UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

VINICIUS MOURA DE OLIVEIRA

ESTUDO E DESENVOLVIMENTO DE CÓDIGO COMPUTACIONAL PARA ANÁLISE DE CHAPAS UTILIZANDO ELEMENTOS FINITOS CST E CSQ

CAMPO MOURÃO

2019

VINICIUS MOURA DE OLIVERA

ESTUDO E DESENVOLVIMENTO DE CÓDIGO COMPUTACIONAL PARA ANÁLISE DE CHAPAS UTILIZANDO ELEMENTOS FINITOS CST E CSQ

Trabalho de Conclusão de Curso de Graduação apresentado à Disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior em Engenharia Civil do Departamento Acadêmico de Construção Civil – DACOC - da Universidade Tecnológica Federal do Paraná - UTFPR, para obtenção do título de bacharel em engenharia civil.

Orientador: Prof. Dr. Leandro Waidemam

CAMPO MOURÃO

2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Campo Mourão
Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Construção Civil
Coordenação de Engenharia Civil

TERMO DE APROVAÇÃO

Trabalho de Conclusão de Curso

ESTUDO E DESENVOLVIMENTO DE CÓDIGO COMPUTACIONAL PARA ANÁLISE DE CHAPAS UTILIZANDO ELEMENTOS FINITOS CST E CSQ

por

Vinicius Moura de Oliveira

Este Trabalho de Conclusão de Curso foi apresentado às 09h00min do dia 19 de junho de 2019 como requisito parcial para a obtenção do título de ENGENHEIRO CIVIL, pela Universidade Tecnológica Federal do Paraná. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Dr. Marcelo Rodrigo Carreira
(UTFPR)

Prof. Dr. Jeferson Rafael Bueno
(UTFPR)

Prof. Dr. Leandro Waidemam
(UTFPR)
Orientador

Responsável pelo TCC: **Prof. Me. Valdomiro Lubachevski Kurta**

Coordenador do Curso de Engenharia Civil:
Prof. Dr(a). Paula Cristina de Souza

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

AGRADECIMENTOS

Primeiramente agradeço a Deus, por sempre abençoar meu caminho, iluminar meus pensamentos e me proteger. Por me proporcionar grandes conquistas e me dar forças todo momento ao longo de minha vida.

Aos meus pais Maria Celina de Moura de Oliveira e Claudenir de Oliveira, por todo sustento, educação e, principalmente, o amor que sempre me deram. Por me mostrarem como levar uma vida digna e com humildade, sempre deixando claro o valor que nossa família tem.

Ao meu orientador Professor Dr. Leandro Waidemam, por todos os ensinamentos, materiais, correções e atenção que direcionou a mim. Agradeço pela amizade e por ter me aceito para desenvolver este trabalho.

A todos que me ajudaram neste trabalho, agradeço aos meus parceiros de projeto Fernando Freire e Leandro Vieira, e em especial a meu amigo Lucas Guedes, por ter me ajudado com o desenvolvimento do *software*, passando todo seu conhecimento em programação e mostrando onde havia erros no código computacional. Também agradeço a Giovane Avancini pela ajuda com as análises utilizando o *software* ANSYS.

Aos muitos amigos que tive o prazer de conviver ao longo destes anos e sempre estiveram ao meu lado tanto nos bons quanto nos maus momentos.

A todos professores que contribuíram com minha formação acadêmica.

A minha família por todo o apoio, união e apreço por mim.

RESUMO

Este trabalho tem como objetivo apresentar a formulação matemática e desenvolver um código computacional com base no Método dos Elementos Finitos para análise elástica de chapas utilizando os elementos finitos *Constant Strain Triangle* (*CST*) e *Constant Strain Quadrilateral* (*CSQ*). Estes elementos foram utilizados para discretizar a estrutura e obter as equações algébricas para a solução do problema proposto. Ambos os elementos possuem dois graus de liberdade por nó, sendo eles os deslocamentos no plano, na direção dos eixos coordenados. O elemento *CST* possui geometria triangular e é definido por 3 nós localizados em seus vértices, sendo o campo de deslocamentos em seu interior do elemento aproximado por meio de um polinômio de primeiro grau. A localização dos nós é análoga para o elemento *CSQ*, ou seja, nós em cada um de seus 4 vértices. Possui geometria retangular e campo de deslocamentos aproximado por um polinômio de segundo grau. A formulação da matriz de rigidez dos elementos finitos foi obtida a partir do Princípio dos Trabalhos Virtuais e processo de integração analítica. Com a finalidade de validar a formulação apresentada, foi desenvolvido um código computacional em linguagem de programação *Python* direcionado a comunidade acadêmica, capaz de simular o comportamento estrutural de chapas com geometria, carregamentos e condições de vinculação diversas. Por fim, este trabalho apresenta três exemplos de diferentes casos de chapas, sendo os resultados obtidos comparados com os fornecidos por outros autores e *softwares* específicos da área.

Palavras-chave: Método dos Elementos Finitos. Análise elástica de chapas. Elementos *CST* e *CSQ*.

ABSTRACT

This work aims to present the mathematical formulation and develop a computational code based on the Finite Element Method for the elastic analysis of plates using Constant Strain Triangle (CST) and Quadrilateral Constant Strain (CSQ) elements. These elements were used to discretize the structure and to obtain the algebraic equations for the solution of the proposed problem. Both elements have two degrees of freedom per node, being they the displacements in the plane, in the direction of the coordinated axes. The CST element has triangular geometry and is defined by 3 nodes located at the vertices, the field of displacements within the approximate element by means of a first-degree polynomial. The location of the nodes are analogous to the CSQ element, that is, nodes in each of its 4 vertices. It has rectangular geometry and displacement field approximated by a second-degree polynomial. The formulation of the stiffness matrix of the finite elements was obtained from the Principle of Virtual Works and the process of analytical integration. In order to validate the presented formulation, a computational code was developed in Python programming language directed to the academic community, able to simulate the structural behavior of plates with geometry, loads and diverse bonding conditions. Finally, this paper presents three examples of different cases of plates, the results obtained being compared with those provided by other authors and specific software in the area.

Keywords: Finite Element Method. Elastic analysis of plates. Elements CST e CSQ.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE SÍMBOLOS E SIGLAS

CST	Constant Strain Triangle
CSQ	Constant Strain Quadrangular
MEF	Método dos Elementos Finitos
PTV	Princípio dos Trabalhos Virtuais
U_e^*	Trabalho virtual externo
U_i^*	Trabalho virtual interno
σ	Tensão normal
τ	Tensão de cisalhamento
θ_p	Ângulo das direções perpendiculares ao plano das tensões principais
θ_s máxima	Ângulo das direções perpendiculares ao plano da tensão de cisalhamento
u	Deslocamento nodal - na direção direção x
v	Deslocamento nodal - na direção direção y
ξ	Coordenadas baricêntricas do elementos *CSQ* – na direção do eixo x
η	Coordenadas baricêntricas do elementos *CSQ* – na direção do eixo y
ε	Deformação normal
γ	Deformação angular
υ	Coeficiente de Poisson
E	Módulo de elasticidade longitudinal
G	Módulo de elasticidade transversal
[D]	Matriz de elasticidade
[B]	Matriz deformação
{F}	Vetor de forças nodais
$\{u\}^{T*}$	Vetor transposto de deslocamento virtual externo;
$\{\varepsilon\}^{T*}$	Vetor transposto de deformações virtuais internas;
[k]	Matriz de rigidez do elemento do elemento finito

$\{u\}$	Vetor de deslocamentos nodais
$[N]$	Matriz das funções de forma
$\{\sigma\}$	Vetor de tensões
$\{\varepsilon\}$	Vetor de deformações
$[k]_{global}$	Matriz de rigidez global da estrutura

SUMÁRIO

1 INTRODUÇÃO

As problemáticas de análise estrutural, desde os primórdios da engenharia civil, apresentam grandes desafios no campo da Engenharia de Estruturas. Segundo Sussekind (1981) a análise estrutural é a parte da Mecânica que estuda as estruturas, consistindo este estudo na determinação dos esforços e das deformações a que elas ficam submetidas quando solicitadas por agentes externos (cargas, variações térmicas, movimento de seus apoios, etc.).

A grande maioria das estruturas é composta de diferentes elementos estruturais interligados entre si por ligações que podem ser contínuas ou discretas, de modo a formar um conjunto resistente. Esse conjunto de elementos devem possuir capacidade de receber e transmitir esforços até às fundações.

Com o objetivo de resolver problemas complexos de análise estrutural, é habitual recorrer à métodos numéricos aproximados, uma vez que estes problemas são representados por equações diferenciais parciais com condições de contorno definidas. Normalmente, na engenharia civil, um problema complexo é resolvido fragmentando-o em vários problemas mais simples, facilitando a solução do problema.

De acordo com o exposto, o método numérico mais utilizado para a resolução de tais problemas é o Método dos Elementos Finitos (MEF). Conforme Azevedo (2003) com o grande desenvolvimento que o MEF teve na década de 1960 e com a popularização dos recursos computacionais, passou a ser prática e corrente a análise de estruturas de geometria arbitrária, constituídas por múltiplos materiais e sujeitas a qualquer tipo de carregamento. Este avanço é tão significativo que outros métodos, como o método das diferenças finitas, por exemplo, deixaram praticamente de ser utilizados.

O MEF consiste na discretização de um sistema contínuo em vários elementos com finitos graus de liberdade e geometria simples, chamados de elementos finitos. Através desse método é possível obter resultados aproximados, no qual sua precisão depende do número de nós e elementos.

Frequentemente utilizados em estruturas metálicas e em ligações de estruturas de madeira, os elementos de chapa também possuem aplicabilidade em estruturas de concreto armado, no caso de viga-parede e pilar-parede.

De acordo com esse contexto, este trabalho tem como objetivo principal apresentar a formulação matemática e desenvolver um código computacional embasado no MEF para análise elástica linear de chapas utilizando os elementos finitos *Constant Strain Triangle* (*CST*) e *Constant Strain Quadrilateral* (*CSQ*).

1.1 Objetivos

1.1.1 Objetivo Geral

Apresentar a formulação matemática com desenvolvimento de código computacional com base no Método dos Elementos Finitos (MEF) para análise elástica linear de chapas utilizando os elementos *CST* e *CSQ*.

1.1.2 Objetivos Específicos

- Desenvolver teórica e numericamente um modelo numérico fundamentado no Método dos Elementos Finitos aplicado ao estudo de chapas;
- Implementar, em linguagem Python, um programa computacional que incorpore tal modelo e contemple a análise elástica linear de chapas com geometria, carregamentos e condições de vinculação diversas;
- Processar alguns exemplos de chapas no *software* desenvolvido;
- Verificar a precisão dos resultados obtidos, bem como a coerência da implementação computacional, por meio de resultados fornecidos por outros autores e por *softwares* específicos da área.

1.2 Justificativa

As chapas são elementos estruturais comumente utilizados em estruturas metálicas, ligações de estruturas de madeira e nas estruturas de concreto armado no caso de vigas-parede e pilar-parede. As vigas metálicas de seção I são elementos lineares formadas por 3 chapas de aço (uma alma e duas mesas) que, devido a esbeltez local, necessitam de avaliação quanto à possibilidade de instabilidade local. Essas chapas que compõe as vigas de seção I proporcionam maior rigidez, menor peso próprio do sistema estrutural e economia de seção transversal quando comparadas às vigas de concreto armado.

Os elementos de chapa são elementos definidos como laminares, e sofrem solicitações no próprio plano, sendo mais comum o carregamento uniforme ao longo da sua espessura. Estes elementos estão sujeitos à problemas de elasticidade plana, que segundo Ribeiro (2004), se

dividem em dois grupos: problemas de estado plano de tensão e problemas de estado plano de deformação.

Em meio a diversos métodos numéricos utilizados para solucionar problemas de Engenharia de Estruturas, no presente trabalho, optou-se por adotar o Método dos Elementos Finitos. O motivo de tal escolha justifica-se por ser este um método que fornece soluções precisas e ser de fácil implementação computacional.

Os elementos finitos que serão utilizados nesse trabalho são os elementos *CST* e *CSQ*, ambos os elementos possuem dois graus de liberdade por nó, sendo eles os deslocamentos vertical e horizontal. O elemento *CST* possui geometria triangular e é definido por 3 nós localizados em seus vértices, sendo o campo de deslocamentos em seu interior do elemento aproximado por meio de um polinômio de primeiro grau. A localização dos nós é análoga para o elemento *CSQ* que possui geometria retangular e campo de deslocamentos aproximado por um polinômio de segundo grau.

O elemento *CST* é o elemento finito mais simples para a análise de chapas. Segundo Savassi (1996) dentre as formas de polígonos que podem ser adotados para realizar a discretização da chapa em elementos finitos a triangular apresenta interesse particular. Com ela torna-se possível aproximar qualquer contorno em forma curva. Já o elemento *CSQ*, por interpolar os deslocamentos com funções de graus maiores, proporciona melhores resultados que o elemento *CST* quando se utiliza o mesmo número de elementos na malha.

Por fim, é importante salientar a contribuição deste trabalho no âmbito acadêmico, uma vez que como produto final pretende-se apresentar um código computacional para análise elástica linear de chapas que estará disponível a docentes e discentes da universidade de modo a auxiliá-los em seus estudos e no desenvolvimento de pesquisas, bem como para uso didático em disciplinas da área de Engenharia de Estruturas.

1.3 Apresentação

Neste primeiro capítulo foi apresentada uma visão geral sobre as chapas como elementos estruturais, assim como a idealização do MEF e a importância dos métodos numéricos no desenvolvimento de *softwares* aplicados.

No capítulo 2 é realizada uma revisão bibliográfica contendo um breve histórico do MEF e sua aplicabilidade para análise de elementos de chapa, seguido da dedução, a partir do PTV, das matrizes de rigidez dos elementos e dos vetores de forças nodais equivalentes.

No terceiro capítulo são apresentados todos os aspectos referentes à implementação computacional da formulação desenvolvida, contendo a descrição de cada sub-rotina de cálculo. O funcionamento geral do *software* desenvolvido é ilustrado por meio de um fluxograma contendo o esquema geral de cálculo.

No quarto capítulo são apresentados os resultados obtidos através de simulações realizadas com o *software* desenvolvido, sendo os resultados obtidos validados através da comparação com os fornecidos por outros autores e também provenientes da análise realizada com o auxílio do programa ANSYS.

Por fim, o quinto capítulo traz as considerações finais do trabalho e também algumas sugestões para trabalhos futuros. O código fonte do programa desenvolvido, o arquivo de entrada e o arquivo de saída são expostos sob a forma de apêndices.

2 REVISÃO BIBLIOGRÁFICA

2.1 Chapas

O sistema estrutural é definido como sendo uma estrutura física qualquer composta por uma série de elementos estruturais adequadamente associados, formando um conjunto resistente. Esses elementos são corpos sólidos elástico-deformáveis, com a capacidade de receber e transmitir esforços de uma forma geral (WAIDEMAM, 2004).

Segundo Azevedo (2004), os elementos estruturais podem ser classificados quanto à sua geometria como reticulados (lineares), laminares (superfície) ou sólidos (blocos). Estes últimos são os mais genéricos, sendo classificados como sólidos os que não apresentarem características que as permitam enquadrar no grupo dos laminares ou reticulados.

Os lineares têm duas dimensões pequenas em relação a uma terceira. Nessa categoria estão os pilares, os tirantes, as vigas os pórticos e arcos. Os laminares têm uma dimensão pequena em relação as outras duas. São elementos estruturais muito importantes, permitindo a obtenção de maior rigidez e maior economia em relação aos elementos lineares, se estes fossem utilizados para composição de soluções equivalentes. É o caso dos pisos de edifícios em que um elemento estrutural (monolítico) de superfície pode ser mais econômico do que um conjunto de vigas pré-moldadas apenas justapostas. Os elementos laminares podem, por sua vez, desempenhar funções de chapas, placas e cascas. As chapas são planas, com ações que tem que tem sua resultante contida em seu plano médio. As placas, também planas, mas recebem ações ortogonais ao plano médio. Por sua vez as cascas apresentam superfície media curva ou poligonal. Quanto aos blocos, nenhuma das dimensões predomina sobre as outras (SAVASSI, 1996).

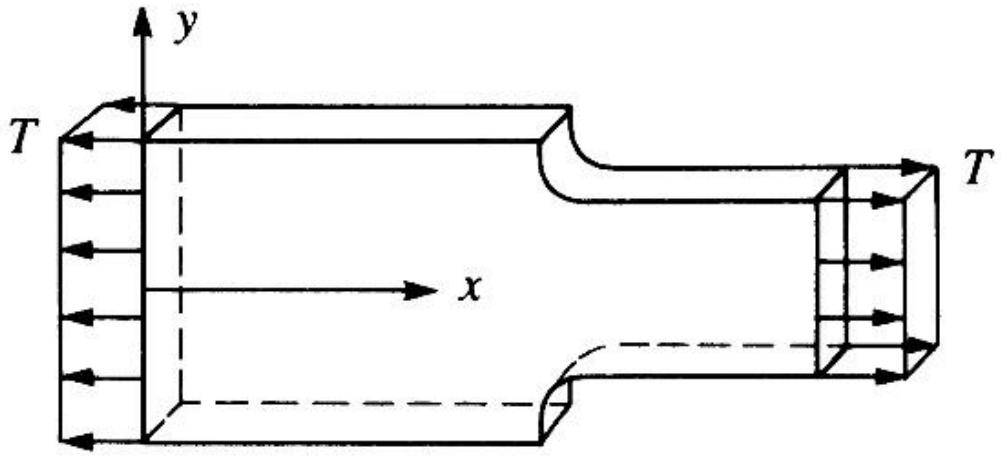
Waidemam (2004) explica que um sistema estrutural é contínuo desde que o menor elemento estrutural retirado do sistema possua as mesmas propriedades físicas e que não haja descontinuidade no conjunto. Em função dessa continuidade os sistemas estruturais contínuos possuem um número infinito de graus de liberdade, fato este que dificulta a resolução do problema.

2.1.1 Estado Plano de Tensão

O estado plano de tensões pode ocorrer em elementos estruturais chamados de superfície ou laminares, submetidos a ações que obedeçam simetria em relação ao plano médio. É o caso

típico das chapas (SAVASSI, 1996), ilustrado na Figura 1. Ainda sobre o estado plano de tensões, Logan (2007) o define como um estado de tensão no qual a tensão normal e as tensões de cisalhamento direcionadas perpendicularmente ao plano são assumidas como zero. Em outras palavras, o estado plano de tensão em um ponto é representado exclusivamente por três componentes que agem sobre um elemento que tenha uma orientação específica neste ponto. (HIBBELER, 2004).

Figura 1 - Problema de estado plano de tensão.



Fonte: Logan (2007).

Neste caso, o estado de tensões é completamente definido pelas componentes de tensões σ_x σ_y e τ_{xy}, constantes ao longo da espessura (RIBEIRO, 2004). Estas tensões podem ser representadas pelo vetor da equação (1):

$$\{\sigma\} = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} \tag{1}$$

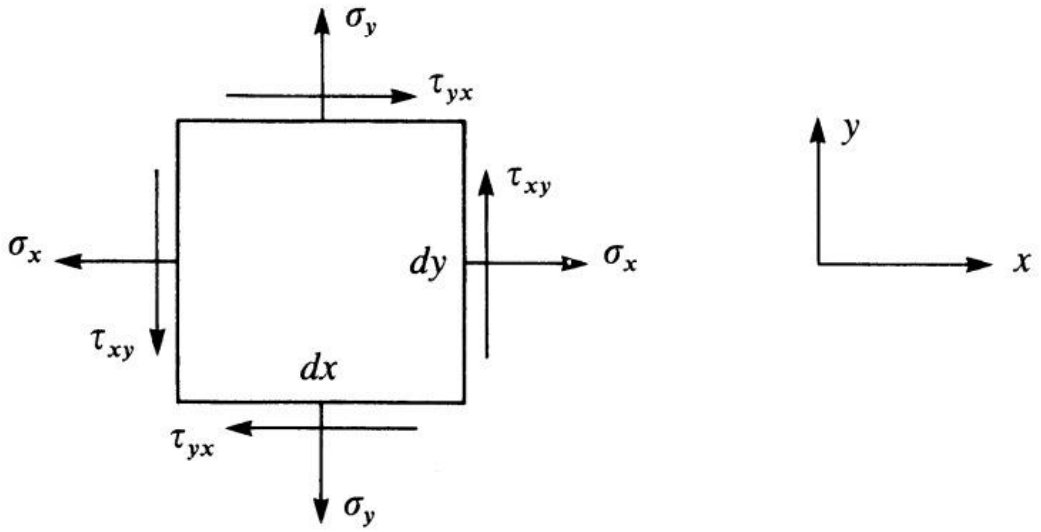
Para o estado plano de tensões ilustrado na Figura 2, é possível determinar as tensões principais e de cisalhamento máxima no plano. Essas tensões podem ser obtidas nas equações (2),(3) e (4).

$$\sigma_1 = \frac{\sigma_x + \sigma_y}{2} + \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \tau_{xy}^2} = \sigma_{máx} \tag{2}$$

$$\sigma_2 = \frac{\sigma_x + \sigma_y}{2} - \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \tau_{xy}^2} = \sigma_{\min} \tag{3}$$

$$\tau_{máx} = \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \tau_{xy}^2} \tag{4}$$

Figura 2 - Estado plano de tensão.



Fonte: Logan (2007).

Além disso, os ângulos principais θ_p e θ_s, que definem as direções principais e de cisalhamento máximo no plano, respectivamente, são definidos por:

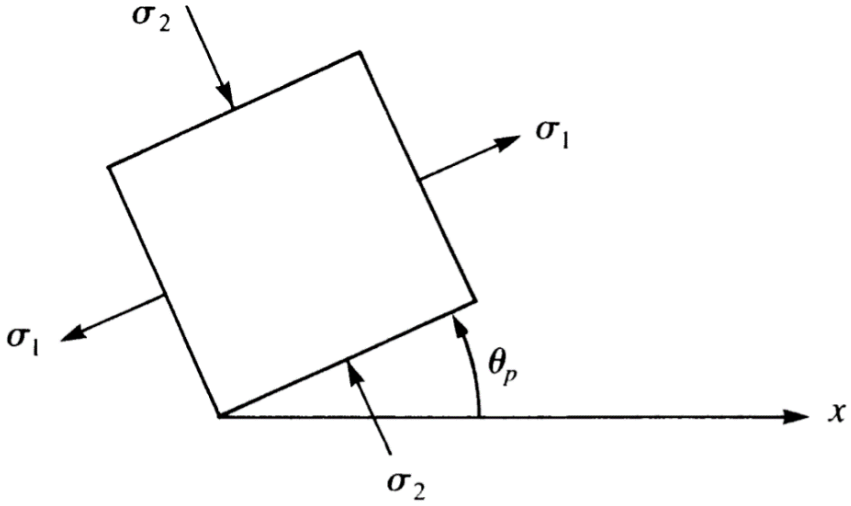
$$\tan 2\theta_p = \frac{2\tau_{xy}}{\sigma_x - \sigma_y} \tag{5}$$

$$\tan 2\theta_s = -\frac{\sigma_x - \sigma_y}{2\tau_{xy}} \tag{6}$$

A Figura 3 representa as tensões principais σ_1 e σ_2 e o ângulo θ_p. É importante lembrar que a tensão de cisalhamento é zero nos planos principais e também que há uma tensão normal nos planos onde ocorre a tensão de cisalhamento máxima. Essa tensão é chamada de tensão normal média e pode ser obtida através da equação (7). A Figura 4 representa a tensão normal média, a tensão de cisalhamento máxima no plano e o ângulo θ_s.

