1=tkinter.Entry(entrys_material,textvar=var1)
Caixa2=tkinter.Entry(entrys_material,textvar=var2)
Caixa3=tkinter.Entry(entrys_grid,textvar=var3)
Caixa4=tkinter.Entry(entrys_grid,textvar=var4)
Caixa5=tkinter.Entry(entrys_carregamento,textvar=var5)
Caixa6=tkinter.Entry(entrys_carregamento,textvar=var6)
bt_grid=tkinter.Button(tools,text='Aplicar Grid',command=aplicar_grid)
bt_material=tkinter.Button(tools,text='Aplicar Material',command=aplicar_material)
bt_draw=tkinter.Button(bts_modo,text='Desenhar',command=desenhar)
bt_select=tkinter.Button(bts_modo,text='Selecionar',command=selecionar)
bts_apoios=tkinter.Frame(tools,width=400)
bt_apoio1=tkinter.Button(bts_apoios,text='Inserir Apoio',command=apoioP)
bt_apoioRemove=tkinter.Button(bts_apoios,text='Remover',command=Remover_apoio)
bt1_carregamento=tkinter.Button(bts_carregamento,text='Inserir
Força',command=carregamento)
bt2_carregamento=tkinter.Button(bts_carregamento,text='Remover',command=remover_c
arregamento)
bts_main=tkinter.Frame(tools,width=400)
bt_calcular=tkinter.Button(bts_main,text='Calcular',command=Calcular,width=8,height=
2)
bt_resetar=tkinter.Button(bts_main,text='Resetar',command=Resetar,width=8,height=2
)
var_etapas=tkinter.IntVar(tk)
var_etapas.set(1)
caixa_etapas=tkinter.Entry(tools,textvar=var_etapas)
var_x=tkinter.BooleanVar(tk)
var_y=tkinter.BooleanVar(tk)
checks_apoio=tkinter.Frame(tools,width=400)
check_x=tkinter.Checkbutton(checks_apoio,text='X', variable=var_x,font='Times 12')
check_y=tkinter.Checkbutton(checks_apoio,text='Y', variable=var_y,font='Times 12')
canvas = tkinter.Canvas(f2,background='white',scrollregion=(0,0,2000,2000))
hbar=tkinter.Scrollbar(f2,orient='horizontal')
vbar=tkinter.Scrollbar(f2,orient='vertical')
hbar.config(command=canvas.xview)
vbar.config(command=canvas.yview)
hbar.pack(side='bottom',fill='x')
vbar.pack(side='right',fill='y')
CreateGrid()
```

```
620  tk.bind('<Escape>',Cancelar)
621  tk.bind('<Delete>',Deletar)
622  canvas.bind('<Button-1>',PointClick)
623  canvas.bind('<Motion>',Movimento)
624  canvas.config(width=2000,height=2000)
625  canvas.config(xscrollcommand=hbar.set, yscrollcommand=vbar.set)
626  canvas.pack(side='right',fill='both',expand=True)
627
628  ########## Declaração da Posição Onde cada Elemento Gráfico Ficará
##########
629  tools.pack(side='left',fill='y',expand=True)
630  lb_modo.pack()
631  bts_modo.pack()
632  lb_apoio.pack()
633  checks_apoio.pack()
634  check_x.pack(side='left')
635  check_y.pack(side='left')
636  bts_apoios.pack()
637  bt_apoio1.pack(side='left')
638  bt_apoioRemove.pack(side='left')
639  lb_material.pack()
640  propriedades_material.pack()
641  bt_material.pack()
642  lb_grid.pack()
643  propriedades_grid.pack()
644  bt_grid.pack()
645  lb_carregamento.pack()
646  propriedades_carregamento.pack()
647  labels_material.pack(side='left')
648  entrys_material.pack(side='left')
649  labels_grid.pack(side='left')
650  entrys_grid.pack(side='left')
651  labels_carregamento.pack(side='left')
652  entrys_carregamento.pack(side='left')
653  bts_carregamento.pack()
654  bt1_carregamento.pack(side='left')
655  bt2_carregamento.pack(side='left')
656  bt_draw.pack(side='left')
657  bt_select.pack(side='left')
658  fonte_Frame.pack()
659  lb_executar.pack()
660  caixa_etapas.pack()
661  bts_main.pack()
662  bt_calcular.pack(side='left')
663  bt_resetar.pack(side='left')
664  label1.pack()
665  label2.pack()
666  label3.pack()
667  label4.pack()
668  label5.pack()
669  label6.pack()
670  Caixa1.pack()
671  Caixa2.pack()
```

```
672 Caixa3.pack()
673 Caixa4.pack()
674 Caixa5.pack()
675 Caixa6.pack()
676 f.pack(fill='both',expand=True)
677 f2.pack(fill='both',expand=True)
678 fonte_Texto.pack()
679 caixa_Fonte.pack()
680 tools.configure(width=400)
681
682 ########## Configuração para Maximizar a Janela ao Iniciar ##########
683 tk.state('zoomed')
684
685 ########## Loop Principal do Programa ##########
686 tk.mainloop()
```

APÊNDICE B – CÓDIGO FONTE AUXILIAR (elem_finitos.py)

O código auxiliar é o responsável pela implementação do Método dos Elementos Finitos, ele trabalha com matrizes e faz uso da equação (3.38) para a correta implementação do método.

```
1  ########## Importanto Bibliotecas Necessárias ##########
2  import numpy
3
4  ########## Função para Definir uma Matriz Vazia ##########
5  def zeros(linha,coluna):
6      matriz=[]
7      for i in range(linha):
8          matriz.append([])
9          for j in range(coluna):
10             matriz[i].append(0)
11     return matriz
12
13  ########## Função Módulo para se Obter o Comprimento das Barras
##########
14  modulo=lambda no1,no2: ((no2[0]-no1[0])**2+(no1[1]-no2[1])**2)**(1/2)
15
16  ########## Função Principal Responsável pelo Calculo das Flechas Utilizando o
MEF ##########
17  def principal(coord,conect,E,list_A,travados,Forcas):
18      global matriz,M_forcas,inversa
19      nn=len(coord)
20      nel=len(conect)
21      ngl=2*nn
22      kg=zeros(ngl,ngl)
23      for i in conect:
24          no1=coord[i[0]-1]
25          no2=coord[i[1]-1]
26          L=modulo(no1,no2)
27          cos=(no2[0]-no1[0])/L
28          sen=(no2[1]-no1[1])/L
29          Ke=[[cos**2,cos*sen,-(cos**2),-cos*sen],[cos*sen,sen**2,-cos*sen,-
(sen**2)],[-(cos**2),-cos*sen,cos**2,cos*sen],[-cos*sen,-(sen**2),cos*sen,sen**2]]
30          Ke=[[E*list_A[conect.index(i)]/L*q for q in p] for p in Ke]
31          m1=(2*i[0]-1)-1
32          m2=(2*i[0])-1
33          m3=(2*i[1]-1)-1
34          m4=(2*i[1])-1
35          lista=[m1,m2,m3,m4]
36          for j in lista:
37              for k in lista:
38                  kg[j][k]+=Ke[lista.index(j)][lista.index(k)]
39      matriz=numpy.array(kg)
40      matriz=numpy.delete(numpy.delete(matriz,travados,0),travados,1)
```

```
41      inversa=numpy.linalg.inv(matriz)
42      M_forcas=numpy.delete(numpy.array(Forcas),travados,0)
43      M_forcas=M_forcas*-1
44      flexa=numpy.matmul(inversa,M_forcas)
45      return flexa
```