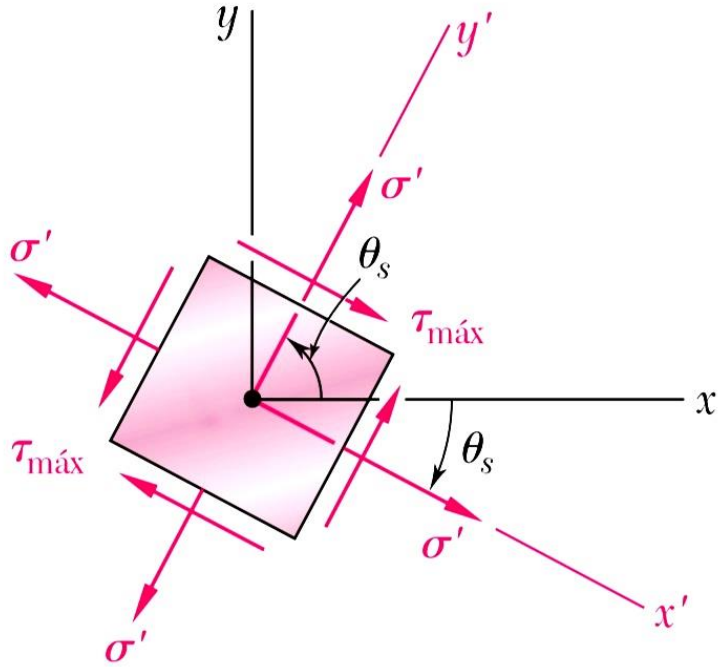
$$\sigma' = \sigma_{méd} = \frac{\sigma_x + \sigma_y}{2} \tag{7}$$

Figura 3 - Tensões principais e suas direções.



Fonte: Logan (2007).

Figura 4 - Tensão média e tensão de cisalhamento máxima e suas direções.



Fonte: Adaptado de Beer et al (2011).

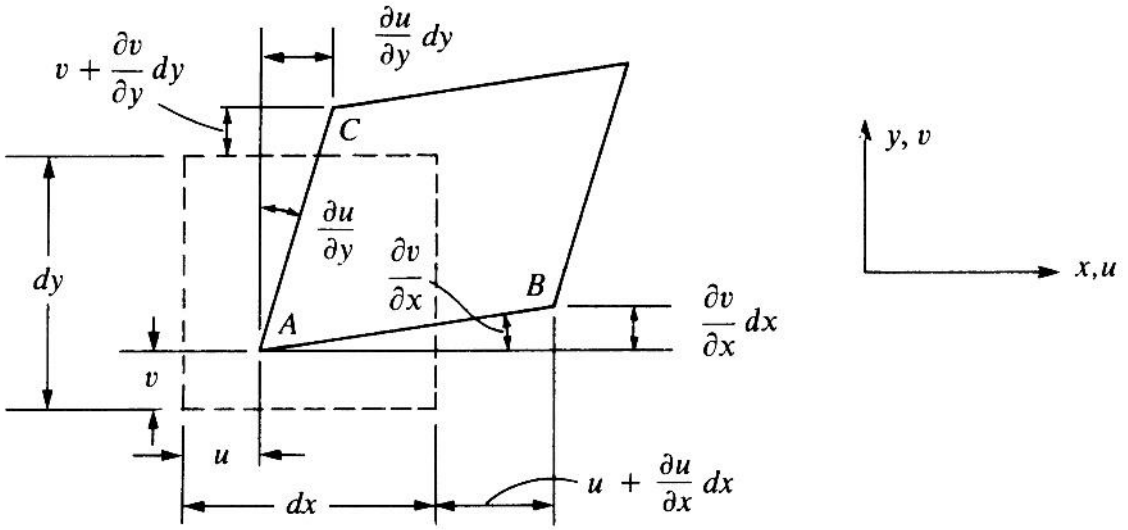
2.1.2 Estado Plano de Deformação

O estado plano de deformações pode ocorrer em elementos estruturais alongados. Neles basta considerar lâminas definidas por dois cortes paralelos e ortogonais ao eixo da estrutura.

Se essas lâminas não forem influenciadas pelas longínquas ações de extremidade de corpo alongando e se o carregamento não variar ao longo do comprimento da estrutura, pode-se dizer que o comportamento de uma lâmina é praticamente o mesmo que o de suas vizinhanças próximas (SAVASSI, 1996).

Na Figura 5, um elemento infinitesimal é utilizado para representar o estado plano de deformação em algum ponto de uma estrutura. O elemento é mostrado para ter deslocamentos u e v nas direções x e y do ponto A, e para deslocar ou estender uma quantidade adicional ($\partial u/\partial x$) dx ao longo da linha AB, e ($\partial v/\partial y$) dy ao longo da linha AC nas direções x e y, respectivamente. Além disso, observando as linhas AB e AC, verifica-se que o ponto B se move para cima em uma quantidade ($\partial v/\partial x$) dx em relação a A, e o ponto C se move para direita em uma quantidade ($\partial u/\partial y$) dy em relação a A (LOGAN, 2007).

Figura 5 - Deslocamentos e rotações de um elemento no plano x-y.



Fonte: Logan (2007).

Com base nas equações da teoria da elasticidade, tem-se que as deformações ε_x e ε_y são as mudanças no comprimento por unidade de comprimento de fibras dos materiais originalmente paralelas aos eixos x e y, respectivamente, quando o elemento sofre deformação. Estas deformações são então chamadas de deformações normais. A deformação γ_{xy} é a mudança no ângulo reto original entre dx e dy quando o elemento sofre deformação. A deformação γ_{xy} é então chamada deformação angular ou deformação por cisalhamento.

Das definições gerais das deformações normais e de cisalhamento obtém-se a equação (8):

$$\varepsilon_x = \frac{\partial u}{\partial x} \;\; ; \;\; \varepsilon_y = \frac{\partial v}{\partial y} \;\; ; \;\; \gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \tag{8}$$

As deformações dadas pelas equações (8), são geralmente representadas pelo vetor da equação (9):

$$\{\varepsilon\} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \tag{9}$$

2.1.3 Relação entre Tensão e Deformação

Se o material em um ponto estiver sujeito a um estado de tensão triaxial, σ_x, σ_y, σ_z, deformações normais associadas ε_x, ε_y, ε_z serão desenvolvidas no material. As tensões podem ser relacionadas com as deformações pelo princípio da superposição, coeficiente de Poisson, $\varepsilon_{lat} = -\upsilon . \varepsilon_{long}$ e pela lei de Hooke, como aplicável na direção uniaxial, $\varepsilon = \sigma/E$ (HIBELLER, 2004).

Para apresentar como isso é feito, primeiro deve-se considerar a deformação do elemento na direção x, causada pela aplicação isolada de cada tensão normal. Quando σ_x é aplicada, o elemento alonga-se na direção x e a deformação ε'_x nessa direção é dado pela equação (10):

$$\varepsilon'_x = \frac{\sigma_x}{E} \tag{10}$$

As aplicações de σ_y e σ_z provoca a contração do elemento com as deformações ε''_x e ε'''_x na direção x. Aqui, dado pelas equações (11) e (12):

$$\varepsilon''_x = -\nu \frac{\sigma_y}{E} \tag{11}$$

$$\varepsilon'''_x = -\nu \frac{\sigma_z}{E} \tag{12}$$

Quando essas três deformações normais são superpostas, a deformação normal ε_x é determinada para o estado de tensão. Equações semelhantes podem ser desenvolvidas para as deformações normais nas direções y e z (HIBELLER, 2004), que resulta em na equação (13):

$$\begin{aligned} \varepsilon_x &= \frac{1}{E}\left[\sigma_x - \nu\left(\sigma_y + \sigma_z\right)\right] \\ \varepsilon_y &= \frac{1}{E}\left[\sigma_y - \nu\left(\sigma_x + \sigma_z\right)\right] \\ \varepsilon_z &= \frac{1}{E}\left[\sigma_z - \nu\left(\sigma_x + \sigma_y\right)\right] \end{aligned} \tag{13}$$

Essas equações expressam a Lei de Hooke generalizada para um estado triaxial de tensão.

Ainda Hibeller (2004) relata que se for aplicada uma tensão de cisalhamento τ_{xy} ao elemento, observações experimentais indicam que o material será distorcido somente devido a uma deformação por cisalhamento γ_{xy}, isto é, τ_{xy} não causará outras deformações no material. Da mesma forma, τ_{yz} e τ_{xz} provocarão somente deformações por cisalhamento γ_{yz} e γ_{xz}, respectivamente. Portanto, a lei de Hooke para tensão de cisalhamento e deformação por cisalhamento pode ser escrita pela equação (14):

$$\gamma_{xy} = \frac{1}{G}\tau_{xy} \;\; ; \;\; \gamma_{yz} = \frac{1}{G}\tau_{yz} \;\; ; \;\; \gamma_{xz} = \frac{1}{G}\tau_{xz} \tag{14}$$

nas quais o módulo de elasticidade transversal *G* é dado na equação (15):

$$G = \frac{E}{2\left(1+\nu\right)} \tag{15}$$

Utilizando as equações (13) e (14), e considerando apenas as componentes de tensão não nulas para caso do estado plano de tensão, as relações de tensão-deformação podem ser expressas pela equação (16):

$$\sigma_x = \frac{E}{1-\nu^2}\left(\varepsilon_x + \nu\varepsilon_y\right)$$
$$\sigma_y = \frac{E}{1-\nu^2}\left(\varepsilon_y + \nu\varepsilon_x\right) \tag{16}$$
$$\tau_{xy} = \frac{E}{2(1+\nu)}\gamma_{xy} = G\gamma_{xy}$$

no qual E é o módulo de elasticidade e ν é o coeficiente de Poisson para o material. Na relação tensão-deformação de cisalhamento o módulo de elasticidade transversal G foi introduzido.

As relações de tensão-deformação dadas pela equação (16) podem ser convenientemente escritas em forma matricial pela equação (17):

$$\begin{Bmatrix}\sigma_x\\ \sigma_y\\ \tau_{xy}\end{Bmatrix} = \frac{E}{1-\nu^2}\begin{bmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & \dfrac{1-\nu}{2}\end{bmatrix}\begin{Bmatrix}\varepsilon_x\\ \varepsilon_y\\ \gamma_{xy}\end{Bmatrix} \tag{17}$$

que de forma abreviada pode ser representada pela equação (18),

$$\{\sigma\} = [D]\{\varepsilon\} \tag{18}$$

no qual a matriz de elasticidade [D] para o estado plano de tensão, é dada pela equação (19):

$$[D] = \frac{E}{1-\nu^2}\begin{bmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & \dfrac{1-\nu}{2}\end{bmatrix} \tag{19}$$

2.2 Método dos Elementos Finitos

O método dos elementos finitos (MEF), é uma técnica computacional usada para obter soluções aproximadas de problemas de valor limite em engenharia. Em termos simples, um problema de valor limite é um problema matemático no qual uma ou mais variáveis dependentes, devem satisfazer uma equação diferencial em qualquer ponto dentro de um domínio conhecido de variáveis independentes, no qual deve-se satisfazer condições específicas

no limite do domínio (HUTTON, 2004). Segundo Logan (2007) o método dos elementos finitos é um método numérico utilizado para resolver problemas de engenharia. As típicas áreas problemáticas de interesse em engenharia e física matemática que são solucionáveis pelo método dos elementos finitos incluem: análise estrutural, transferência de calor, fluxo de fluido, transporte de massa e potencial eletromagnético.

Diversos problemas com importância para a engenharia podem ser descritos em termos de equações com derivadas parciais. Com exceção de alguns casos particulares, não é possível obter uma solução analítica exata para estes problemas. O método dos elementos finitos é, atualmente, o método numérico mais utilizado para obter soluções aproximadas para este tipo de problemas (PEREIRA, 2004). Ainda sobre a problemática da impossibilidade de obter-se uma solução analítica exata em alguns casos particulares, Savassi (1996) acrescenta que de modo geral há necessidade de admitir certas hipóteses simplificadoras, que tornem tratáveis as estruturas ou elementos estruturais que se pretenda (projetar) calcular (e construir). Essas simplificações visam permitir a obtenção de um modelo idealizado da estrutura, para o qual seja possível estabelecer o equacionamento do problema em foco, de maneira a poder obter sua resposta (deslocamentos generalizados, tensões, deformações, etc.) quando sob agentes diversos (forças generalizadas, gradientes térmicos, etc.).

Soluções aproximadas baseadas em técnicas numéricas e computação digital são obtidas com maior frequência em análise de engenharia de problemas complexos. A análise por elementos finitos é uma técnica poderosa para obter tais soluções aproximadas com boa precisão (HUTTON, 2004). Para Logan (2007) estas soluções analíticas geralmente requerem a solução de equações diferenciais ordinárias ou parciais, que, devido às geometrias complicadas, cargas e propriedades do material, não são normalmente obtidas. Por isso, são necessários os métodos numéricos, para se obter soluções aceitáveis.

Para deixar claro os princípios básicos do Método dos Elementos Finitos, Fish (2007) explica que a ideia por de trás deste método é a divisão ou discretização de um corpo em um número finito de partes, chamada de elementos finitos, que são interligados por nós. A este conjunto formado pelos elementos finitos dá-se o nome de malha, que por sua vez está diretamente relacionado com a exatidão da solução. Segundo Souza (2003) a precisão do método depende da quantidade de nós e elementos, e do tamanho e tipo dos elementos presentes na malha.

Um dos aspectos mais importantes do MEF diz respeito a sua convergência. Embora trata-se de um método aproximado, pode-se demonstrar que em uma malha consistente, a medida que o tamanho dos elementos finitos tende a zero, e consequentemente, a quantidade

de nós tende a infinito, a solução obtida converge para a solução exata do problema (SOUZA, 2003).

A formulação do MEF pode ser baseada no método dos deslocamentos, em modelos de equilíbrio, ou em métodos híbridos e mistos. De todos estes métodos, aquele que apresenta uma maior simplicidade e, consequentemente, uma maior versatilidade é o método dos deslocamentos (AZEVEDO, 2003).

> A estrutura é subdividida em partes denominadas elementos finitos, em cujos domínios admite-se conhecida a forma das funções incógnitas. A denominação "elemento finito" foi escolhida para destacar tratamento diferente daquele considerado na formulação local, feita sobre elemento infinitesimal. Com a utilização de procedimentos diretos da mecânica (no caso de elementos estruturais muito simples) ou com uso de formulações variacionais (baseadas em energia) ou, ainda, de resíduos ponderados, obtém-se o relacionamento aproximado entre deslocamentos de certos pontos do elemento (nós) e as respectivas forças (nodais). Para tratar a estrutura toda bastará providenciar a montagem dos elementos e considerar as respectivas equações algébricas que representam o relacionamento entre os deslocamentos e as forças nodais. (SAVASSI, 1996, p. 3)

2.3 Método dos Elementos Finitos Aplicado à Análise de Chapas

2.3.1 Princípio dos Trabalhos Virtuais (PTV)

O Princípio dos Trabalhos Virtuais (PTV) estabelece que o trabalho realizado pelas forças internas na deformação virtual do corpo é igual ao trabalho realizado pelas forças externas nos deslocamentos virtuais dos seus pontos de aplicação (AZEVEDO, 2003).

Segundo Gesualdo (2010) a formulação do PTV é baseada na afirmação de que *"para um sistema em equilíbrio ao qual se impõe deformações e deslocamentos compatíveis com suas vinculações, pode-se afirmar que o trabalho virtual produzido pelas forças externas nos deslocamentos é igual ao trabalho interno das tensões nas deformações virtuais internas"*. Este princípio é representado na equação (20):

$$W_e^* = W_i^* \tag{20}$$

A aplicação do PTV significa impor uma variação de deformações e deslocamentos virtuais, implicando, assim, em um trabalho virtual externo gerado por uma variação de deslocamentos u_i e em um trabalho virtual interno gerado por uma variação de deformação ε (GESUALDO, 2010).

Portanto, ao se aplicar o PTV, na forma vetorial, obtém-se:

$$\{u\}^{T*} \cdot \{F\} = \int_V \{\varepsilon\}^{T*} \cdot \{\sigma\} dV \tag{21}$$

No qual,

{F}: vetor de forças nodais reais do elemento;

$\{u\}^{T*}$: vetor transposto de deslocamento virtual externo;

$\{\varepsilon\}^{T*}$: vetor transposto de deformações virtuais internas;

$\{\sigma\}$: vetor de tensões reais.

Para interpolar os deslocamentos no interior dos elementos com os deslocamentos nodais dos mesmos, pode-se utilizar funções de aproximação chamadas de funções de forma, que em forma de matriz simplificada podem ser representadas por:

$$\{u\} = [N] \cdot \{u^e\} \tag{22}$$

onde [N] é a matriz das funções de forma.

O vetor de deformações do elemento bidimensional foi apresentado na equação (9). Substituindo a equação (9) na equação (22) e utilizando uma matriz [B] que seja o produto entre os operadores diferenciais e a matriz das funções de forma, tem-se que:

$$\{\varepsilon\} = [B].\{u\} \tag{23}$$

Da mesma forma, quando se trabalha com valores virtuais, é possível escrever que:

$$\{\varepsilon\}^{*} = [B].\{u\}^{*} \tag{24}$$

Utilizando as definições de matriz transposta é possível reescrever a equação (24) como:

$$\{\varepsilon\}^{T*} = \{u\}^{T*}.[B]^{T} \tag{25}$$

Substituindo a equação (23) na relação entre tensão e deformação dada pela equação (18), obtém-se:

$$\{\sigma\} = [D] \cdot [B] \cdot \{u\} \tag{26}$$

Substituindo as equações (25) e (26) na equação (21), obtém-se:

$$\{u\}^{T*} \{F\} = \int_V \{u\}^{T*} . [B]^T . [D] . [B] . \{u\} \, dV \tag{27}$$

Como o deslocamento virtual $\{u^e\}^{T*}$ é uma constante (valores não nulos de deslocamentos impostos nas nós), este pode ser colocado fora da integral. Consequentemente, será cancelado com o mesmo valor que aparece no primeiro membro. Assim:

$$\{F\} = \left(\int_V [B]^T \cdot [D] \cdot [B] \cdot dV \right) \cdot \{u\} \tag{28}$$

que escrita de forma abreviada pela equação (29) obtêm-se:

$$\{F\} = [k] . \{u\} \tag{29}$$

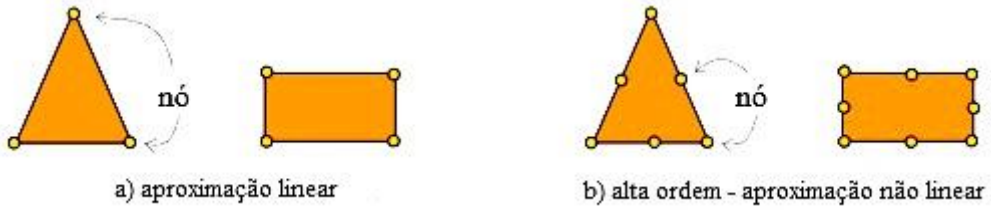
no qual a matriz de rigidez do elemento $[k^e]$ é dada pela equação (30):

$$[k^e] = \int_V [B]^T . [D] . [B] . dV \tag{30}$$

2.3.2 Elemento Triangular *CST* (*Constant Strain Triangle*)

Para o caso de modelos bidimensionais, basicamente, empregam-se elementos do tipo triangular e retangular. Estes devem ter no mínimo 3 e 4 nós, respectivamente, posicionados em cada um de seus vértices. Também é possível o uso de elementos com nós adicionais ao longo de suas arestas, tendo-se, então, os chamados elementos de alta ordem, mais complexos e que permitem representações polinomiais quadráticas. (GESUALDO, 2010)

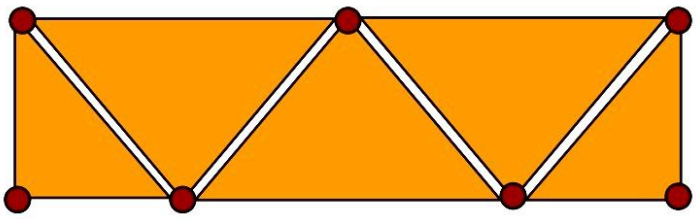
Figura 6 - Elementos e quantidades de nós.



Fonte: Adaptado de Gesualdo (2010).

Ainda utilizando as definições dadas por Gesualdo (2010), o modelo completo corresponde à associação dos diversos elementos conectados pelos nós. No processo de cálculo, apenas são garantidos que os deslocamentos associados aos graus de liberdade dos nós de conectividade são iguais. Assim, ao longo dos lados dos elementos as deformações são especificas para cada elemento e dependem do seu comportamento interno.

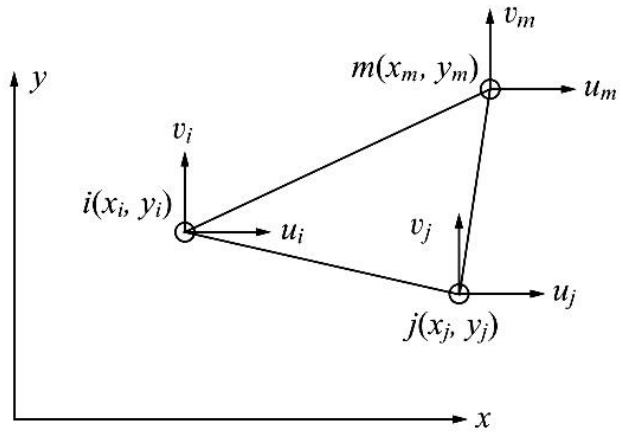
Figura 7 - Conexão entre os elementos - compatibilidade de deslocamentos apenas nos nós.



Fonte: Gesualdo (2010).

Segundo Logan (2007) para analisar o elemento de chapa, considera-se o elemento triangular ilustrado na Figura 8. A chapa discretizada foi dividida em elementos triangulares, cada um com 3 nós (i, j e m). Os elementos triangulares são particularmente interessantes quando os corpos estudados possuem limites com formas irregulares. Além disso, as expressões relacionadas ao elemento triangular são relativamente simples quando comparadas aos demais tipos de elementos. No elemento apresentado, cada nó tem dois graus de liberdade – deslocamentos em x e em y. Por simplicidade, considera-se que u_i e v_i representem os componentes de deslocamento do nó i nas direções x e y, respectivamente. Na Figura 8 tem-se ainda que (x_i, y_i), (x_j, y_j) e (x_m, y_m) são coordenadas nodais conhecidas dos nós i, j, e m, respectivamente.

Figura 8 - Elemento triangular básico mostrando seus graus de liberdade.



Fonte: Logan (2007).

Todas as formulações são baseadas no sistema anti-horário de rotulagem de nós, embora uma formulação baseada em um sistema de rotulagem no sentido horário possa ser usada. É importante ressaltar que um processo de rotulagem consistente para todo o corpo é necessário para evitar problemas nos cálculos, como áreas de elementos negativos (LOGAN, 2007).

O vetor de deslocamentos nodais é dado por,

$$\{d\} = \begin{Bmatrix} d_i \\ d_j \\ d_m \end{Bmatrix} = \begin{Bmatrix} u_i \\ v_i \\ u_j \\ v_j \\ u_m \\ v_m \end{Bmatrix} \tag{31}$$

Seleciona-se uma função de deslocamento linear para cada deslocamento como,

$$\begin{aligned} u(x,y) &= a_1 + a_2 x + a_3 y \\ v(x,y) &= a_4 + a_5 x + a_6 y \end{aligned} \tag{32}$$

onde u(x, y) e v(x, y) descrevem deslocamentos em qualquer ponto interior (x_i, y_i) do elemento.

A função linear garante que a compatibilidade seja satisfeita. Uma função linear com pontos finais específicos tem apenas um caminho pelo qual passar, isto é, através dos dois pontos. Portanto, a função linear garante que os deslocamentos ao longo da borda e nos nós

compartilhados por elementos adjacentes sejam iguais. Usando a equação (32) a função de deslocamento geral {ψ}, que contém as funções u e v, pode ser expressa como,

$$\{\psi\} = \begin{Bmatrix} a_1 + a_2 x + a_3 y \\ a_4 + a_5 x + a_6 y \end{Bmatrix} = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{Bmatrix} \tag{33}$$

As constantes *a* nas equações (32), são obtidas substituindo as coordenadas dos pontos nodais nas equações (32). Assim:

$$\begin{aligned} u_i &= u(x_i, y_i) = a_1 + a_2 x_i + a_3 y_i \\ u_j &= u(x_j, y_j) = a_1 + a_2 x_j + a_3 y_j \\ u_m &= u(x_m, y_m) = a_1 + a_2 x_m + a_3 y_m \\ v_i &= v(x_i, y_i) = a_4 + a_5 x_i + a_6 y_i \\ v_j &= v(x_j, y_j) = a_4 + a_5 x_j + a_6 y_j \\ v_m &= v(x_m, y_m) = a_4 + a_5 x_m + a_6 y_m \end{aligned} \tag{34}$$

Escrevendo-se as três primeiras da equação (34) na forma matricial, tem-se:

$$\begin{Bmatrix} u_i \\ u_j \\ u_m \end{Bmatrix} = \begin{bmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_m & y_m \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} \tag{35}$$

que resolvidas para os "*a*'s", obtém-se:

$$\{a\} = [x]^{-1} \{u\} \tag{36}$$

onde [x] é uma matriz 3 por 3. O método dos cofatores é um método que possibilita encontrar o inverso de [x]. Portanto,

$$[x]^{-1} = \frac{1}{2A}\begin{bmatrix} \alpha_i & \alpha_j & \alpha_m \\ \beta_i & \beta_j & \beta_m \\ \gamma_i & \gamma_j & \gamma_m \end{bmatrix} \tag{37}$$

onde,

$$2A = \det[x] = \begin{bmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_m & y_m \end{bmatrix} \tag{38}$$

é a determinante de [x],

$$2A = x_i\left(y_j - y_m\right) + x_j\left(y_m - y_i\right) + x_m\left(y_i - y_j\right) \tag{39}$$

sendo A a área do triângulo, e

$$\begin{array}{lll} \alpha_i = x_j y_m - y_j x_m & \alpha_j = y_i x_m - x_i y_m & \alpha_m = x_i y_j - y_i x_j \\ \beta_i = y_j - y_m & \beta_j = y_m - y_i & \beta_m = y_i - y_j \\ \gamma_i = x_m - x_j & \gamma_j = x_i - x_m & \gamma_m = x_j - x_i \end{array} \tag{40}$$

Tendo determinado $[x]^{-1}$, a equação (36) pode ser expressa em forma de matriz expandida

$$\begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \frac{1}{2A}\begin{bmatrix} \alpha_i & \alpha_j & \alpha_m \\ \beta_i & \beta_j & \beta_m \\ \gamma_i & \gamma_j & \gamma_m \end{bmatrix}\begin{Bmatrix} u_i \\ u_j \\ u_m \end{Bmatrix} \tag{41}$$

Em procedimento análogo, para os três últimos das equações (34) obtém-se:

$$\begin{Bmatrix} a_4 \\ a_5 \\ a_6 \end{Bmatrix} = \frac{1}{2A}\begin{bmatrix} \alpha_i & \alpha_j & \alpha_m \\ \beta_i & \beta_j & \beta_m \\ \gamma_i & \gamma_j & \gamma_m \end{bmatrix}\begin{Bmatrix} v_i \\ v_j \\ v_m \end{Bmatrix} \tag{42}$$

Definidas as constantes presentes nas funções de deslocamento u(x, y) e v(x, y), é possível definir as funções de aproximação de $\{\psi\}$. O procedimento será definido para o deslocamento *u*, sendo análogo o procedimento para a definição do deslocamento *v*. Assim, começando com as equações (32) expressas em forma matricial, tem-se:

$$\{u\} = \begin{bmatrix} 1 & x & y \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} \tag{43}$$

substituindo a equação (41) na equação (43), obtém-se:

$$\{u\} = \frac{1}{2A} \begin{bmatrix} 1 & x & y \end{bmatrix} \begin{bmatrix} \alpha_i & \alpha_j & \alpha_m \\ \beta_i & \beta_j & \beta_m \\ \gamma_i & \gamma_j & \gamma_m \end{bmatrix} \begin{Bmatrix} u_i \\ u_j \\ u_m \end{Bmatrix} \tag{44}$$

que pode ser rescrita de tal forma que,

$$\{u\} = \frac{1}{2A} \begin{bmatrix} 1 & x & y \end{bmatrix} \begin{Bmatrix} \alpha_i u_i + \alpha_j u_j + \alpha_m u_m \\ \beta_i u_i + \beta_j u_j + \beta_m u_m \\ \gamma_i u_i + \gamma_j u_j + \gamma_m u_m \end{Bmatrix} \tag{45}$$

multiplicando as duas matrizes na equação (45) e rearranjando-a, obtém-se:

$$u(x,y) = \frac{1}{2A} \left\{ \left(\alpha_i + \beta_i x + \gamma_i y\right) u_i + \left(\alpha_j + \beta_j x + \gamma_j y\right) u_j + \left(\alpha_m + \beta_m x + \gamma_m y\right) u_m \right\} \tag{46}$$

e, analogamente:

$$v(x,y) = \frac{1}{2A} \left\{ \left(\alpha_i + \beta_i x + \gamma_i y\right) v_i + \left(\alpha_j + \beta_j x + \gamma_j y\right) v_j + \left(\alpha_m + \beta_m x + \gamma_m y\right) v_m \right\} \tag{47}$$

Para expressar as equações (46) e (47) para *u* e *v* em uma forma mais simples, define-se:

$$
\begin{aligned}
N_i &= \frac{1}{2A}\left(\alpha_i + \beta_i x + \gamma_i y\right) \\
N_j &= \frac{1}{2A}\left(\alpha_j + \beta_j x + \gamma_j y\right) \\
N_m &= \frac{1}{2A}\left(\alpha_m + \beta_m x + \gamma_m y\right)
\end{aligned} \tag{48}
$$

Assim, usando as equações (48), é possível escrever:

$$
\begin{aligned}
u\left(x,y\right) &= N_i u_i + N_j u_j + N_m u_m \\
v\left(x,y\right) &= N_i v_i + N_j v_j + N_m v_m
\end{aligned} \tag{49}
$$

E rearranjada na forma matricial:

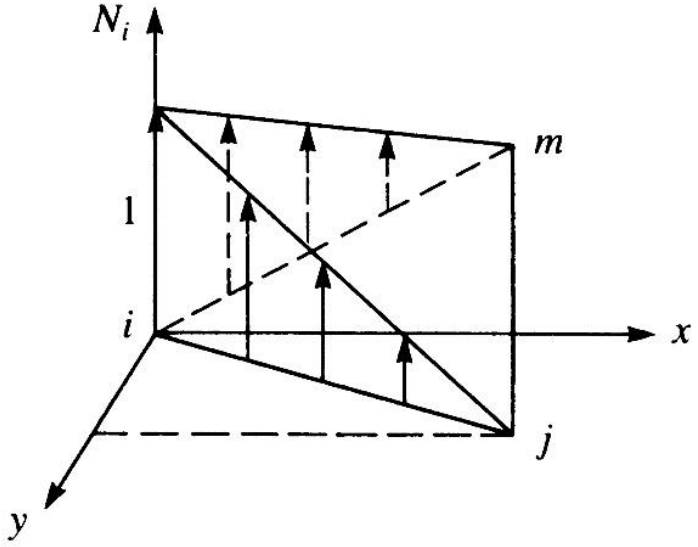
$$
\{\psi\} = \begin{bmatrix} N_i & 0 & N_j & 0 & N_m & 0 \\ 0 & N_i & 0 & N_j & 0 & N_m \end{bmatrix} \begin{Bmatrix} u_i \\ v_i \\ u_j \\ v_j \\ u_m \\ v_m \end{Bmatrix} \tag{50}
$$

Ou, simplesmente:

$$
\{\psi\} = [N]\{d\} \tag{51}
$$

As funções de forma representam a forma de {ψ} quando plotadas sobre superfície de um elemento típico. A Figura 9 mostra a variação da forma do N_i plotando sobre a superfície de um elemento típico. Note que N_i não é igual a zero, exceto ao longo de uma linha conectando e incluindo os nós j e m (LOGAN, 2007). As funções de forma também são usadas para determinar as forças do corpo e da superfície nos nós do elemento.

Figura 9 - Variação de Ni sobre a superfície x-y de um elemento típico.



Fonte: Logan (2007).

Como visto no item 4.1.2, deste trabalho, as deformações associadas ao elemento bidimensional são dadas por:

$$\{\varepsilon\} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \begin{Bmatrix} \dfrac{\partial u}{\partial x} \\ \dfrac{\partial v}{\partial y} \\ \dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x} \end{Bmatrix} \tag{52}$$

Usando as equações (49) para os deslocamentos, tem-se:

$$\frac{\partial u}{\partial x} = u_{,x} = \frac{\partial}{\partial x}\left(N_i u_i + N_j u_j + N_m u_m \right) \tag{53}$$

ou,

$$u_{,x} = N_{i,x} u_i + N_{j,x} u_j + N_{m,x} u_m \tag{54}$$

onde a vírgula seguida por uma variável indica diferenciação em relação à essa variável. Considera-se $u_{i,x} = 0$ porque $u_i = u(x_i,y_i)$ é um valor constante, da mesma forma que, $u_{j,x} = 0$ e $u_{m,x} = 0$.

Usando equação (48), é possível avaliar as expressões para as derivadas das funções de forma na equação (54) como segue:

$$N_{i,x} = \frac{1}{2A}\frac{\partial}{\partial x}\left(\alpha_i + \beta_i x + \gamma_i y\right) = \frac{\beta_i}{2A} \tag{55}$$

$N_{j,x}$ e $N_{m,x}$ são similares. Portanto, usando as equações (54) e (55) na equação (53), tem-se:

$$\frac{\partial u}{\partial x} = \frac{1}{2A}\left(\beta_i u_i + \beta_j u_j + \beta_m u_m\right) \tag{56}$$

Da mesma forma pode-se obter,

$$\begin{aligned} \frac{\partial v}{\partial y} &= \frac{1}{2A}\left(\gamma_i v_i + \gamma_j v_j + \gamma_m v_m\right) \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} &= \frac{1}{2A}\left(\gamma_i u_i + \beta_i v_i + \gamma_j u_j + \beta_j v_j + \gamma_m u_m + \beta_m v_m\right) \end{aligned} \tag{57}$$

Usando as equações (56) e (57) na equação (52), obtém-se:

$$\{\varepsilon\} = \frac{1}{2A}\begin{bmatrix} \beta_i & 0 & \beta_j & 0 & \beta_m & 0 \\ 0 & \gamma_i & 0 & \gamma_j & 0 & \gamma_m \\ \gamma_i & \beta_i & \gamma_j & \beta_j & \gamma_m & \beta_m \end{bmatrix}\begin{Bmatrix} u_i \\ v_i \\ u_j \\ v_j \\ u_m \\ v_m \end{Bmatrix} \tag{58}$$

ou,

$$\{\varepsilon\} = \begin{bmatrix} B_i & B_j & B_m \end{bmatrix}\begin{Bmatrix} d_i \\ d_j \\ d_m \end{Bmatrix} \tag{59}$$

onde,

$$[B_i] = \frac{1}{2A}\begin{bmatrix} \beta_i & 0 \\ 0 & \gamma_i \\ \gamma_i & \beta_i \end{bmatrix} \quad [B_i] = \frac{1}{2A}\begin{bmatrix} \beta_j & 0 \\ 0 & \gamma_j \\ \gamma_j & \beta_j \end{bmatrix} \quad [B_m] = \frac{1}{2A}\begin{bmatrix} \beta_m & 0 \\ 0 & \gamma_m \\ \gamma_m & \beta_m \end{bmatrix} \tag{60}$$

Finalmente, na forma de matriz simplificada, a equação (59) pode ser escrita como

$$\{\varepsilon\} = [B]\{d\} \tag{61}$$

onde,

$$[B] = \begin{bmatrix} B_i & B_j & B_m \end{bmatrix} \tag{62}$$

A matriz B é independente de *x* e *y*. Depende unicamente das coordenadas nodais do elemento, como visto nas equações (60) e (40). Assim, as deformações na equação (59) serão constantes.

Como visto no item 2.1.3 deste trabalho, a relação de tensão-deformação no estado plano é dada por:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = [D]\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \tag{63}$$

onde [D] é dado pela equação (19) para problemas de tensão no plano. As tensões {σ} são constantes em todo o domínio do elemento.

A matriz de rigidez para o elemento *CST* pode ser obtida dada pela equação (30) derivada do PTV. Como o elemento tem espessura *t* constante, tem-se:

$$[k] = t\iint [B]^T [D][B]\,dxdy \tag{64}$$

onde o integrando não é uma função de x ou y e, portanto, pode ser retirado da integral para gerar:

$$[k] = tA[B]^T [D][B] \tag{65}$$

onde A é dado pela equação (39), [*B*] é dado é dado pela equação (62) e [*D*] é dado é dado pela equação (19).

Da equação (65) vê-se que [*k*] é uma função das coordenadas nodais e das propriedades mecânicas *E* e ν. A expansão da equação (65) para um elemento é:

$$[k] = \begin{bmatrix} [k_{ii}] & [k_{ij}] & [k_{im}] \\ [k_{ji}] & [k_{jj}] & [k_{jm}] \\ [k_{mi}] & [k_{mj}] & [k_{mm}] \end{bmatrix} \tag{66}$$

onde as submatrizes são dadas por:

$$\begin{aligned} [k_{ii}] &= [B_i]^T [D][B_i] tA \\ [k_{ij}] &= [B_i]^T [D][B_j] tA \\ [k_{im}] &= [B_i]^T [D][B_m] tA \end{aligned} \tag{67}$$

e assim em diante. Nas equações (67), [B_i], [B_j] e [B_m] são definidos pelas equações (60). A matriz [*k*] é definida como uma matriz 6 por 6 (sua ordem é igual ao número de graus de liberdade por nó multiplicado pelo número total de nós por elemento. O elemento triangular é composto por 3 nós e possui 2 graus de liberdade por nó).

Para o elemento *CST*, as forças do corpo e da superfície podem ser agrupadas nos nós com resultados equivalentes e adicionadas às forças nodais concentradas para obter o vetor de forças do elemento. As equações do elemento são então dadas por,

$$\begin{Bmatrix} f_{1x} \\ f_{1y} \\ f_{2x} \\ f_{2y} \\ f_{3x} \\ f_{3y} \end{Bmatrix} = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{16} \\ k_{21} & k_{22} & \cdots & k_{26} \\ \vdots & \vdots & \ddots & \vdots \\ k_{61} & k_{62} & \cdots & k_{66} \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{Bmatrix} \tag{68}$$

Ao substituir as matrizes [*D*] e [*B*] na equação (65), obtém-se:

$$[k] = \frac{Et}{4A\left(1-\nu^2\right)} \begin{bmatrix} \beta_i & 0 & \gamma_i \\ 0 & \gamma_i & \beta_i \\ \beta_j & 0 & \gamma_j \\ 0 & \gamma_j & \beta_j \\ \beta_m & 0 & \gamma_m \\ 0 & \gamma_m & \beta_m \end{bmatrix} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \dfrac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} \beta_i & 0 & \beta_j & 0 & \beta_m & 0 \\ 0 & \gamma_i & 0 & \gamma_j & 0 & \gamma_m \\ \gamma_i & \beta_i & \gamma_j & \beta_j & \gamma_m & \beta_m \end{bmatrix} \tag{69}$$

Multiplicando as matrizes na equação (69), obtém-se a matriz de rigidez explícita do elemento *CST*:

$$[k] = \frac{Et}{4A\left(1-\nu^2\right)} \begin{bmatrix} \beta_i^2 + C\gamma_i^2 & \dfrac{1+\nu}{2}\beta_i\gamma_i & \beta_i\beta_j + C\gamma_i\gamma_j & \nu\beta_i\gamma_j + C\beta_j\gamma_i & \beta_i\beta_m + C\gamma_i\gamma_m & \nu\beta_i\gamma_m + C\beta_i\gamma_m \\ & \gamma_i^2 + C\beta_i^2 & \nu\beta_j\gamma_i + C\beta_i\gamma_j & \gamma_i\gamma_j + C\beta_i\beta_j & \nu\beta_m\gamma_i + C\beta_i\gamma_m & \gamma_i\gamma_m + C\beta_i\beta_m \\ & & \beta_j^2 + C\gamma_j^2 & \dfrac{1+\nu}{2}\beta_j\gamma_j & \beta_j\beta_m + C\gamma_j\gamma_m & \nu\beta_j\gamma_m + C\beta_m\gamma_j \\ & \textit{Simétrica} & & \gamma_j^2 + C\beta_j^2 & \nu\beta_m\gamma_j + C\beta_j\gamma_m & \gamma_j\gamma_m + C\beta_j\beta_m \\ & & & & \beta_m^2 + C\gamma_m^2 & \dfrac{1+\nu}{2}\beta_m\gamma_m \\ & & & & & \gamma_m^2 + C\beta_m^2 \end{bmatrix} \tag{70}$$

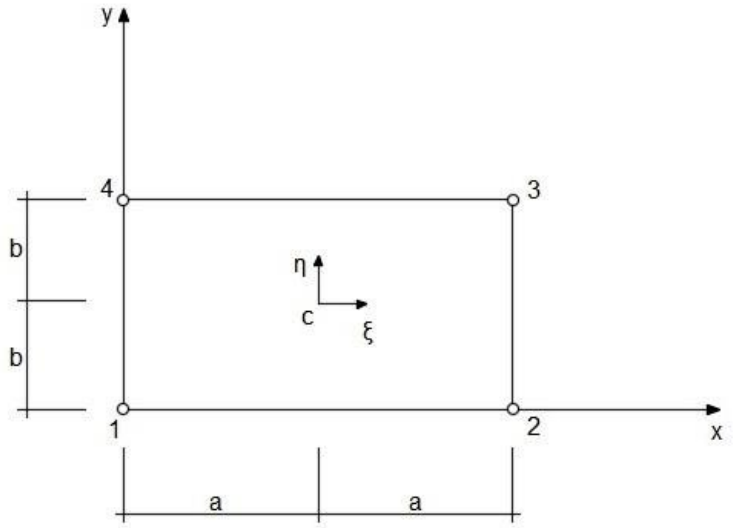
Note que [k] é uma função da diferença nas coordenas nodais x e y, como indicado pelos γ e β, das propriedades do material E e ν, e da espessura t e da área de superfície A do elemento. Onde a constante C = (1 - ν)/2.

2.3.3 Elemento Retangular *CSQ* (*Constant Strain Quadrangular*)

Como relatado anteriormente, o elemento *CST* é o elemento mais simples para a análise de chapas. Todavia, segundo Logan (2007) em um elevado número de casos com interesse prático, a discretização através de elementos finitos *CSQ* é mais cômoda. Por interpolar os deslocamentos com funções de graus maiores, o elemento *CSQ* proporciona melhores resultados que o elemento *CST* quando se utiliza o mesmo número de elementos na malha (LOGAN, 2007).

O elemento *CSQ* é definido por 4 nós, também localizados nos vértices de seus elementos. Os graus de liberdade dos nós estão contidos no plano dos elementos, onde apenas considera-se os deslocamentos na vertical e na horizontal (x e y); seus pontos nodais são convencionalmente numerados a partir do número 1 e no sentido anti-horário, como ilustra a Figura 10.

Figura 10 - Elemento *CSQ* com Coordenadas Baricêntricas.



Fonte: Autoria Própria.

No caso dos elementos retangulares, para se facilitar as integrações, as coordenadas adimensionais baricêntricas (ξ, η) são mais cômodas de se trabalhar do que as coordenadas cartesianas (*x, y*) (SAVASSI, 1996). As coordenadas baricêntricas são definidas por:

$$\xi = \frac{(x - x_c)}{a} \qquad \eta = \frac{(y - y_c)}{b} \tag{71}$$

onde 2*a* e 2*b* são as dimensões do elemento finito nas direções x e y, respectivamente.

Deste modo a integração de certa função f(x,y) no domínio do retângulo fica:

$$\int_A f(x,y)\,dA = \int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta)\,d\xi\,d\eta \tag{72}$$

Inicialmente, são determinadas as funções de interpolação do elemento de quatro pontos nodais e de coordenadas locais normalizadas ortogonais (ξ, η) representado na Figura 10, em que ($-1 \le \xi \le +1$) e ($-1 \le \eta \le +1$). Como essas funções devem atender à propriedade do delta de Kronecker ($N_i(x_i,y_i)=\delta_{ij}$), elas podem ser obtidas através da multiplicação de funções lineares na coordenada ξ por funções lineares na coordenada η, que têm valor unitário em um dos pontos nodais e valor nulo no outro ponto nodal. (SORIANO, 2009)

Seleciona-se então uma função de deslocamento linear para cada elemento como,

$$\begin{aligned} u(\xi,\eta) &= \alpha_1 + \alpha_2\xi + \alpha_3\xi\eta + \alpha_4\eta \\ v(\xi,\eta) &= \alpha_5 + \alpha_6\xi + \alpha_7\xi\eta + \alpha_8\eta \end{aligned} \tag{73}$$

onde u(ξ, η) e v(ξ, η) descrevem deslocamentos em qualquer ponto interior do elemento.

Considerando por enquanto apenas a função u, a equação (73) pode ser expressa em forma matricial:

$$\{u\} = \begin{bmatrix} 1 & \xi & \xi\eta & \xi \end{bmatrix} \begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{Bmatrix} \tag{74}$$

e em forma de matriz simplificada:

$$\{u\} = [\omega]\{\alpha\} \tag{75}$$

onde [ω] é a matriz que contém as coordenadas baricêncentricas dos pontos nodais.

As constantes α, são obtidas substituindo as coordenadas baricêntricas dos pontos nodais nas equações (73). Assim, considerando apenas a função u:

$$\begin{aligned} u_1 &= u(-1,-1) = \alpha_1 - \alpha_2 + \alpha_3 - \alpha_4 \\ u_2 &= u(+1,-1) = \alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 \\ u_3 &= u(+1,+1) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \\ u_4 &= u(-1,+1) = \alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 \end{aligned} \tag{76}$$

que expressa de forma de matricial resulta em:

$$\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{Bmatrix} \tag{77}$$

e em forma de matriz simplificada:

$$\{u_i\} = [\omega_i]\{\alpha\} \tag{78}$$

que, resolvidas para os "α's", obtém-se:

$$\{\alpha\} = [\omega_i]^{-1}\{u_i\} \tag{79}$$

Substituindo a equação (79) na equação (75), obtem-se:

$$\{u\} = [\omega][\omega_i]^{-1}\{u_i\} \tag{80}$$

onde define-se que:

$$[N] = [\omega][\omega_i]^{-1} \tag{81}$$

Sabendo que a matriz $[\omega_i]^{-1}$ é a matriz inversa da matriz $[\omega_i]$ definida pela equação (77) e pode ser escrita de modo que:

$$[\omega_i]^{-1} = \left(\frac{1}{4}\right)\begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{82}$$

Deste modo, substituindo as equações (74) e (82) na equação (81), obtém-se:

$$[N] = [1 \quad \xi \quad \xi\eta \quad \xi]\left(\frac{1}{4}\right)\begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{83}$$

As funções de interpolação podem ser expressas por:

$$N_1 = \frac{(1-\xi)(1-\eta)}{4} \; ; \; N_2 = \frac{(1+\xi)(1-\eta)}{4} \; ; \; N_3 = \frac{(1+\xi)(1+\eta)}{4} \; ; \; N_4 = \frac{(1-\xi)(1+\eta)}{4} \tag{84}$$

Como informado anteriormente, a dedução matemática foi desenvolvida considerando apenas os deslocamentos horizontais *u*, porém esta dedução é análoga para os deslocamentos verticais *v*. De modo que:

$$\begin{aligned} \{u\} &= [N]\{u_i\} \\ \{v\} &= [N]\{v_i\} \end{aligned} \tag{85}$$

Sabendo que a matriz [B], que relaciona deformações e deslocamentos nodais, é obtida através do produto entre os operadores diferenciais e as funções de interpolação como representada pela equação (23), tem-se que:

$$[B]=\begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{bmatrix} \tag{86}$$

Do produto das matrizes da equação (86), obtém-se:

$$[B]=\begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_3}{\partial x} & 0 & \frac{\partial N_4}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_3}{\partial y} & 0 & \frac{\partial N_4}{\partial y} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x} & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial x} \end{bmatrix} \tag{87}$$

sendo sua matriz transposta,

$$[B]^T=\begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_1}{\partial y} \\ 0 & \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} \\ \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_2}{\partial y} \\ 0 & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} \\ \frac{\partial N_3}{\partial x} & 0 & \frac{\partial N_3}{\partial y} \\ 0 & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x} \\ \frac{\partial N_4}{\partial x} & 0 & \frac{\partial N_4}{\partial y} \\ 0 & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial x} \end{bmatrix} \tag{88}$$

Por fim, para chegar à matriz de rigidez do elemento *CSQ*, substitui-se a equação (72) na equação (64). Deste modo, obtém-se:

$$[k] = t \cdot a \cdot b \cdot \int_{-1}^{1} \int_{-1}^{1} [B]^{T} \cdot [D] \cdot [B] \cdot d\xi \cdot d\eta \tag{89}$$

Ao resolver a equação (89) pelo processo de integração analítica, obtém-se a matriz de rigidez do elemento finito *CSQ*. Esta matriz é simétrica e de ordem oito, como segue:

$$[k] = \frac{t \cdot E}{(1-\nu^2)} \cdot \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} & k_{17} & k_{18} \\ \vdots & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} & k_{27} & k_{28} \\ \vdots & & k_{33} & k_{34} & k_{35} & k_{36} & k_{37} & k_{38} \\ \vdots & & & k_{44} & k_{45} & k_{46} & k_{47} & k_{48} \\ \vdots & & & \iddots & k_{55} & k_{56} & k_{57} & k_{58} \\ \vdots & & \iddots & & & k_{66} & k_{67} & k_{68} \\ \vdots & \iddots & & & & & k_{77} & k_{78} \\ sim. & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & k_{88} \end{bmatrix} \tag{90}$$

$$k_{11} = \frac{2b^2 + (1-\nu)a^2}{6ab};$$

$$k_{12} = \frac{(1+\nu)}{8};$$

$$k_{13} = \frac{-4b^2 + (1-\nu)a^2}{12ab};$$

$$k_{14} = \frac{(-1+3\nu)}{8};$$

$$k_{15} = \frac{-2b^2 + (-1+\nu)a^2}{12ab};$$

$$k_{16} = \frac{(-1-\nu)}{8};$$

$$k_{17} = \frac{b^2 + (-1+\nu)a^2}{6ab};$$

$$k_{18} = \frac{(1-3\nu)}{8};$$

$$k_{22} = \frac{2a^2 + (1-\nu)b^2}{6ab};$$

$$k_{23} = \frac{(1-3\nu)}{8};$$

$$k_{24} = \frac{a^2 + (-1+\nu)b^2}{6ab};$$

$$k_{25} = \frac{(-1-\nu)}{8};$$

$$k_{26} = \frac{-2a^2 + (-1+\nu)b^2}{12ab};$$

$$k_{27} = \frac{(-1+3\nu)}{8};$$

$$k_{28} = \frac{-4a^2 + (1-\nu)b^2}{12ab};$$

$$k_{33} = \frac{2b^2 + (1-\nu)a^2}{6ab};$$

$$k_{34} = \frac{(-1-\nu)}{8};$$

$$k_{35} = \frac{b^2 + (-1+\nu)a^2}{6ab};$$

$$k_{36} = \frac{(-1+3\nu)}{8};$$

$$k_{37} = \frac{-2b^2 + (-1+\nu)a^2}{12ab};$$

$$k_{38} = \frac{(1+\nu)}{8};$$

$$k_{44} = \frac{2a^2 + (1-\nu)b^2}{6ab};$$

$$k_{45} = \frac{(1-3\nu)}{8};$$

$$k_{46} = \frac{-4a^2 + (1-\nu)b^2}{12ab};$$

$$k_{47} = \frac{(1+\nu)}{8};$$

$$k_{48} = \frac{-2a^2 + (-1+\nu)b^2}{12ab};$$

$$k_{55} = \frac{2b^2 + (1-\nu)a^2}{6ab};$$

$$k_{56} = \frac{(1+\nu)}{8};$$

$$k_{57} = \frac{-4b^2 + (1-\nu)a^2}{12ab};$$

$$k_{58} = \frac{(-1+3\nu)}{8};$$

$$k_{66} = \frac{2a^2 + (1-\nu)b^2}{6ab};$$

$$k_{67} = \frac{(1-3\nu)}{8};$$

$$k_{68} = \frac{a^2 + (-1+\nu)b^2}{6ab};$$

$$k_{77} = \frac{2b^2 + (1-\nu)a^2}{6ab};$$

$$k_{78} = \frac{(-1-\nu)}{8};$$

$$k_{88} = \frac{2a^2 + (1-\nu)b^2}{6ab};$$

3 ASPECTOS COMPUTACIONAIS

Como mencionado anteriormente, neste trabalho foi desenvolvido um algoritmo computacional em linguagem de programação *Python*, com base no método dos elementos finitos e voltado para análise elástica de chapas. A linguagem *Python* foi adotada por ser uma linguagem de alto nível e possuir bibliotecas que facilitam o desenvolvimento do código computacional. No desenvolvimento do *software* utilizou-se a biblioteca *Numpy* para resolução e manipulação das matrizes e vetores e a biblioteca *Math* para resolução de funções trigonométricas.

O *software* foi desenvolvido com a divisão de três módulos principais, sendo: entrada de dados, processamento e saída de dados.

No módulo de entrada de dados é realizada a leitura dos dados do problema através de um arquivo de texto (".txt") elaborado pelo usuário, contendo, em sequência pré-determinada, informações sobre a estrutura em análise, sendo elas: propriedades físicas e geométricas de cada elemento, vinculações e carregamentos nodais. O apêndice B traz um arquivo de entrada de dados comentado.

No módulo de processamento são realizados todos os cálculos necessários para se determinar os resultados desejados. Dentre eles, vale ressaltar a transformação de coordenadas, elaboração do vetor de esforços, montagem da matriz de rigidez da estrutura em coordenadas globais, resolução do sistema linear de equações algébricas, determinação de esforços e deslocamentos.

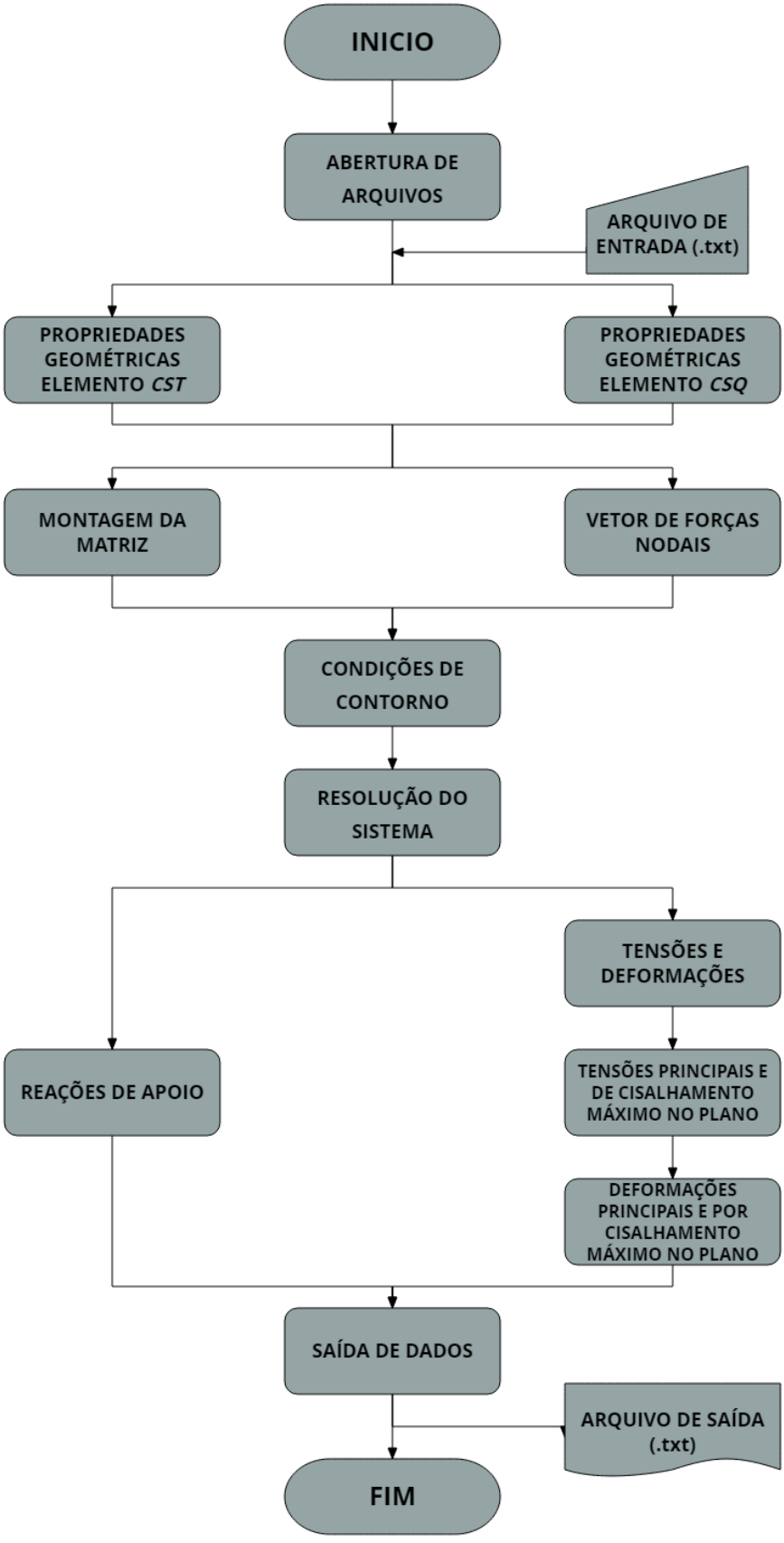
Por fim, o módulo de saída de dados tem como objetivo organizar e apresentar de forma simples os resultados das análises realizadas em um arquivo ".txt" contendo informações a respeito dos deslocamentos nodais, tensões e deformações, tensões principais, tensão de cisalhamento máxima no plano, deformações principais, deformação por cisalhamento máxima no plano, ângulos principais e as reações de apoio. O apêndice C apresenta um arquivo de saída de dados gerado pelo programa desenvolvido.

3.1 Esquema Geral de Cálculo

O fluxograma da Figura 11 apresenta uma melhor visualização da organização do código computacional que está dividido em etapas e por operação, sendo cada processo mostrado uma representação de uma sub-rotina do programa.

A descrição detalhada de cada sub-rotina será apresentada no item seguinte do trabalho.

Figura 11 - Esquema geral de cálculo



Fonte: Autoria própria

3.2 Sub-Rotinas

3.2.1 Abertura de Arquivos

Nesta sub-rotina informa-se o nome do arquivo dos dados de entrada que será aberto durante a execução do programa, onde estará contida toda informação da estrutura analisada, como propriedades físicas, geométricas, características dos esforços e definição dos apoios. A mesma executa a interpretação do arquivo de entrada de dados, acoplando as informações em suas respectivas variáveis.

3.2.2 Propriedades Geométricas

Uma vez que as variáveis já receberam os dados correspondentes, a presente sub-rotina é responsável por realizar os cálculos das propriedades geométricas de cada elemento. Neste caso, foram implantadas duas sub-rotinas, uma voltada para realizar os cálculos das propriedades dos elementos *CST* (β, γ e A) e outra voltada para as propriedades dos elementos *CSQ* (a e b).

3.2.3 Matriz de Rigidez

Nesta sub-rotina a matriz de rigidez global da estrutura [k] é montada a partir do cálculo individual das matrizes de rigidez dos elementos *CST* e *CSQ* segundo as equações da matrizes (70) e (90).

Após obter as matrizes de rigidez dos elementos, deve-se agrupá-las de forma que os termos de cada elemento sejam alocados em suas posições pré-estabelecidas dentro da matriz [k], de modo que os termos referentes aos nós comuns entre dois elementos ou mais se somem. A equação (91) demonstra, simplificadamente, esse procedimento.

$$[k]_{global} = \begin{bmatrix} k_{11}^1 & k_{12}^1 & k_{13}^1 & k_{14}^1 & k_{15}^1 & k_{16}^1 & \cdots & 0 \\ k_{21}^1 & k_{22}^1 & k_{23}^1 & k_{24}^1 & k_{25}^1 & k_{26}^1 & \cdots & 0 \\ k_{31}^1 & k_{32}^1 & k_{33}^1 & k_{34}^1 & k_{35}^1 & k_{36}^1 & \cdots & 0 \\ k_{41}^1 & k_{42}^1 & k_{43}^1 & k_{44}^1 + k_{11}^2 & k_{45}^1 + k_{12}^2 & k_{46}^1 + k_{13}^2 & \cdots & 0 \\ k_{51}^1 & k_{52}^1 & k_{53}^1 & k_{54}^1 + k_{21}^2 & k_{55}^1 + k_{22}^2 & k_{56}^1 + k_{23}^2 & \cdots & 0 \\ k_{61}^1 & k_{62}^1 & k_{63}^1 & k_{64}^1 + k_{31}^2 & k_{65}^1 + k_{32}^2 & k_{66}^1 + k_{33}^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{nn}^n \end{bmatrix} \tag{91}$$

3.2.4 Vetor de Forças Nodais

Nesta etapa ocorre a montagem do vetor de forças nodais {F} que consiste nos carregamentos nodais de acordo com a equação (92):

$$\{F\} = \begin{Bmatrix} f_{1x} \\ f_{1y} \\ f_{2x} \\ f_{2y} \\ \vdots \\ f_{ny} \end{Bmatrix} \tag{92}$$

3.2.5 Condições de Contorno

Nesta sub-rotina são inseridas as condições de contorno na matriz de rigidez da estrutura e no vetor de forças globais.

As condições de contorno são inseridas conforme os graus de liberdade de cada nó. Para tal, adiciona-se, na matriz de rigidez, o número 1 na posição da diagonal principal da linha correspondente ao grau de liberdade restrito. As demais posições de linha e coluna referente ao grau de liberdade, e, a posição do mesmo no vetor de esforços, são anuladas.

Esse processo é repetido para todos os nós da malha utilizada, como está representado de forma generalizada na equação (93):

$$\begin{Bmatrix} k_{11} & 0 & k_{13} & k_{14} & 0 & k_{16} & \cdots & k_{1n} \\ 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ k_{31} & 0 & k_{33} & k_{34} & 0 & k_{36} & \cdots & k_{3n} \\ k_{41} & 0 & k_{43} & k_{44} & 0 & k_{46} & \cdots & k_{4n} \\ 0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\ k_{61} & 0 & k_{63} & k_{64} & 0 & k_{66} & \cdots & k_{6n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ k_{n1} & 0 & k_{n3} & k_{n4} & 0 & k_{n6} & \cdots & k_{nn} \end{Bmatrix} . \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ \vdots \\ v_n \end{Bmatrix} = \begin{Bmatrix} f_{1x} \\ 0 \\ f_{2x} \\ f_{2y} \\ 0 \\ f_{3y} \\ \vdots \\ f_{ny} \end{Bmatrix} \tag{93}$$

3.2.6 Resolução de Sistemas

Nesta sub-rotina é realizada a resolução de sistemas lineares de equações algébricas utilizando a biblioteca *Numpy*. A solução do sistema linear permite encontrar o vetor de deslocamentos nodais.

3.2.7 Reações de Apoio

Com os deslocamentos nodais que foram obtidos através da etapa anterior, cabe a essa sub-rotina a multiplicação entre o vetor de deslocamento nodais com a matriz de rigidez global da estrutura sem o emprego das condições de contorno, obtendo-se um vetor que contém todas as forças nodais da estrutura, de acordo com a equação (94).

$$\{F\} = [k].\{u\} \tag{94}$$

É válido destacar que o vetor de esforços obtido através da equação anterior não somente contém os valores das reações de apoio como todas as forças externas aplicadas na estrutura. Entretanto, são interessados apenas os valores contidos nas posições referentes aos nós que apresentem algum tipo de restrição.

3.2.8 Tensões e deformações

Esta etapa tem a função de obter as tensões e deformações de cada nó da malha utilizada.

De posse dos valores de deslocamentos fornecidos pela solução do sistema, obtém-se os valores das deformações nodais (normais e de cisalhamento) através das equações (58) e (23).

Utilizando-se as relações de tensão-deformação dadas pela equação (17), formulada através da Lei de Hooke generalizada, obtém-se os valores das tensões nodais (normais e de cisalhamento).

3.2.9 Tensões Principais

Com os valores das tensões e deformações nodais obtidos na etapa anterior, determina-se as tensões principais, que são as tensões normais máxima e mínima, e a tensão de cisalhamento máxima no plano, através das equações (2), (3) e (4), respectivamente. Determina-se também, através das equações (5) e (6), os ângulos principais que definem as direções principal e de cisalhamento máximo no plano.

3.2.10 Saída de Dados

Nesta sub-rotina, informa-se o nome do arquivo dos dados de saída gerado no formato ".txt". O arquivo contém todos os resultados obtidos como deslocamentos nodais, reações de apoio, tensões e deformações, tensões principais, tensão de cisalhamento máxima e os ângulos principais. Para fins acadêmicos e didáticos, o código computacional desenvolvido neste trabalho encontra-se ao final deste trabalho em forma de apêndice A.

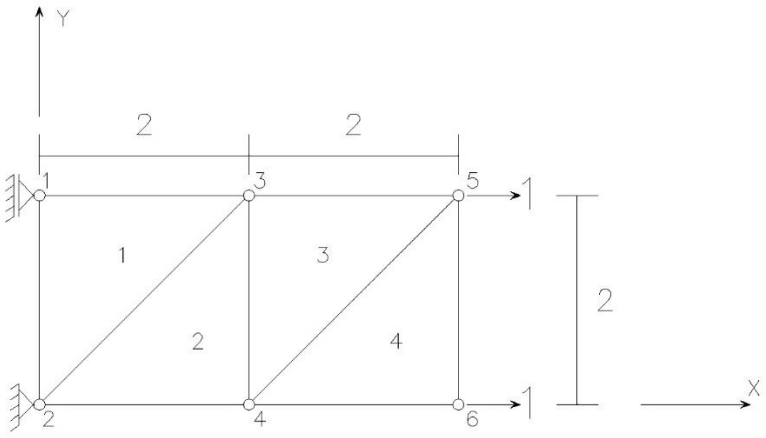
4 RESULTADOS E DISCUSSÕES

Neste capítulo é apresentado a análise e avalição dos resultados obtidos pelo programa desenvolvido, por meio de 3 exemplos utilizando diferentes malhas. Os resultados foram comparados com os obtidos por outros autores, pelo programa ANSYS versão acadêmica e por meio resolução analítica.

4.1 Exemplo 1

Este exemplo foi proposto no livro de Savassi (1996), o qual foi analisado utilizando um programa computacional desenvolvido pelo autor. O exemplo aborda uma chapa já discretizada com 6 nós e 4 elementos *CST*, no qual as condições de carregamento, vinculações e comprimento de cada elemento estão demonstradas na Figura 12.

Figura 12 - Chapa retangular submetida a cargas nodais



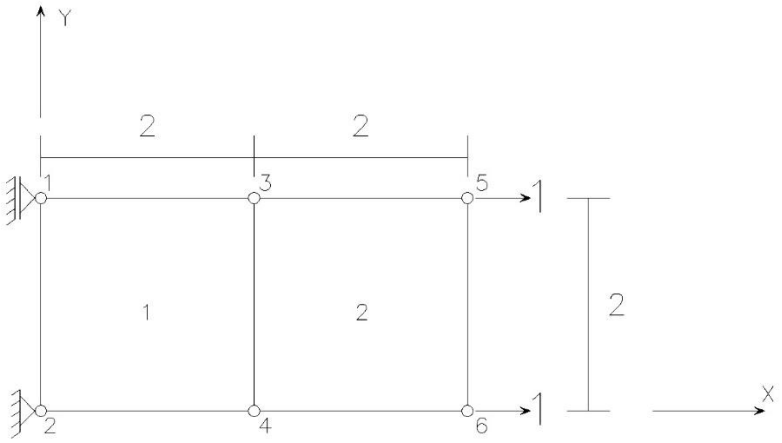
Fonte: Adaptado de Savassi (1996)

Todos os elementos possuem as mesmas propriedades físicas e geométricas, sendo estas adimensionais: espessura t = 1, módulo de elasticidade longitudinal E = 1, e coeficiente de Poisson υ = 0,25. As dimensões demonstradas na figura acima também são adimensionais.

Como o algoritmo implementado neste trabalho contempla análises com malhas compostas por dois tipos de elementos finitos, para esse exemplo foram utilizadas duas malhas, uma composta apenas por elementos *CSQ*, ilustrada pela

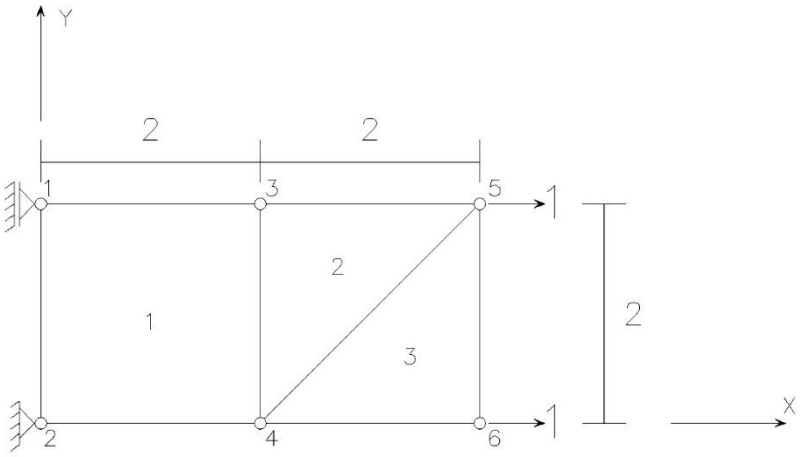
Figura 13, e outra composta pelos dois elementos, ilustrada pela Figura 14, a fim de verificar a validade e precisão do programa desenvolvido com esses tipos de malhas.

Figura 13 - Malha com elementos *CSQ*



Fonte: Autoria própria

Figura 14 - Malha mista



Fonte: Autoria própria

Deste modo, a estrutura foi simulada usando as malhas demonstradas acima no *software* elaborado, e os resultados de deslocamentos nodais, assim como os esforços internos obtidos foram confrontados com aqueles expostos pelo autor, de acordo com as tabelas a seguir.

Tabela 1 - Deslocamentos Nodais para o Exemplo 1.

Nó	Deslocamento em x			Deslocamento em y		
	CSQ	**Mista**	**Savassi**	***CSQ***	**Mista**	**Savassi**
1	0,00	0,00	0,00	-0,50	-0,50	-0,50
2	0,00	0,00	0,00	0,00	0,00	0,00
3	2,00	2,00	2,00	-0,50	-0,50	-0,50
4	2,00	2,00	2,00	0,00	0,00	0,00
5	4,00	4,00	4,00	-0,50	-0,50	-0,50
6	4,00	4,00	4,00	0,00	0,00	0,00

Tabela 2 - Deformações Nodais para o Exemplo 1.

Nó	ε_x			ε_y			γ_{xy}		
	CSQ	**Mista**	**Savassi**	***CSQ***	**Mista**	**Savassi**	***CSQ***	**Mista**	**Savassi**
1	1,00	1,00	1,00	-0,25	-0,25	-0,25	0,00	0,00	0,00
2	1,00	1,00	1,00	-0,25	-0,25	-0,25	0,00	0,00	0,00
3	1,00	1,00	1,00	-0,25	-0,25	-0,25	0,00	0,00	0,00
4	1,00	1,00	1,00	-0,25	-0,25	-0,25	0,00	0,00	0,00
5	1,00	1,00	1,00	-0,25	-0,25	-0,25	0,00	0,00	0,00
6	1,00	1,00	1,00	-0,25	-0,25	-0,25	0,00	0,00	0,00

Tabela 3 - Tensões Nodais para o Exemplo 1.

Nó	σ_x			σ_y			τ_{xy}		
	CSQ	**Mista**	**Savassi**	***CSQ***	**Mista**	**Savassi**	***CSQ***	**Mista**	**Savassi**
1	1,00	1,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00
2	1,00	1,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00
3	1,00	1,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00
4	1,00	1,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00
5	1,00	1,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00
6	1,00	1,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00

Tabela 4 - Tensões e direções principais nodais para o Exemplo 1.

Nó	σ_1			σ_2			θ_p		
	CSQ	**Mista**	**Savassi**	***CSQ***	**Mista**	**Savassi**	***CSQ***	**Mista**	**Savassi**
1	1,00	1,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00
2	1,00	1,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00
3	1,00	1,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00
4	1,00	1,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00
5	1,00	1,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00
6	1,00	1,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00

Diante da comparação dos resultados apresentados nas tabelas acima, nota-se que os valores obtidos pelo programa desenvolvido neste trabalho e os obtidos pelo desenvolvido por Savassi (1996) são iguais.

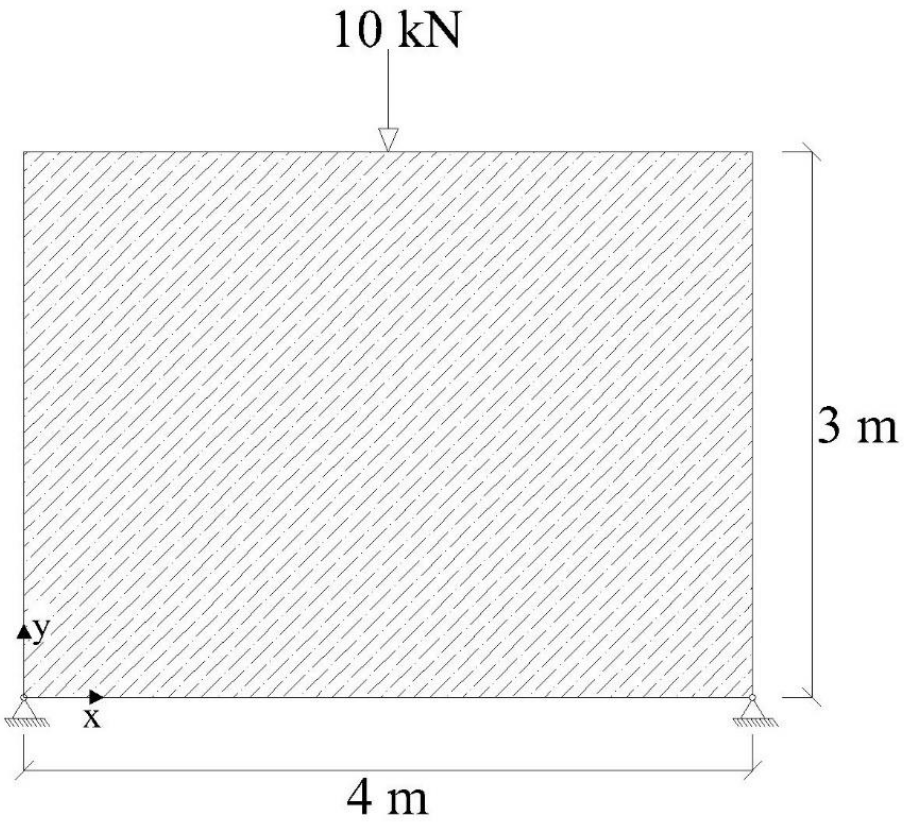
Desta forma, pode-se concluir que o programa computacional desenvolvido também se mostra eficiente utilizando malhas compostas apenas por um dos elementos e malhas mistas.

4.2 Exemplo 2

A Figura 15 representa uma chapa bi-apoiada solicitada por uma carga concentrada no centro da face superior do elemento estrutural que possui as características físicas e geométricas a seguir: Módulo de Elasticidade Transversal E = 17×10^6 kN/m^2, Coeficiente de Poisson υ = 0,25 e espessura t = 0,15 m. As dimensões da chapa e a carga concentrada estão demonstradas na Figura 15.

Para realizar a análise elástica linear da estrutura abordada por meio do programa desenvolvido, foram utilizadas duas malhas, uma apenas com elementos *CST* e outra apenas com elementos *CSQ*. É válido ressaltar que, neste exemplo, não foi analisada a convergência de malhas. O objetivo é apenas verificar se os resultados obtidos pelo *software* desenvolvido e se assemelham aos fornecidos pelo ANSYS. A quantidade de nós e elementos de cada malha estão especificadas na Tabela 5. Ambas malhas são regulares.

Figura 15 - Chapa bi-apoiada submetida a carga concentrada.



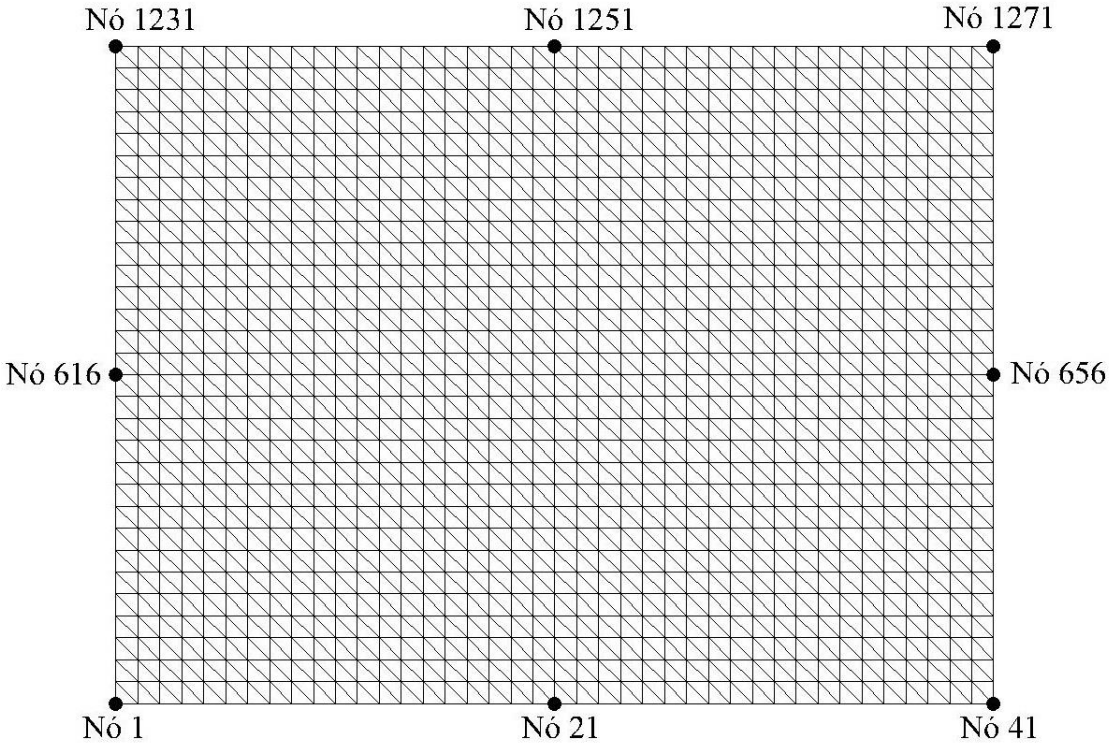
Fonte: Autoria própria.

Tabela 5 - Número de nós e elementos das malhas

Elemento da malha	Número de nós	Número de elementos
CST (40x30)	1271	2400
CSQ (40x30)	1271	1200

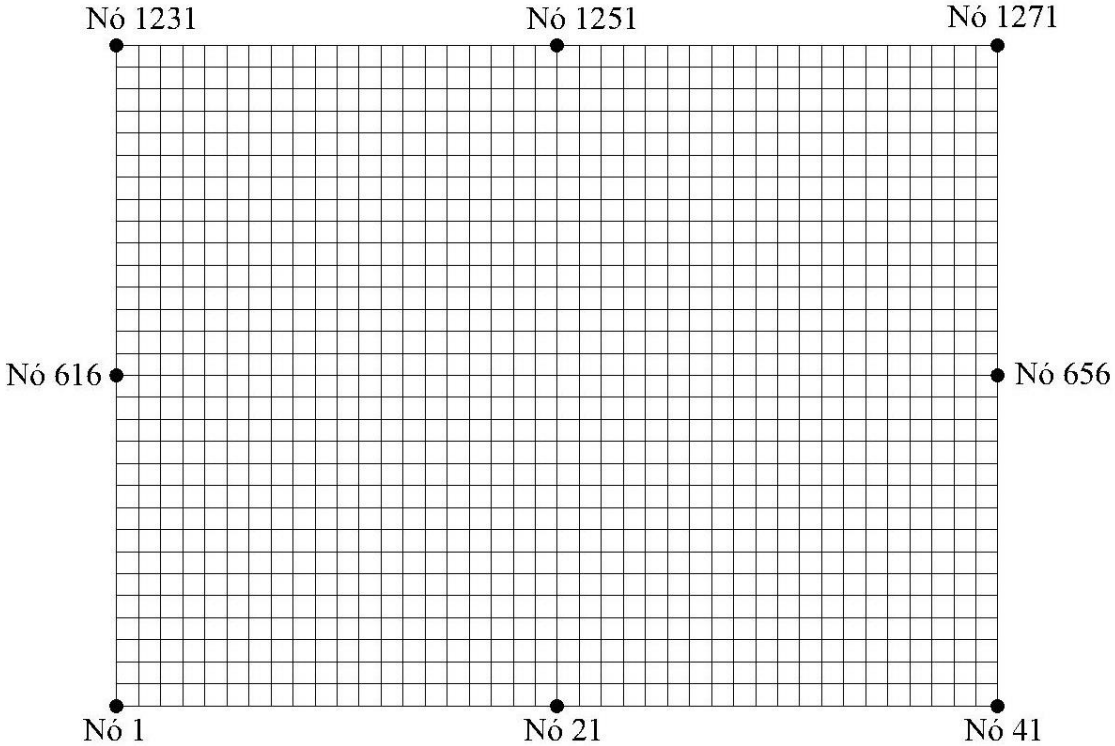
A convenção utilizada para se enumerar os nós foi seguindo o sentido da esquerda partindo do nó inferior esquerdo como demonstram as Figuras 16 e 17.

Figura 16 – Discretização da chapa utilizando elementos *CST*.



Fonte: Autoria própria.

Figura 17 - Discretização da chapa utilizando elementos *CSQ*.



Fonte: Autoria própria.

A fim de comparar os resultados obtidos pelo programa desevolvido com outro *software* específico da área foi utilizado o programa ANSYS versão acadêmica, no qual também é possivel realizar análise elástica linear de chapas utilizando elementos finitos *CST* e *CSQ*. Deste modo, a estrutura foi simulada usando as malhas já expecificadas no *software* elaborado e no ANSYS.

Em posse dos resultados obtidos por ambos programas, foram comparados os valores máximos referentes aos deslocamentos e tensões da estrutura, estes valores estão demonstrados nas tabelas abaixo:

Tabela 6 - Valores mínimos obtidos pela malha de elementos *CST* para o exemplo 2.

Valores mínimos (malha *CST*)				
	Programa desenvolvido		**ANSYS**	
	Nó	**Valor**	**Nó**	**Valor**
u (m)	1256	-8,47e-07	1256	-8,47e-07
v (m)	1251	-2,04e-05	1251	-2,04e-05
σ_x (kPa)	1251	-186,93	1251	-186,93
σ_y (kPa)	1	-449,77	1	-449,77
τ_{xy} (kPa)	1	-216,90	1	-216,90
σ_1 (kPa)	1251	-180,58	1251	-180,58
σ_2 (kPa)	1	-569,27	1	-569,27

Tabela 7 - Valores máximos obtidos pela malha de elementos *CST* para o exemplo 2.

Valores máximos (malha *CST*)				
	Programa desenvolvido		**ANSYS**	
	Nó	**Valor**	**Nó**	**Valor**
u (m)	1245	2,05e-06	1245	2,05e-06
σ_x (kPa)	12	16,518	12	16,518
σ_y (kPa)	1066	1,9307	1066	1,9307
τ_{xy} (kPa)	41	158,77	41	158,77
σ_1 (kPa)	21	16,52	21	16,52
σ_2 (kPa)	8	0,906	8	0,906

Tabela 8 - Valores mínimos obtidos pela malha de elementos *CSQ* para o exemplo 2.

Valores mínimos (malha *CSQ*)				
	Programa desenvolvido		**ANSYS**	
	Nó	**Valor**	**Nó**	**Valor**
u (m)	1254	-1,46e-06	1254	-1,46e-06
v (m)	1251	-2,29e-05	1251	-2,29e-05
σ_x (kPa)	1251	-402,46	1251	-402,46
σ_y (kPa)	1251	-716,31	1251	-716,31
τ_{xy} (kPa)	1	-356,62	1	-356,62
σ_1 (kPa)	1251	-402,46	1251	-402,46
σ_2 (kPa)	1	-943,45	1	-943,45

Tabela 9 - Valores máximos obtidos pela malha de elementos *CSQ* para o exemplo 2.

Valores máximos (malha *CSQ*)				
	Programa desenvolvido		**ANSYS**	
	Nó	**Valor**	**Nó**	**Valor**
u (m)	1248	1,46e-06	1248	1,46e-06
σ_x (kPa)	21	17,469	21	17,469
σ_y (kPa)	2	12,946	2	12,946
τ_{xy} (kPa)	41	356,62	41	356,62
σ_1 (kPa)	2	92,33	2	92,33
σ_2 (kPa)	5	2,34	5	2,34

Verificam-se pelas tabelas que os resultados obtidos através do programa desenvolvido neste trabalho são idênticos aos obtidos pelo ANSYS, comprovando que o *software* foi implementado de maneira correta.

Neste exemplo pode-se perceber que os valores máximos e mínimos dos deslocamentos se deram próximos ao ponto de aplicação da carga. Os valores mínimos das tensões normais e tensões principais, além do ponto de aplicação de carga, se deram próximos aos apoios. Com relação aos valores máximos das tensões normais e tensões principais, estas se deram na borda inferior da chapa por ser a região mais solicitada por esforços de tração da estrutura. Os valores máximo e mínimo das tensões de cisalhamento se deram nos apoios.

Outro ponto importante neste exemplo é com relação a disposição dos elementos nas malhas utilizadas. Como o problema analisado é simétrico, os valores de deslocamentos e tensões obtidos pela malha de elementos *CSQ* seguem uma simetria com relação ao nó solicitado pela carga concentrada. Isso pode ser notado comparando o valor máximo com o valor mínimo das tensões de cisalhamento, que em módulo são iguais. Já os valores obtidos

pela malha de elementos *CST* não possuem simetria, isso ocorre por não se tratar de uma malha simétrica, no qual se mudar a orientação dos elementos triangulares com relação ao eixo de simetria para o lado oposto, a solução irá inverter.

Comparando os resultados obtidos pela malha *CST* com a malha *CSQ*, nota-se que os valores são muito distantes, isso ocorre pelo fato das malhas utilizadas não serem refinadas o suficiente para a convergência da solução. Sendo assim, não é possível concluir se os valores obtidos estão próximos ao valor real, pois, para isso, seria necessário a análise de convergência de malhas.

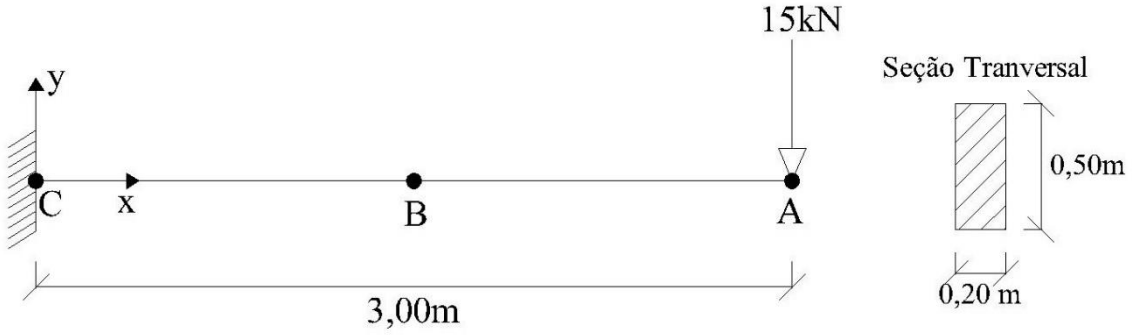
4.3 Exemplo 3

Neste último exemplo foi realizada a análise de uma viga em balanço solicitada por uma carga concentrada em sua extremidade ilustrada pela Figura 18. A viga foi discretizada em elementos de chapa visando estabelecer um estudo sobre as tensões a mesma apresenta.

O objetivo do exemplo foi avaliar a convergência dos resultados obtidos pelo programa desenvolvido, de acordo com o refinamento das malhas utilizando apenas elementos *CST* ou *CSQ*. Os resultados de deslocamentos e tensões serão comparados com a solução analítica do problema.

A Figura 18 contém as informações necessárias para resolução do problema, sendo elas dimensões (comprimento e seção transversal), condições de carregamento e vinculação da viga. Todos os elementos que a compõem apresentam módulo de elasticidade longitudinal E = $35x10^6$ kN/m², coeficiente de Poisson ν = 0,20 e espessura t igual à base da seção transversal da viga.

Figura 18 - Viga em balanço sujeita a uma força concentrada na extremidade.



Fonte: Autoria própria.

Para analisar a convergência de resultados, serão utilizadas oito malhas contendo apenas elementos *CST* e outras oito com apenas elementos *CSQ*. Todas as malhas são regulares, no qual as que possuem apenas elementos *CST*, são compostas por triângulos retângulos.

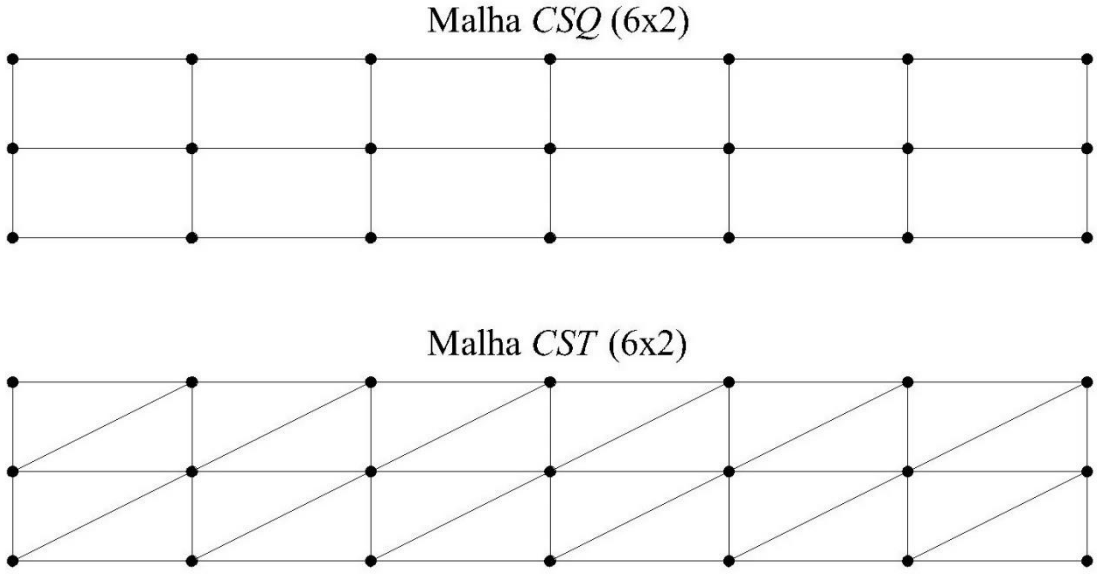
A fim de verificar a eficiência das aproximações do programa, será o utilizado o erro relativo que pode ser calculado conforme a equação (95):

$$e(\%) = \frac{V_o - V_t}{V_t} \cdot 100 \tag{95}$$

No qual V_o corresponde ao valor obtido pelo programa desenvolvido e V_t ao valor utilizado na comparação, resultante da solução analítica.

Para as malhas de elementos *CST*, foram adotados o dobro de elementos comparado com as malhas de elementos *CSQ* que possuem o mesmo número de nós; essa relação foi utilizada por ser necessário no mínimo dois elementos *CST* na forma de triângulo retângulo para abranger a mesma área que um elemento *CSQ* ocuparia. Essa relação está demonstrada na Figura 19.

Figura 19 - Malhas de elementos *CST* e *CSQ* discretizadas com mesmo número de nós.



Fonte: Autoria própria.

O número de nós e elementos de cada malha são apresentados na tabela a seguir, no qual a convenção utilizada para apresentar a disposição das malhas está exemplicado na Figura 19 pelos valores entre parenteses.

Tabela 10 - Número de nós e elementos das malhas para o exemplo 4.

Malha	Elemento *CST*		Elemento *CSQ*	
	Elementos	Nós	Elementos	Nós
1 (2x1)	4	6	2	6
2 (6x2)	24	21	12	21
3 (12x3)	72	52	36	52
4 (20x5)	200	126	100	126
5 (45x15)	1350	736	675	736
6 (80x30)	4800	2511	2400	2511
7 (100x32)	6400	3333	3200	3333
8 (120x40)	9600	4961	4800	4961

Em posse dos valores obtidos de forma analítica referentes ao deslocamento vertical no ponto A e tensões na borda tracionada dos pontos B e C, estes foram comparados com os resultados obtidos pelo programa elaborado utilizando as malhas já especificadas, como mostra os gráficos a seguir:

Gráfico 1 - Convergência de deslocamento em y no ponto A utilizando malhas de elementos *CST*.

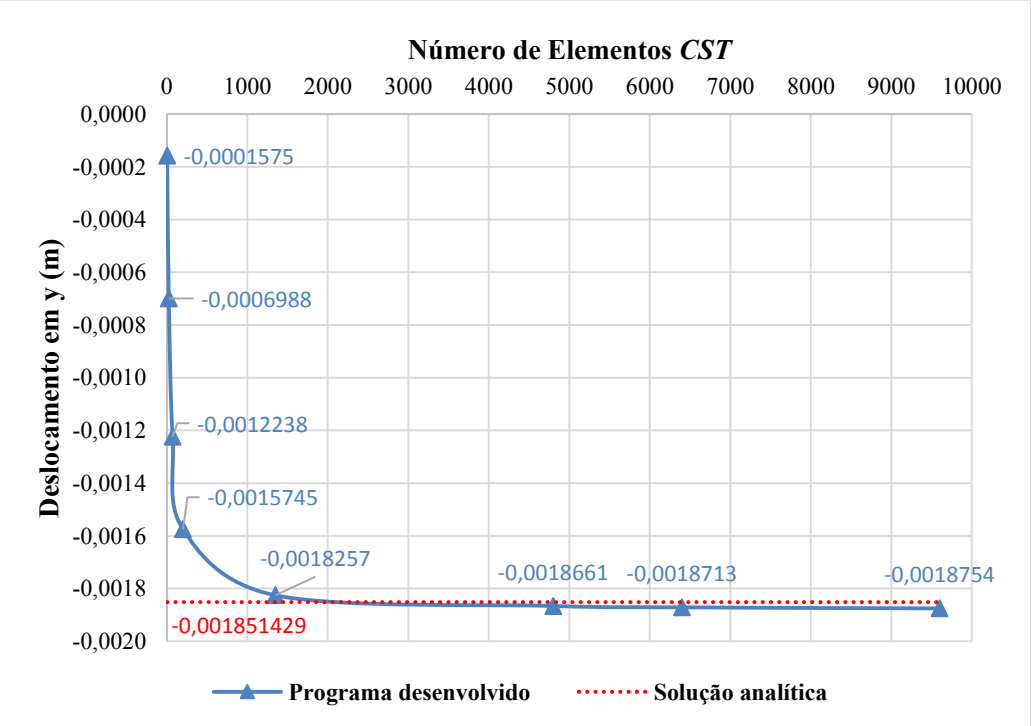


Gráfico 2 - Convergência de deslocamento em y no ponto A utilizando malhas de elementos *CSQ*.

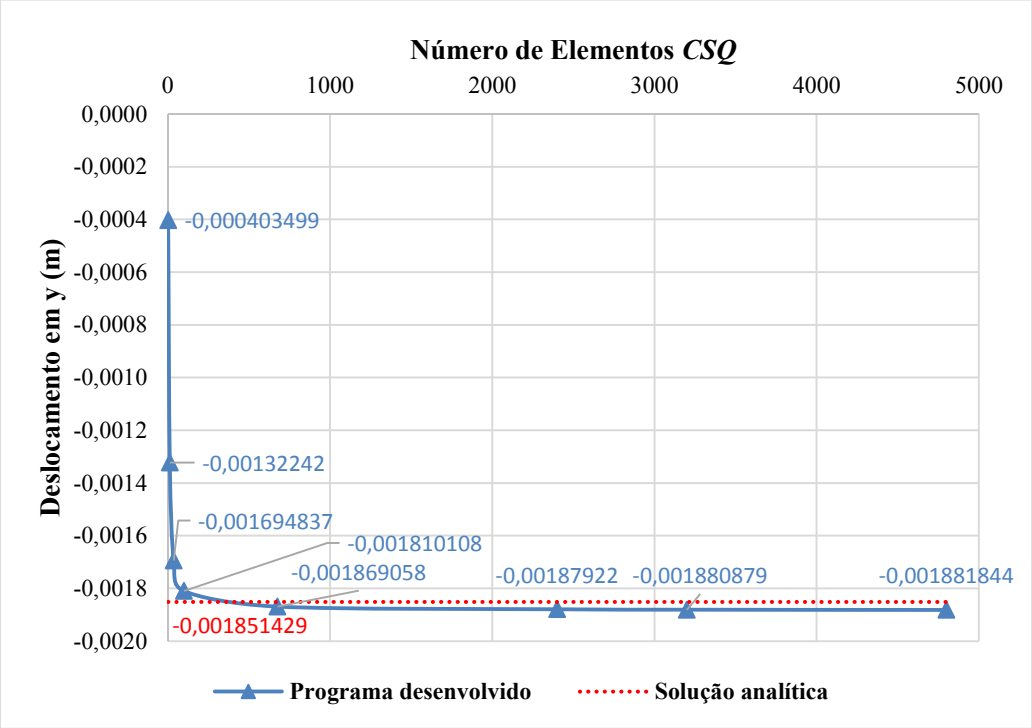


Gráfico 3 - Convergência de tensão em x no ponto B utilizando malhas de elementos *CST*.

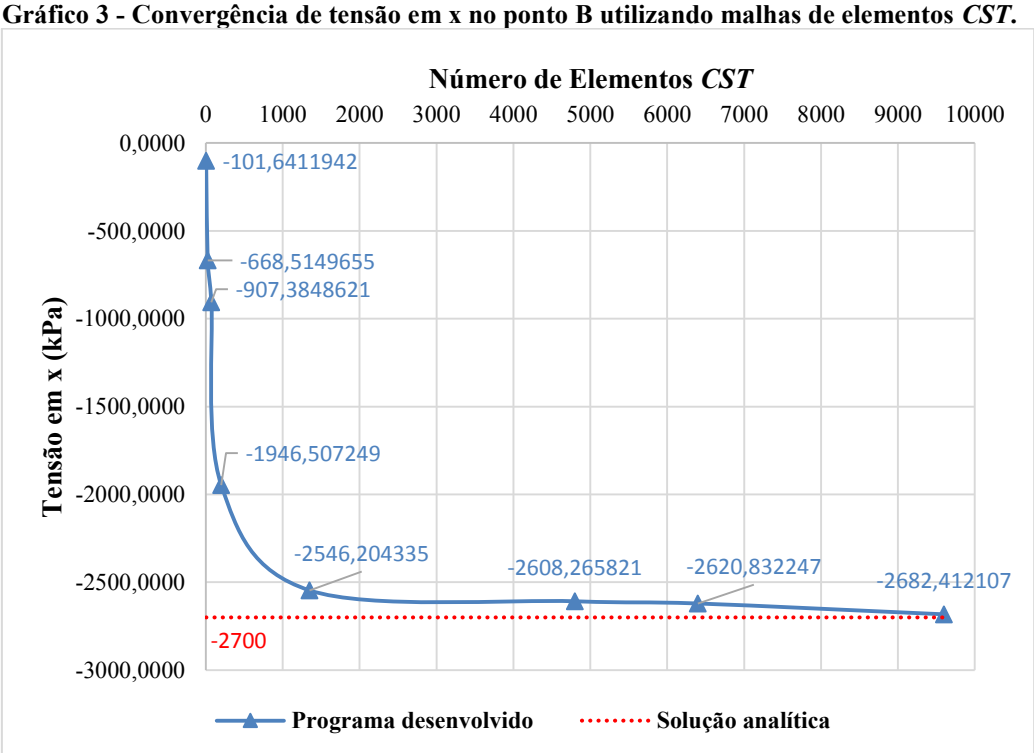


Gráfico 4 - Convergência de tensão em x no ponto B utilizando malhas de elementos *CSQ*.

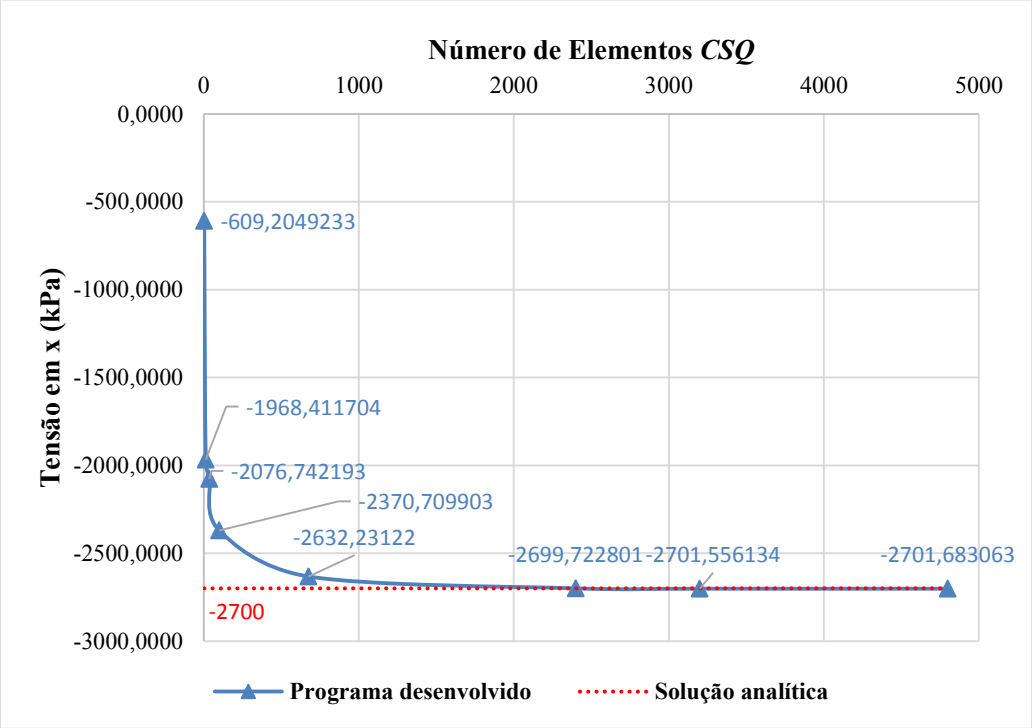


Gráfico 5 - Convergência de tensão em x no ponto C utilizando malhas de elementos *CST*.

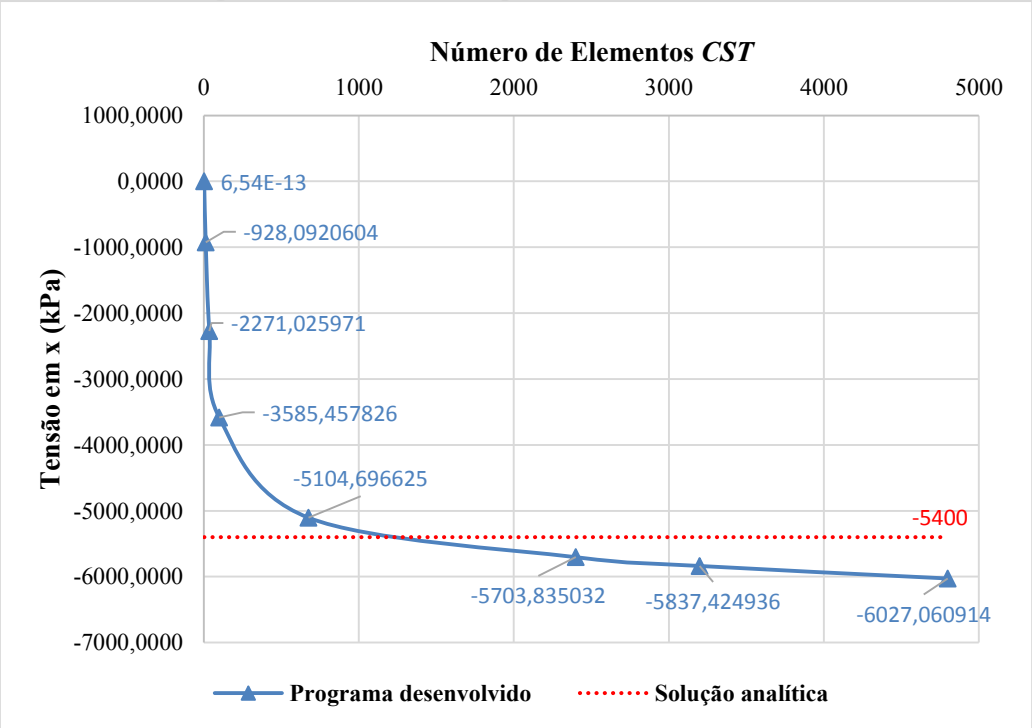
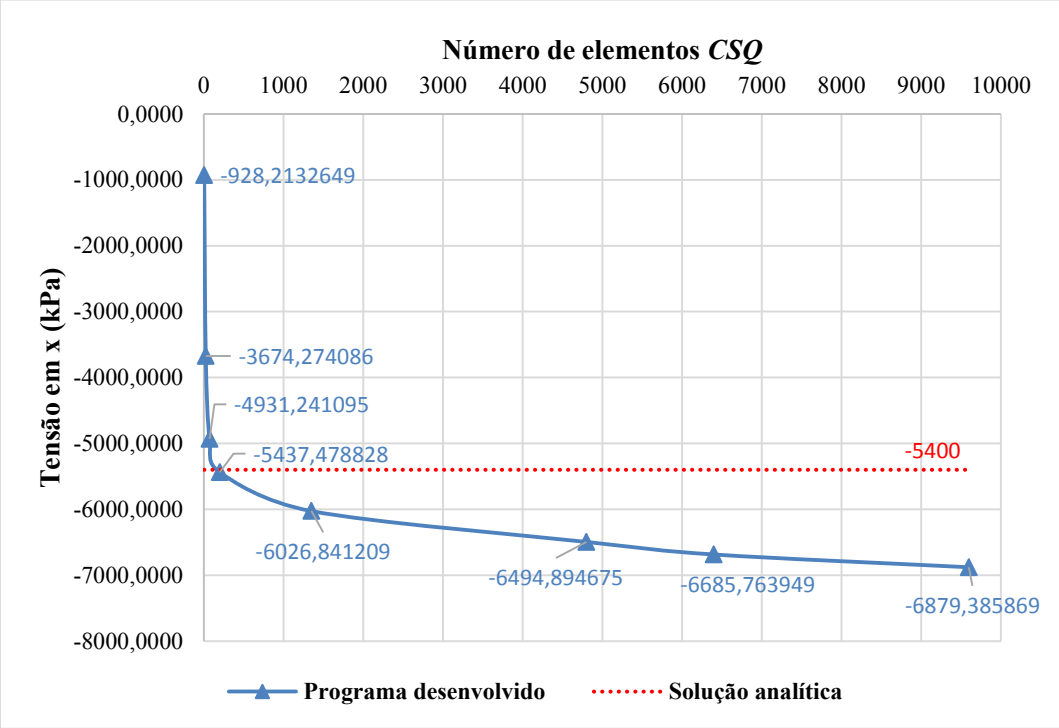


Gráfico 6 - Convergência de tensão em x no ponto C utilizando malhas de elementos *CSQ*.



Os Gráficos 7, 8 e 9 mostram as comparações entre a eficiência das aproximações das malhas de elementos *CST* com as malhas de elementos *CSQ*, por meio dos valores dos erros relativos referentes aos resultados apresentados nos gráficos acima.

Gráfico 7 - Erro relativo dos deslocamentos em y no ponto A.

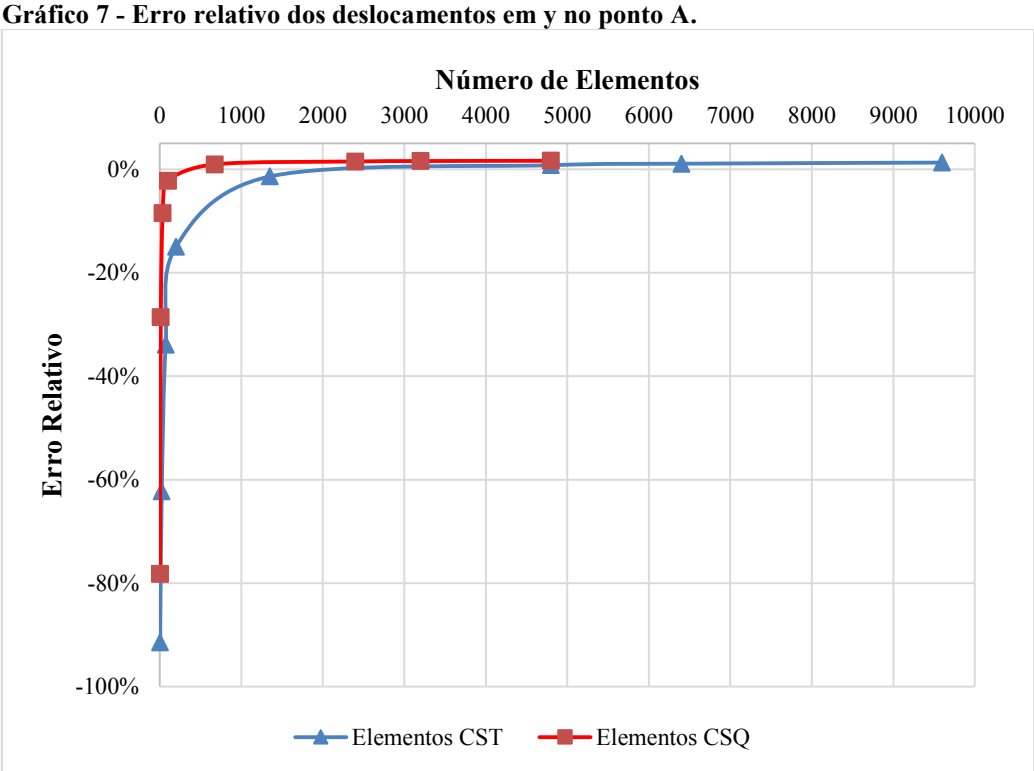


Gráfico 8 - Erro relativo das tensões x no ponto B.

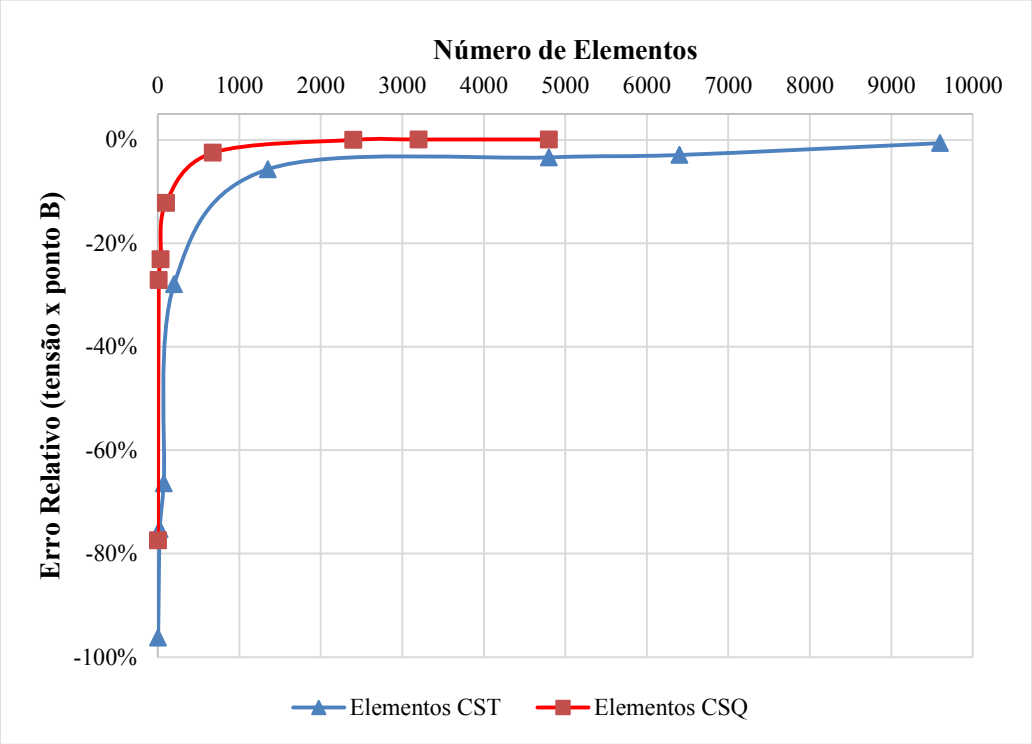
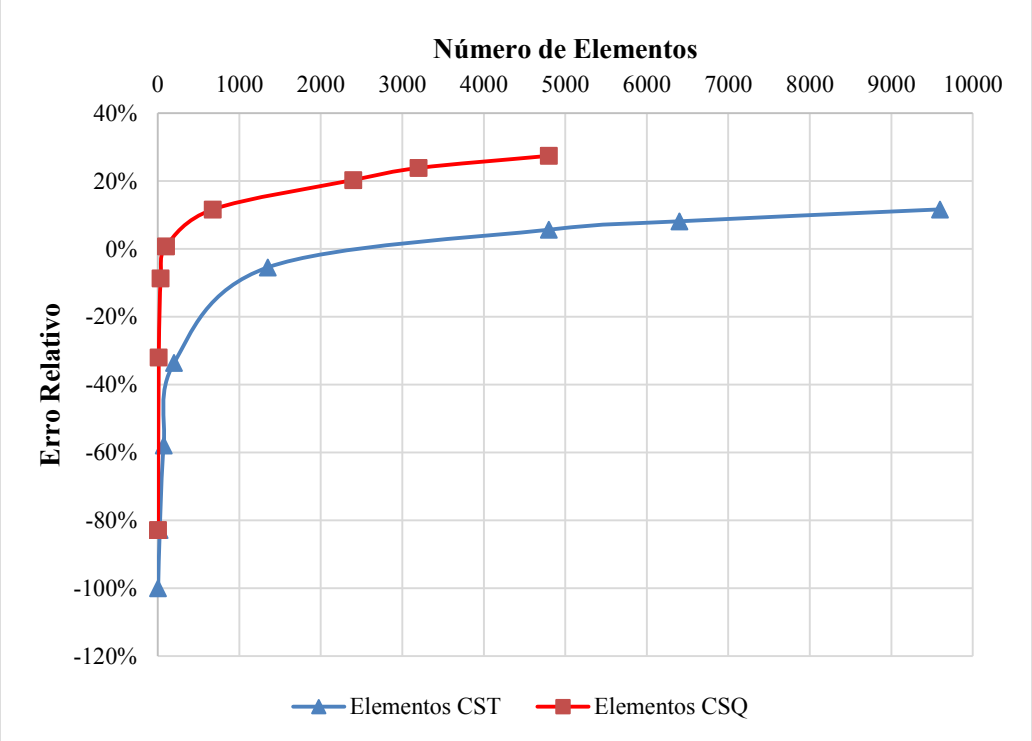


Gráfico 9 - Erro relativo das tensões x no ponto C



Por meio da análise dos gráficos, nota-se que os valores obtidos pelas malhas menos refinadas, como a malha 1 de ambos elementos, possuem um erro relativo muito alto nos três pontos analisados.

Em relação aos valores de deslocamentos em y no ponto B, é possível notar que os resultados apresentaram boa convergência para ambos elementos, porém malhas com elementos *CST* necessitam de maior número de elementos para obter uma boa aproximação do valor real.

Sobre os valores de tensões em x no ponto B, as malhas com elementos *CSQ* apresentaram melhor convergência comparado as malhas com elementos *CST*. De acordo com Gráfico 8, nota-se que só a malha 8 obteve boa aproximação da solução analítica.

Já os valores referentes a tensão em x no ponto C, apenas a malha 4 de elementos *CSQ* obteve convergência consideravél, porém nas malhas seguintes ocorre um distanciamento do valor obtido pela solução analítica. O mesmo acontece com as malhas de elementos *CST*, que apresentou um distanciamento a partir da malha 6. Isso ocorre por haver uma concentração de tensões nesse ponto e, a medida que se refina a malha, o elemento finito se aproxima de um ponto fornecendo tensões que tendem ao infinito.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo principal apresentar a formulação matemática com desenvolvimento de código computacional embasado no Método dos Elementos Finitos para análise elástica linear de chapas utilizando os elementos finitos *Constant Strain Triangle* (*CST*) e *Constant Strain Quadrilateral* (*CSQ*).

A partir do Princípio dos Trabalhos Virtuais, da Lei de Hooke generalizada e outros conceitos pertinentes à Mecânica dos Sólidos e dos Materiais, elaborou-se a matriz de rigidez dos elementos adotados. Vale ressaltar que para aproximar o campo de deslocamentos dos elementos finitos, foram utilizados polinômios de primeiro grau para o elemento *CST* e polinômios de segundo grau para o elemento *CSQ*.

Em posse da matriz de rigidez dos elementos, deu-se início ao processo de implementação computacional. Foram elaboradas sub-rotinas de cálculo seguindo uma ordem pré-estabelecida de execução, o que deu origem a um *software* modular, flexível e de fácil manuseamento, que pode ser utilizado para fins acadêmicos e facilmente adaptado para outras finalidades.

A fim de verificar a validade e precisão do programa desenvolvido, foram apresentados neste trabalho três exemplos de casos de chapas submetidas a diferentes tipos de carregamentos, com propriedades físicas e geométricas distintas. Nos exemplos, os resultados obtidos pelo *software* elaborado foram comparados com os fornecidos pela bibliografia especializada, por *softwares* específicos da área e os obtidos por meio de solução analítica. Assim, pode-se notar que a formulação apresentada foi desenvolvida corretamente e que o *software* implementado é válido para a realização da análise elástica linear de chapas.

As análises mostraram que, para haver convergência de resultados, malhas compostas apenas por elementos *CST* necessitam de maior número de elementos que as compostas apenas por elementos *CSQ*. Sabendo que a precisão dos resultados depende da malha e das funções de aproximação do elemento, conclui-se que isso ocorre pelo fato do elemento *CSQ* possuir funções de aproximação com grau mais elevado do que o elemento *CST*.

Por fim, pode-se propor como sugestão de trabalhos futuros, o uso de elementos finitos que possuem maior número de nós e com funções de aproximação de grau mais elevado, abordando ainda a não-linearidade física e/ou geométrica do elemento estrutural estudado.

REFERÊNCIAS

AZEVEDO, Álvaro F. M. **Método dos elementos finitos**. 1. ed. Porto, 2003. Disponível em: <http://www.fe.up.pt/~alvaro>. Acesso em: 02 mai. 2018.

BEER, Ferdinand P. et al. **Mecânica dos materiais**. 5. ed. Porto Alegre-RS, AMGH editora Ltda, 2011.

COOK, Robert D. et al. **Concepts and applications of finite element analysis**. New York: John Wiley & Sons, 1989.

FISH, Jacob; BELYTSCHKO, Ted. **Um Primeiro Curso em Elementos Finitos**. 1. ed. Rio de Janeiro, RJ: LTC, 2009.

GESUALDO, Francisco A. R. **Método dos elementos finitos**. 2010. 54f. Notas de aula – Programa de Pós-Graduação em Engenharia Civil – Universidade do Federal de Uberlândia, 2010.

HIBBELER, Russell. C. **Resistência dos materiais**. 5. ed. São Paulo: Prentice Hall, 2004.

HUTTON, David V. **Fundamentals of Finite Element Analysis**. 1. ed. The McGraw-Hill, 2004.

LOGAN, Daryl L. **A First Course in the Finite Element Method**. 4. ed. Plateville: Thomson, 2007.

PEREIRA, Orlando J. B. A. **Introdução ao método dos elementos finitos na análise de problemas planos de elasticidade**. Lisboa, 2004. Disponível em: <http://www.civil.ist.utl.pt/ae2/IMEFAPPE.pdf>. Acesso em: 19 mai. 2018.

RIBEIRO, Fernando L. B. **Introdução ao Método dos Elementos Finitos**. Universidade Federal do Rio de Janeiro, 2004.

SANTOS, Gláucia G. M. **Análise sistemática de vigas-parede biapoiadas de concreto armado.** 173 f. Dissertação (Mestrado em Engenharia Civil) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 1999.

SAVASSI, W. **Introdução ao método dos elementos finitos**: em análise linear de estruturas. São Carlos: EESC-USP, 1996.

SORIANO, Humberto Lima. Elementos Finitos – Formulação e Aplicação na Estática e Dinâmica das Estruturas. Rio de Janeiro: Editora Ciência Moderna Ltda., 2009.

SOUZA, Remo M. **O Método dos elementos finitos aplicado ao problema de condução de calor**, NiCAE, Departamento de Engenharia Civil, Universidade Federal do Pará, Belém, 2003. Disponível em: < http://www.ufpa.br/nicae/integrantes/remo_souza/TrabPublicados/Apostilas/ApostilaElementosFinitosNiCAE.pdf> Acesso em: 29 mai. 2018.

SUSSEKIND, J. C. **Curso de análise estrutural**. 6. ed. Porto Alegre-Rio de Janeiro: Editora Globo, 1981.

WAIDEMAM, Leandro. **Análise dinâmica de placas delgadas utilizando elementos finitos triangulares e retangulares**. 168 f. Dissertação (Mestrado em Engenharia Civil) – Universidade Estadual Paulista Júlio de Mesquita Filho, Ilha Solteira, 2004.

WAIDEMAM, Leandro. **Formulação do método dos elementos de contorno para placas enrijecidas considerando-se não-linearidades física e geométrica**. 222 f. Tese (Doutorado em Engenharia de Estruturas) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos. 2008.

APÊNDICE A - CÓDIGO FONTE DO PROGRAMA COMPUTACIONAL APRESENTADO

```
#importa as bibliotecas
import numpy as np
import math

#DESCRICAO
#Num_NO                 Numero de NOS
#Num_Ele_CST            Numero de elementos CST
#Num_Ele_CSQ            Numero de elementos CSQ
#Num_AP                 Numero de apoios
#Num_For_x              Numero de forcas em x
#Num_For_y              Numero de forcas em x
#Coord_NO((:,:))        Numero do NO e coordenadas x e y dos nós
#Lig_Ele_CST((:,:))     Numero do elemento CST, no inicial, no intermediário
                        e seu nó final
#Lig_Ele_CSQ((:,:))     Numero do elemento CSQ, no inicial, no intermediário
                        1, no intermediário 2 e seu nó final
#Tipo_AP((:,:))         Numero do NO, grau de liberdade em x e em y
#Forca_x((:,:))         Numero do NO e valor da forca em x
#Forca_y((:,:))         Numero do NO e valor da forca em y
#Prop_Ele_CST((:,:))    Numero do elemento CST, espessura, Módulo de
                        Elasticidade Longitudinal e Coeficiente de Poisson.
#Prop_Ele_CSQ((:,:))    Numero do elemento CSQ, espessura, Módulo de
                        Elasticidade Longitudinal e Coeficiente de Poisson
#Beta_NO_i(:)           Beta do no incial do elemento CST
#Beta_NO_int(:)         Beta do no intermediario do elemento CST
#Beta_NO_f(:)           Beta do no final do elemento CST
#Gama_NO_i(:)           Gama do no incial do elemento CST
#Gama _NO_int(:)        Gama do no intermediario do elemento CST
#Gama _NO_f(:)          Gama do no final do elemento CST
#A(:)                   Area do elemento CST
#a(:)                   a do elemento CSQ
#b(:)                   b do elemento CSQ
#ME_CST((:,:))          Matriz de rigidez do elemento CST
#ME_CSQ((:,:))          Matriz de rigidez do elemento CSQ
#MR((:,:))              Matriz de rigidez global da estrutura
#MR1((:,:))             Matriz proveniente da Condição de Contorno
#Vetor_Esf(:)           Vetor de cargas nodais
#Desloc_Nod(:)          Vetor de deslocamentos nodais
#Reacoes_AP(:)          Vetor de reações de apoio
#Deform_Nod((:,:))      Deformacoes em x, y e xy
#Tensao_Nod((:,:))      Tensoes em x, y e xy

#Tensoes_Principais((:,:))    Tensoes principais e tensao de cisalhamento
                              maxima
#Angulos_Principais((:,:))    Angulos principais p e s



def abertura_de_arquivos():

    arquivo = open('entrada.txt','r')

    #Leitura de Dados
    primeira_linha = arquivo.readline().split(' ')

    Num_NO, Num_Ele_CST, Num_Ele_CSQ, Num_AP, Num_For_x, Num_For_y = [int(e)
for e in primeira_linha]


    Coord_NO = []
    Lig_Ele_CST = []
```

```
Lig_Ele_CSQ = []
Tipo_AP = []
Forca_x = []
Forca_y = []
Prop_Ele_CST = []
Prop_Ele_CSQ = []



for i in range(0, Num_NO):
    Coord_NO.append(arquivo.readline().rstrip('\n').split(' '))
# Converte a matriz para flutuantes
Coord_NO = [list(map(float, i)) for i in Coord_NO]

if (Num_Ele_CST != 0):
    for i in range(0, Num_Ele_CST):
        Lig_Ele_CST.append(arquivo.readline().rstrip('\n').split(' '))
    # Converte a matriz para inteiros
    Lig_Ele_CST = [list(map(int, i)) for i in Lig_Ele_CST]

if (Num_Ele_CSQ != 0):
    for i in range(0, Num_Ele_CSQ):
        Lig_Ele_CSQ.append(arquivo.readline().rstrip('\n').split(' '))
    # Converte a matriz para inteiros
    Lig_Ele_CSQ = [list(map(int, i)) for i in Lig_Ele_CSQ]
for i in range(0, Num_AP):
    Tipo_AP.append(arquivo.readline().rstrip('\n').split(' '))
# Converte a matriz para inteiros
Tipo_AP = [list(map(int, i)) for i in Tipo_AP]


if (Num_For_x != 0):
    for i in range(0, Num_For_x):
        Forca_x.append(arquivo.readline().rstrip('\n').split(' '))
    # Converte a matriz para flutuantes
    Forca_x = [list(map(float, i)) for i in Forca_x]


if (Num_For_y != 0):
    for i in range(0, Num_For_y):
        Forca_y.append(arquivo.readline().rstrip('\n').split(' '))
    # Converte a matriz para flutuantes
    Forca_y = [list(map(float, i)) for i in Forca_y]

if (Num_Ele_CST != 0):
    for i in range(0, Num_Ele_CST):

        Prop_Ele_CST.append(arquivo.readline().rstrip('\n').split(' '))
    # Converte a matriz para flutuantes
    Prop_Ele_CST = [list(map(float, i)) for i in Prop_Ele_CST]


if (Num_Ele_CSQ != 0):
    for i in range(0, Num_Ele_CSQ):

        Prop_Ele_CSQ.append(arquivo.readline().rstrip('\n').split(' '))
    # Converte a matriz para flutuantes
    Prop_Ele_CSQ = [list(map(float, i)) for i in Prop_Ele_CSQ]
```

```
    return Num_NO, Num_Ele_CST, Num_Ele_CSQ, Num_AP, Num_For_x, Num_For_y,
Coord_NO, Lig_Ele_CST, Lig_Ele_CSQ, Tipo_AP, Forca_x, Forca_y, Prop_Ele_CST,
Prop_Ele_CSQ



def Propriedades_Geometricas_CST(Num_Ele_CST):
    Beta_NO_i = np.zeros(Num_Ele_CST)
    Beta_NO_int = np.zeros(Num_Ele_CST)
    Beta_NO_f = np.zeros(Num_Ele_CST)

    Gama_NO_i = np.zeros(Num_Ele_CST)
    Gama_NO_int = np.zeros(Num_Ele_CST)
    Gama_NO_f = np.zeros(Num_Ele_CST)

    A = np.zeros(Num_Ele_CST)

    for i in range (0, Num_Ele_CST):
        NO_i = Lig_Ele_CST[i][1]
        NO_int = Lig_Ele_CST[i][2]
        NO_f = Lig_Ele_CST[i][3]

        NO_i_x = Coord_NO[(NO_i)-1][1]
        NO_i_y = Coord_NO[(NO_i)-1][2]
        NO_int_x = Coord_NO[(NO_int)-1][1]
        NO_int_y = Coord_NO[(NO_int)-1][2]
        NO_f_x = Coord_NO[(NO_f)-1][1]

        Beta_NO_i[(Lig_Ele_CST[i][0])-1] = (NO_int_y)-(NO_f_y)
        Beta_NO_int[(Lig_Ele_CST[i][0])-1] = (NO_f_y)-(NO_i_y)
        Beta_NO_f[(Lig_Ele_CST[i][0])-1] = (NO_i_y)-(NO_int_y)

        Gama_NO_i[(Lig_Ele_CST[i][0])-1] = (NO_f_x)-(NO_int_x)
        Gama_NO_int[(Lig_Ele_CST[i][0])-1] = (NO_i_x)-(NO_f_x)
        Gama_NO_f[(Lig_Ele_CST[i][0])-1] = (NO_int_x)-(NO_i_x)

        A[(Lig_Ele_CST[i][0])-1]            =            ((NO_i_x*(NO_int_y-
NO_f_y))+(NO_int_x*(NO_f_y-NO_i_y))+(NO_f_x*(NO_i_y-NO_int_y)))/2

    return  Beta_NO_i,  Beta_NO_int,  Beta_NO_f,  Gama_NO_i,  Gama_NO_int,
Gama_NO_f, A



def Propriedades_Geometricas_CSQ(Num_Ele_CSQ):
    a = np.zeros(Num_Ele_CSQ)
    b = np.zeros(Num_Ele_CSQ)

    for i in range (0, Num_Ele_CSQ):
        NO_i = Lig_Ele_CSQ[i][1]
        NO_int_1 = Lig_Ele_CSQ[i][2]
        NO_f = Lig_Ele_CSQ[i][4]

        NO_i_x = Coord_NO[(NO_i)-1][1]
        NO_i_y = Coord_NO[(NO_i)-1][2]
        NO_int_1_x = Coord_NO[(NO_int_1)-1][1]
        NO_f_y = Coord_NO[(NO_f)-1][2]

        a[(Lig_Ele_CSQ[i][0])-1] = (NO_int_1_x - NO_i_x)/2
        b[(Lig_Ele_CSQ[i][0])-1] = (NO_f_y - NO_i_y)/2
```

```
    return a, b


def Matriz_de_Rigidez(Num_NO, Num_Ele_CST, Num_Ele_CSQ):

    ME_CST = np.zeros ((6,6))
    ME_CSQ = np.zeros ((8,8))
    MR = np.zeros(((2*Num_NO),(2*Num_NO)))

    if (Num_Ele_CST != 0):

        for i in range(0, Num_Ele_CST):

            t = Prop_Ele_CST[i][1]
            E = Prop_Ele_CST[i][2]
            Poisson = Prop_Ele_CST[i][3]
            C = (1-(Poisson))/2
            k = (t*E)/(4*A[i]*(1-(Poisson**2)))

            ME_CST[0][0] = k*((Beta_NO_i[i]**2)+(C*(Gama_NO_i[i]**2)))
            ME_CST[0][1] = k*((1+Poisson)/2)*Beta_NO_i[i]*Gama_NO_i[i]
            ME_CST[0][2] =
k*((Beta_NO_i[i]*Beta_NO_int[i])+(C*Gama_NO_i[i]*Gama_NO_int[i]))
            ME_CST[0][3] =
k*((Poisson*Beta_NO_i[i]*Gama_NO_int[i])+(C*Beta_NO_int[i]*Gama_NO_i[i]))
            ME_CST[0][4] =
k*((Beta_NO_i[i]*Beta_NO_f[i])+(C*Gama_NO_i[i]*Gama_NO_f[i]))
            ME_CST[0][5] =
k*((Poisson*Beta_NO_i[i]*Gama_NO_f[i])+(C*Beta_NO_f[i]*Gama_NO_i[i]))

            ME_CST[1][0] = k*((1+Poisson)/2)*Beta_NO_i[i]*Gama_NO_i[i]
            ME_CST[1][1] = k*((Gama_NO_i[i]**2)+(C*(Beta_NO_i[i]**2)))
            ME_CST[1][2] =
k*((Poisson*Beta_NO_int[i]*Gama_NO_i[i])+(C*Beta_NO_i[i]*Gama_NO_int[i]))
            ME_CST[1][3] =
k*((Gama_NO_i[i]*Gama_NO_int[i])+(C*Beta_NO_i[i]*Beta_NO_int[i]))
            ME_CST[1][4] =
k*((Poisson*Beta_NO_f[i]*Gama_NO_i[i])+(C*Beta_NO_i[i]*Gama_NO_f[i]))
            ME_CST[1][5] =
k*((Gama_NO_i[i]*Gama_NO_f[i])+(C*Beta_NO_i[i]*Beta_NO_f[i]))

            ME_CST[2][0] =
k*((Beta_NO_i[i]*Beta_NO_int[i])+(C*Gama_NO_i[i]*Gama_NO_int[i]))
            ME_CST[2][1] =
k*((Poisson*Beta_NO_int[i]*Gama_NO_i[i])+(C*Beta_NO_i[i]*Gama_NO_int[i]))
            ME_CST[2][2] = k*((Beta_NO_int[i]**2)+(C*(Gama_NO_int[i]**2)))
            ME_CST[2][3] = k*((1+Poisson)/2)*Beta_NO_int[i]*Gama_NO_int[i]
            ME_CST[2][4]=
k*((Beta_NO_int[i]*Beta_NO_f[i])+(C*Gama_NO_int[i]*Gama_NO_f[i]))
            ME_CST[2][5] =
k*((Poisson*Beta_NO_int[i]*Gama_NO_f[i])+(C*Beta_NO_f[i]*Gama_NO_int[i]))

            ME_CST[3][0] =
k*((Poisson*Beta_NO_i[i]*Gama_NO_int[i])+(C*Beta_NO_int[i]*Gama_NO_i[i]))
            ME_CST[3][1] =
k*((Gama_NO_i[i]*Gama_NO_int[i])+(C*Beta_NO_i[i]*Beta_NO_int[i]))
            ME_CST[3][2] = k*((1+Poisson)/2)*Beta_NO_int[i]*Gama_NO_int[i]
            ME_CST[3][3] = k*((Gama_NO_int[i]**2)+(C*(Beta_NO_int[i]**2)))
            ME_CST[3][4] =
k*((Poisson*Beta_NO_f[i]*Gama_NO_int[i])+(C*Beta_NO_int[i]*Gama_NO_f[i]))
```

```
            ME_CST[3][5] =
k*((Gama_NO_int[i]*Gama_NO_f[i])+(C*Beta_NO_int[i]*Beta_NO_f[i]))

            ME_CST[4][0] =
k*((Beta_NO_i[i]*Beta_NO_f[i])+(C*Gama_NO_i[i]*Gama_NO_f[i]))
            ME_CST[4][1] =
k*((Poisson*Beta_NO_f[i]*Gama_NO_i[i])+(C*Beta_NO_i[i]*Gama_NO_f[i]))
            ME_CST[4][2] =
k*((Beta_NO_int[i]*Beta_NO_f[i])+(C*Gama_NO_int[i]*Gama_NO_f[i]))
            ME_CST[4][3] =
k*((Poisson*Beta_NO_f[i]*Gama_NO_int[i])+(C*Beta_NO_int[i]*Gama_NO_f[i]))
            ME_CST[4][4] = k*((Beta_NO_f[i]**2)+(C*(Gama_NO_f[i]**2)))
            ME_CST[4][5] = k*((1+Poisson)/2)*Beta_NO_f[i]*Gama_NO_f[i]

            ME_CST[5][0] =
k*((Poisson*Beta_NO_i[i]*Gama_NO_f[i])+(C*Beta_NO_f[i]*Gama_NO_i[i]))
            ME_CST[5][1] =
k*((Gama_NO_i[i]*Gama_NO_f[i])+(C*Beta_NO_i[i]*Beta_NO_f[i]))
            ME_CST[5][2] =
k*((Poisson*Beta_NO_int[i]*Gama_NO_f[i])+(C*Beta_NO_f[i]*Gama_NO_int[i]))
            ME_CST[5][3] =
k*((Gama_NO_int[i]*Gama_NO_f[i])+(C*Beta_NO_int[i]*Beta_NO_f[i]))
            ME_CST[5][4] = k*((1+Poisson)/2)*Beta_NO_f[i]*Gama_NO_f[i]
            ME_CST[5][5] = k*((Gama_NO_f[i]**2)+(C*(Beta_NO_f[i]**2)))

            No_i = Lig_Ele_CST[i][1]
            No_int = Lig_Ele_CST[i][2]
            No_f = Lig_Ele_CST[i][3]

            MR[((2*(No_i))-2)][((2*(No_i))-2)] = MR[((2*(No_i))-
2)][((2*(No_i))-2)] + ME_CST[0][0]
            MR[((2*(No_i))-2)][((2*(No_i))-1)] = MR[((2*(No_i))-
2)][((2*(No_i))-1)] + ME_CST[0][1]
            MR[((2*(No_i))-2)][((2*(No_int))-2)] = MR[((2*(No_i))-
2)][((2*(No_int))-2)] + ME_CST[0][2]
            MR[((2*(No_i))-2)][((2*(No_int))-1)] = MR[((2*(No_i))-
2)][((2*(No_int))-1)] + ME_CST[0][3]
            MR[((2*(No_i))-2)][((2*(No_f))-2)] = MR[((2*(No_i))-
2)][((2*(No_f))-2)] + ME_CST[0][4]
            MR[((2*(No_i))-2)][((2*(No_f))-1)] = MR[((2*(No_i))-
2)][((2*(No_f))-1)] + ME_CST[0][5]

            MR[((2*(No_i))-1)][((2*(No_i))-2)] = MR[((2*(No_i))-
1)][((2*(No_i))-2)] + ME_CST[1][0]
            MR[((2*(No_i))-1)][((2*(No_i))-1)] = MR[((2*(No_i))-
1)][((2*(No_i))-1)] + ME_CST[1][1]
            MR[((2*(No_i))-1)][((2*(No_int))-2)] = MR[((2*(No_i))-
1)][((2*(No_int))-2)] + ME_CST[1][2]
            MR[((2*(No_i))-1)][((2*(No_int))-1)] = MR[((2*(No_i))-
1)][((2*(No_int))-1)] + ME_CST[1][3]
            MR[((2*(No_i))-1)][((2*(No_f))-2)] = MR[((2*(No_i))-
1)][((2*(No_f))-2)] + ME_CST[1][4]
            MR[((2*(No_i))-1)][((2*(No_f))-1)] = MR[((2*(No_i))-
1)][((2*(No_f))-1)] + ME_CST[1][5]

            MR[((2*(No_int))-2)][((2*(No_i))-2)] = MR[((2*(No_int))-
2)][((2*(No_i))-2)] + ME_CST[2][0]
            MR[((2*(No_int))-2)][((2*(No_i))-1)] = MR[((2*(No_int))-
2)][((2*(No_i))-1)] + ME_CST[2][1]
            MR[((2*(No_int))-2)][((2*(No_int))-2)] = MR[((2*(No_int))-
2)][((2*(No_int))-2)] + ME_CST[2][2]
```

```
            MR[((2*(No_int))-2)][((2*(No_int))-1)] = MR[((2*(No_int))-
2)][((2*(No_int))-1)] + ME_CST[2][3]
            MR[((2*(No_int))-2)][((2*(No_f))-2)] = MR[((2*(No_int))-
2)][((2*(No_f))-2)] + ME_CST[2][4]
            MR[((2*(No_int))-2)][((2*(No_f))-1)] = MR[((2*(No_int))-
2)][((2*(No_f))-1)] + ME_CST[2][5]

            MR[((2*(No_int))-1)][((2*(No_i))-2)] = MR[((2*(No_int))-
1)][((2*(No_i))-2)] + ME_CST[3][0]
            MR[((2*(No_int))-1)][((2*(No_i))-1)] = MR[((2*(No_int))-
1)][((2*(No_i))-1)] + ME_CST[3][1]
            MR[((2*(No_int))-1)][((2*(No_int))-2)] = MR[((2*(No_int))-
1)][((2*(No_int))-2)] + ME_CST[3][2]
            MR[((2*(No_int))-1)][((2*(No_int))-1)] = MR[((2*(No_int))-
1)][((2*(No_int))-1)] + ME_CST[3][3]
            MR[((2*(No_int))-1)][((2*(No_f))-2)] = MR[((2*(No_int))-
1)][((2*(No_f))-2)] + ME_CST[3][4]
            MR[((2*(No_int))-1)][((2*(No_f))-1)] = MR[((2*(No_int))-
1)][((2*(No_f))-1)] + ME_CST[3][5]

            MR[((2*(No_f))-2)][((2*(No_i))-2)] = MR[((2*(No_f))-
2)][((2*(No_i))-2)] + ME_CST[4][0]
            MR[((2*(No_f))-2)][((2*(No_i))-1)] = MR[((2*(No_f))-
2)][((2*(No_i))-1)] + ME_CST[4][1]
            MR[((2*(No_f))-2)][((2*(No_int))-2)] = MR[((2*(No_f))-
2)][((2*(No_int))-2)] + ME_CST[4][2]
            MR[((2*(No_f))-2)][((2*(No_int))-1)] = MR[((2*(No_f))-
2)][((2*(No_int))-1)] + ME_CST[4][3]
            MR[((2*(No_f))-2)][((2*(No_f))-2)] = MR[((2*(No_f))-
2)][((2*(No_f))-2)] + ME_CST[4][4]
            MR[((2*(No_f))-2)][((2*(No_f))-1)] = MR[((2*(No_f))-
2)][((2*(No_f))-1)] + ME_CST[4][5]

            MR[((2*(No_f))-1)][((2*(No_i))-2)] = MR[((2*(No_f))-
1)][((2*(No_i))-2)] + ME_CST[5][0]
            MR[((2*(No_f))-1)][((2*(No_i))-1)] = MR[((2*(No_f))-
1)][((2*(No_i))-1)] + ME_CST[5][1]
            MR[((2*(No_f))-1)][((2*(No_int))-2)] = MR[((2*(No_f))-
1)][((2*(No_int))-2)] + ME_CST[5][2]
            MR[((2*(No_f))-1)][((2*(No_int))-1)] = MR[((2*(No_f))-
1)][((2*(No_int))-1)] + ME_CST[5][3]
            MR[((2*(No_f))-1)][((2*(No_f))-2)] = MR[((2*(No_f))-
1)][((2*(No_f))-2)] + ME_CST[5][4]
            MR[((2*(No_f))-1)][((2*(No_f))-1)] = MR[((2*(No_f))-
1)][((2*(No_f))-1)] + ME_CST[5][5]

    if (Num_Ele_CSQ != 0):


        for i in range(0, Num_Ele_CSQ):

            t = Prop_Ele_CSQ[i][1]
            E = Prop_Ele_CSQ[i][2]
            Poisson = Prop_Ele_CSQ[i][3]
            k  =  (t*E)/(1-(Poisson**2))#produto  das  constantes  que  foram
colocadas para fora da integral

            ME_CSQ[0][0]                =                k*((2*(b[i]**2)+((1-
Poisson)*(a[i]**2)))/(6*a[i]*b[i]))
            ME_CSQ[0][1] = k*((1+Poisson)/8)
```

```
            ME_CSQ[0][2]                =              k*((-4*(b[i]**2)+((1-
Poisson)*(a[i]**2)))/(12*a[i]*b[i]))
            ME_CSQ[0][3] = k*((-1+(3*Poisson))/8)
            ME_CSQ[0][4]                =               k*((-2*(b[i]**2)+((-
1+Poisson)*(a[i]**2)))/(12*a[i]*b[i]))
            ME_CSQ[0][5] = k*((-1-Poisson)/8)
            ME_CSQ[0][6]                  =                k*(((b[i]**2)+((-
1+Poisson)*(a[i]**2)))/(6*a[i]*b[i]))
            ME_CSQ[0][7] = k*((1-(3*Poisson))/8)

            ME_CSQ[1][0] = k*((1+Poisson)/8)
            ME_CSQ[1][1]                =               k*((2*(a[i]**2)+((1-
Poisson)*(b[i]**2)))/(6*a[i]*b[i]))
            ME_CSQ[1][2] = k*((1-(3*Poisson))/8)
            ME_CSQ[1][3]                  =                k*(((a[i]**2)+((-
1+Poisson)*(b[i]**2)))/(6*a[i]*b[i]))
            ME_CSQ[1][4] = k*((-1-Poisson)/8)
            ME_CSQ[1][5]                =               k*((-2*(a[i]**2)+((-
1+Poisson)*(b[i]**2)))/(12*a[i]*b[i]))
            ME_CSQ[1][6] = k*((-1+(3*Poisson))/8)
            ME_CSQ[1][7]                =              k*((-4*(a[i]**2)+((1-
Poisson)*(b[i]**2)))/(12*a[i]*b[i]))

            ME_CSQ[2][0]                =              k*((-4*(b[i]**2)+((1-
Poisson)*(a[i]**2)))/(12*a[i]*b[i]))
            ME_CSQ[2][1] = k*((1-(3*Poisson))/8)
            ME_CSQ[2][2]                =               k*((2*(b[i]**2)+((1-
Poisson)*(a[i]**2)))/(6*a[i]*b[i]))
            ME_CSQ[2][3] = k*((-1-Poisson)/8)
            ME_CSQ[2][4]                  =                k*(((b[i]**2)+((-
1+Poisson)*(a[i]**2)))/(6*a[i]*b[i]))
            ME_CSQ[2][5] = k*((-1+(3*Poisson))/8)
            ME_CSQ[2][6]                =               k*((-2*(b[i]**2)+((-
1+Poisson)*(a[i]**2)))/(12*a[i]*b[i]))
            ME_CSQ[2][7] = k*((1+Poisson)/8)

            ME_CSQ[3][0] = k*((-1+(3*Poisson))/8)
            ME_CSQ[3][1]                  =                k*(((a[i]**2)+((-
1+Poisson)*(b[i]**2)))/(6*a[i]*b[i]))
            ME_CSQ[3][2] = k*((-1-Poisson)/8)
            ME_CSQ[3][3]                =               k*((2*(a[i]**2)+((1-
Poisson)*(b[i]**2)))/(6*a[i]*b[i]))
            ME_CSQ[3][4] = k*((1-(3*Poisson))/8)
            ME_CSQ[3][5]                =              k*((-4*(a[i]**2)+((1-
Poisson)*(b[i]**2)))/(12*a[i]*b[i]))
            ME_CSQ[3][6] = k*((1+Poisson)/8)
            ME_CSQ[3][7]                =               k*((-2*(a[i]**2)+((-
1+Poisson)*(b[i]**2)))/(12*a[i]*b[i]))

            ME_CSQ[4][0]                =               k*((-2*(b[i]**2)+((-
1+Poisson)*(a[i]**2)))/(12*a[i]*b[i]))
            ME_CSQ[4][1] = k*((-1-Poisson)/8)
            ME_CSQ[4][2]                  =                k*(((b[i]**2)+((-
1+Poisson)*(a[i]**2)))/(6*a[i]*b[i]))
            ME_CSQ[4][3] = k*((1-(3*Poisson))/8)
            ME_CSQ[4][4]                =               k*((2*(b[i]**2)+((1-
Poisson)*(a[i]**2)))/(6*a[i]*b[i]))
            ME_CSQ[4][5] = k*((1+Poisson)/8)
            ME_CSQ[4][6]                =              k*((-4*(b[i]**2)+((1-
Poisson)*(a[i]**2)))/(12*a[i]*b[i]))
            ME_CSQ[4][7] = k*((-1+(3*Poisson))/8)
```

```
            ME_CSQ[5][0] = k*((-1-Poisson)/8)
            ME_CSQ[5][1]                =                k*((-2*(a[i]**2)+((-
1+Poisson)*(b[i]**2)))/(12*a[i]*b[i]))
            ME_CSQ[5][2] = k*((-1+(3*Poisson))/8)
            ME_CSQ[5][3]                =               k*((-4*(a[i]**2)+((1-
Poisson)*(b[i]**2)))/(12*a[i]*b[i]))
            ME_CSQ[5][4] = k*((1+Poisson)/8)
            ME_CSQ[5][5]                =                k*((2*(a[i]**2)+((1-
Poisson)*(b[i]**2)))/(6*a[i]*b[i]))
            ME_CSQ[5][6] = k*((1-(3*Poisson))/8)
            ME_CSQ[5][7]                 =                  k*(((a[i]**2)+((-
1+Poisson)*(b[i]**2)))/(6*a[i]*b[i]))

            ME_CSQ[6][0]                 =                  k*(((b[i]**2)+((-
1+Poisson)*(a[i]**2)))/(6*a[i]*b[i]))
            ME_CSQ[6][1] = k*((-1+(3*Poisson))/8)
            ME_CSQ[6][2]                =                k*((-2*(b[i]**2)+((-
1+Poisson)*(a[i]**2)))/(12*a[i]*b[i]))
            ME_CSQ[6][3] = k*((1+Poisson)/8)
            ME_CSQ[6][4]                =               k*((-4*(b[i]**2)+((1-
Poisson)*(a[i]**2)))/(12*a[i]*b[i]))
            ME_CSQ[6][5] = k*((1-(3*Poisson))/8)
            ME_CSQ[6][6]                =                k*((2*(b[i]**2)+((1-
Poisson)*(a[i]**2)))/(6*a[i]*b[i]))
            ME_CSQ[6][7] = k*((-1-Poisson)/8)

            ME_CSQ[7][0] = k*((1-(3*Poisson))/8)
            ME_CSQ[7][1]                =               k*((-4*(a[i]**2)+((1-
Poisson)*(b[i]**2)))/(12*a[i]*b[i]))
            ME_CSQ[7][2] = k*((1+Poisson)/8)
            ME_CSQ[7][3]                 =                k*((-2*(a[i]**2)+((-
1+Poisson)*(b[i]**2)))/(12*a[i]*b[i]))
            ME_CSQ[7][4] = k*((-1+(3*Poisson))/8)
            ME_CSQ[7][5]                  =                  k*(((a[i]**2)+((-
1+Poisson)*(b[i]**2)))/(6*a[i]*b[i]))
            ME_CSQ[7][6] = k*((-1-Poisson)/8)
            ME_CSQ[7][7]                =                k*((2*(a[i]**2)+((1-
Poisson)*(b[i]**2)))/(6*a[i]*b[i]))

            No_i = Lig_Ele_CSQ[i][1]
            No_int_1 = Lig_Ele_CSQ[i][2]
            No_int_2 = Lig_Ele_CSQ[i][3]
            No_f = Lig_Ele_CSQ[i][4]

            MR[((2*(No_i))-2)][((2*(No_i))-2)]       =       MR[((2*(No_i))-
2)][((2*(No_i))-2)] + ME_CSQ[0][0]
            MR[((2*(No_i))-2)][((2*(No_i))-1)]       =       MR[((2*(No_i))-
2)][((2*(No_i))-1)] + ME_CSQ[0][1]
            MR[((2*(No_i))-2)][((2*(No_int_1))-2)]     =     MR[((2*(No_i))-
2)][((2*(No_int_1))-2)] + ME_CSQ[0][2]
            MR[((2*(No_i))-2)][((2*(No_int_1))-1)]     =     MR[((2*(No_i))-
2)][((2*(No_int_1))-1)] + ME_CSQ[0][3]
            MR[((2*(No_i))-2)][((2*(No_int_2))-2)]     =     MR[((2*(No_i))-
2)][((2*(No_int_2))-2)] + ME_CSQ[0][4]
            MR[((2*(No_i))-2)][((2*(No_int_2))-1)]     =     MR[((2*(No_i))-
2)][((2*(No_int_2))-1)] + ME_CSQ[0][5]
            MR[((2*(No_i))-2)][((2*(No_f))-2)]       =       MR[((2*(No_i))-
2)][((2*(No_f))-2)] + ME_CSQ[0][6]
            MR[((2*(No_i))-2)][((2*(No_f))-1)]       =       MR[((2*(No_i))-
2)][((2*(No_f))-1)] + ME_CSQ[0][7]
```

```
            MR[((2*(No_i))-1)][((2*(No_i))-2)]       =       MR[((2*(No_i))-
1)][((2*(No_i))-2)] + ME_CSQ[1][0]
            MR[((2*(No_i))-1)][((2*(No_i))-1)]       =       MR[((2*(No_i))-
1)][((2*(No_i))-1)] + ME_CSQ[1][1]
            MR[((2*(No_i))-1)][((2*(No_int_1))-2)]     =     MR[((2*(No_i))-
1)][((2*(No_int_1))-2)] + ME_CSQ[1][2]
            MR[((2*(No_i))-1)][((2*(No_int_1))-1)]     =     MR[((2*(No_i))-
1)][((2*(No_int_1))-1)] + ME_CSQ[1][3]
            MR[((2*(No_i))-1)][((2*(No_int_2))-2)]     =     MR[((2*(No_i))-
1)][((2*(No_int_2))-2)] + ME_CSQ[1][4]
            MR[((2*(No_i))-1)][((2*(No_int_2))-1)]     =     MR[((2*(No_i))-
1)][((2*(No_int_2))-1)] + ME_CSQ[1][5]
            MR[((2*(No_i))-1)][((2*(No_f))-2)]       =       MR[((2*(No_i))-
1)][((2*(No_f))-2)] + ME_CSQ[1][6]
            MR[((2*(No_i))-1)][((2*(No_f))-1)]       =       MR[((2*(No_i))-
1)][((2*(No_f))-1)] + ME_CSQ[1][7]

            MR[((2*(No_int_1))-2)][((2*(No_i))-2)]   =   MR[((2*(No_int_1))-
2)][((2*(No_i))-2)] + ME_CSQ[2][0]
            MR[((2*(No_int_1))-2)][((2*(No_i))-1)]   =   MR[((2*(No_int_1))-
2)][((2*(No_i))-1)] + ME_CSQ[2][1]
            MR[((2*(No_int_1))-2)][((2*(No_int_1))-2)] = MR[((2*(No_int_1))-
2)][((2*(No_int_1))-2)] + ME_CSQ[2][2]
            MR[((2*(No_int_1))-2)][((2*(No_int_1))-1)] = MR[((2*(No_int_1))-
2)][((2*(No_int_1))-1)] + ME_CSQ[2][3]
            MR[((2*(No_int_1))-2)][((2*(No_int_2))-2)] = MR[((2*(No_int_1))-
2)][((2*(No_int_2))-2)] + ME_CSQ[2][4]
            MR[((2*(No_int_1))-2)][((2*(No_int_2))-1)] = MR[((2*(No_int_1))-
2)][((2*(No_int_2))-1)] + ME_CSQ[2][5]
            MR[((2*(No_int_1))-2)][((2*(No_f))-2)]   =   MR[((2*(No_int_1))-
2)][((2*(No_f))-2)] + ME_CSQ[2][6]
            MR[((2*(No_int_1))-2)][((2*(No_f))-1)]   =   MR[((2*(No_int_1))-
2)][((2*(No_f))-1)] + ME_CSQ[2][7]

            MR[((2*(No_int_1))-1)][((2*(No_i))-2)]   =   MR[((2*(No_int_1))-
1)][((2*(No_i))-2)] + ME_CSQ[3][0]
            MR[((2*(No_int_1))-1)][((2*(No_i))-1)]   =   MR[((2*(No_int_1))-
1)][((2*(No_i))-1)] + ME_CSQ[3][1]
            MR[((2*(No_int_1))-1)][((2*(No_int_1))-2)] = MR[((2*(No_int_1))-
1)][((2*(No_int_1))-2)] + ME_CSQ[3][2]
            MR[((2*(No_int_1))-1)][((2*(No_int_1))-1)] = MR[((2*(No_int_1))-
1)][((2*(No_int_1))-1)] + ME_CSQ[3][3]
            MR[((2*(No_int_1))-1)][((2*(No_int_2))-2)] = MR[((2*(No_int_1))-
1)][((2*(No_int_2))-2)] + ME_CSQ[3][4]
            MR[((2*(No_int_1))-1)][((2*(No_int_2))-1)] = MR[((2*(No_int_1))-
1)][((2*(No_int_2))-1)] + ME_CSQ[3][5]
            MR[((2*(No_int_1))-1)][((2*(No_f))-2)]   =   MR[((2*(No_int_1))-
1)][((2*(No_f))-2)] + ME_CSQ[3][6]
            MR[((2*(No_int_1))-1)][((2*(No_f))-1)]   =   MR[((2*(No_int_1))-
1)][((2*(No_f))-1)] + ME_CSQ[3][7]

            MR[((2*(No_int_2))-2)][((2*(No_i))-2)]   =   MR[((2*(No_int_2))-
2)][((2*(No_i))-2)] + ME_CSQ[4][0]
            MR[((2*(No_int_2))-2)][((2*(No_i))-1)]   =   MR[((2*(No_int_2))-
2)][((2*(No_i))-1)] + ME_CSQ[4][1]
            MR[((2*(No_int_2))-2)][((2*(No_int_1))-2)] = MR[((2*(No_int_2))-
2)][((2*(No_int_1))-2)] + ME_CSQ[4][2]
            MR[((2*(No_int_2))-2)][((2*(No_int_1))-1)] = MR[((2*(No_int_2))-
2)][((2*(No_int_1))-1)] + ME_CSQ[4][3]
```

```
            MR[((2*(No_int_2))-2)][((2*(No_int_2))-2)] = MR[((2*(No_int_2))-
2)][((2*(No_int_2))-2)] + ME_CSQ[4][4]
            MR[((2*(No_int_2))-2)][((2*(No_int_2))-1)] = MR[((2*(No_int_2))-
2)][((2*(No_int_2))-1)] + ME_CSQ[4][5]
            MR[((2*(No_int_2))-2)][((2*(No_f))-2)]   =   MR[((2*(No_int_2))-
2)][((2*(No_f))-2)] + ME_CSQ[4][6]
            MR[((2*(No_int_2))-2)][((2*(No_f))-1)]   =   MR[((2*(No_int_2))-
2)][((2*(No_f))-1)] + ME_CSQ[4][7]

            MR[((2*(No_int_2))-1)][((2*(No_i))-2)]   =   MR[((2*(No_int_2))-
1)][((2*(No_i))-2)] + ME_CSQ[5][0]
            MR[((2*(No_int_2))-1)][((2*(No_i))-1)]   =   MR[((2*(No_int_2))-
1)][((2*(No_i))-1)] + ME_CSQ[5][1]
            MR[((2*(No_int_2))-1)][((2*(No_int_1))-2)] = MR[((2*(No_int_2))-
1)][((2*(No_int_1))-2)] + ME_CSQ[5][2]
            MR[((2*(No_int_2))-1)][((2*(No_int_1))-1)] = MR[((2*(No_int_2))-
1)][((2*(No_int_1))-1)] + ME_CSQ[5][3]
            MR[((2*(No_int_2))-1)][((2*(No_int_2))-2)] = MR[((2*(No_int_2))-
1)][((2*(No_int_2))-2)] + ME_CSQ[5][4]
            MR[((2*(No_int_2))-1)][((2*(No_int_2))-1)] = MR[((2*(No_int_2))-
1)][((2*(No_int_2))-1)] + ME_CSQ[5][5]
            MR[((2*(No_int_2))-1)][((2*(No_f))-2)]   =   MR[((2*(No_int_2))-
1)][((2*(No_f))-2)] + ME_CSQ[5][6]
            MR[((2*(No_int_2))-1)][((2*(No_f))-1)]   =   MR[((2*(No_int_2))-
1)][((2*(No_f))-1)] + ME_CSQ[5][7]

            MR[((2*(No_f))-2)][((2*(No_i))-2)]       =       MR[((2*(No_f))-
2)][((2*(No_i))-2)] + ME_CSQ[6][0]
            MR[((2*(No_f))-2)][((2*(No_i))-1)]       =       MR[((2*(No_f))-
2)][((2*(No_i))-1)] + ME_CSQ[6][1]
            MR[((2*(No_f))-2)][((2*(No_int_1))-2)]     =     MR[((2*(No_f))-
2)][((2*(No_int_1))-2)] + ME_CSQ[6][2]
            MR[((2*(No_f))-2)][((2*(No_int_1))-1)]     =     MR[((2*(No_f))-
2)][((2*(No_int_1))-1)] + ME_CSQ[6][3]
            MR[((2*(No_f))-2)][((2*(No_int_2))-2)]     =     MR[((2*(No_f))-
2)][((2*(No_int_2))-2)] + ME_CSQ[6][4]
            MR[((2*(No_f))-2)][((2*(No_int_2))-1)]     =     MR[((2*(No_f))-
2)][((2*(No_int_2))-1)] + ME_CSQ[6][5]
            MR[((2*(No_f))-2)][((2*(No_f))-2)]       =       MR[((2*(No_f))-
2)][((2*(No_f))-2)] + ME_CSQ[6][6]
            MR[((2*(No_f))-2)][((2*(No_f))-1)]       =       MR[((2*(No_f))-
2)][((2*(No_f))-1)] + ME_CSQ[6][7]

            MR[((2*(No_f))-1)][((2*(No_i))-2)]       =       MR[((2*(No_f))-
1)][((2*(No_i))-2)] + ME_CSQ[7][0]
            MR[((2*(No_f))-1)][((2*(No_i))-1)]       =       MR[((2*(No_f))-
1)][((2*(No_i))-1)] + ME_CSQ[7][1]
            MR[((2*(No_f))-1)][((2*(No_int_1))-2)]     =     MR[((2*(No_f))-
1)][((2*(No_int_1))-2)] + ME_CSQ[7][2]
            MR[((2*(No_f))-1)][((2*(No_int_1))-1)]     =     MR[((2*(No_f))-
1)][((2*(No_int_1))-1)] + ME_CSQ[7][3]
            MR[((2*(No_f))-1)][((2*(No_int_2))-2)]     =     MR[((2*(No_f))-
1)][((2*(No_int_2))-2)] + ME_CSQ[7][4]
            MR[((2*(No_f))-1)][((2*(No_int_2))-1)]     =     MR[((2*(No_f))-
1)][((2*(No_int_2))-1)] + ME_CSQ[7][5]
            MR[((2*(No_f))-1)][((2*(No_f))-2)]       =       MR[((2*(No_f))-
1)][((2*(No_f))-2)] + ME_CSQ[7][6]
            MR[((2*(No_f))-1)][((2*(No_f))-1)]       =       MR[((2*(No_f))-
1)][((2*(No_f))-1)] + ME_CSQ[7][7]

    return MR
```

```
def Vetor_Esforcos():

    Vetor_Esf = np.zeros(2*Num_NO)

#transforma toda a matriz para inteiros
    Forca_x_trunc = np.trunc(Forca_x)
    Forca_x_int = Forca_x_trunc.astype(int)

#transforma toda a matriz para inteiros
    Forca_y_trunc = np.trunc(Forca_y)
    Forca_y_int = Forca_y_trunc.astype(int)

    for i in range(0, Num_NO):
        if (Num_For_x != 0 and i < Num_For_x):
            Vetor_Esf[(2*Forca_x_int[i][0])-2] = Forca_x[i][1]

        if (Num_For_y != 0 and i < Num_For_y):
            Vetor_Esf[(2*Forca_y_int[i][0])-1] = Forca_y[i][1]

    return Vetor_Esf

def Condicoes_de_Contorno():

    MR1 = MR.copy()

    for i in range(0, Num_AP):
        NO = Tipo_AP[i][0]

        if (Tipo_AP[i][1] == 1):
            for j in range(0, (2*Num_NO)):
                MR1[(2*NO)-2][j] = 0
                MR1[j][(2*NO)-2] = 0

            MR1[(2*NO)-2][(2*NO)-2] = 1
            Vetor_Esf[(2*NO)-2] = 0

        if (Tipo_AP[i][2] == 1):
            for j in range(0, (2*Num_NO)):
                MR1[(2*NO)-1][j] = 0
                MR1[j][(2*NO)-1] = 0

            MR1[(2*NO)-1][(2*NO)-1] = 1
            Vetor_Esf[(2*NO)-1] = 0

    return MR1, Vetor_Esf

def Resolucao_Sistema_Linear():

    Desloc_Nod = np.linalg.solve(MR1, Vetor_Esf)

    return Desloc_Nod


def Reacoes_de_Apoio():

    Reacoes_AP = np.matmul(MR, Desloc_Nod)

    return Reacoes_AP
```

```
def Esforcos_Internos(Num_NO):

    Deform_Ele_CST = np.zeros(3)
    MD = np.zeros((3,3))
    Tensao_Ele_CST = np.zeros(3)

    Deform_Nod_1 = np.zeros(3)
    Deform_Nod_2 = np.zeros(3)
    Deform_Nod_3 = np.zeros(3)
    Deform_Nod_4 = np.zeros(3)
    Tensao_Nod_1 = np.zeros(3)
    Tensao_Nod_2 = np.zeros(3)
    Tensao_Nod_3 = np.zeros(3)
    Tensao_Nod_4 = np.zeros(3)

    Soma_Deform_Nod = np.zeros((Num_NO,3))
    Soma_Tensao_Nod = np.zeros((Num_NO,3))
    n = np.zeros(Num_NO)

    Deform_Nod = np.zeros((Num_NO,3))
    Tensao_Nod = np.zeros((Num_NO,3))

    if (Num_Ele_CST != 0):

        for i in range(0, Num_Ele_CST):

            E = Prop_Ele_CST[i][2]
            Poisson = Prop_Ele_CST[i][3]

            No_i = Lig_Ele_CST[i][1]
            No_int = Lig_Ele_CST[i][2]
            No_f = Lig_Ele_CST[i][3]

            u_i = Desloc_Nod[(2*No_i)-2]
            u_int = Desloc_Nod[(2*No_int)-2]
            u_f = Desloc_Nod[(2*No_f)-2]

            v_i = Desloc_Nod[(2*No_i)-1]
            v_int = Desloc_Nod[(2*No_int)-1]
            v_f = Desloc_Nod[(2*No_f)-1]

            Deform_Ele_CST[0]                                                    =
(1/(2*A[i]))*((Beta_NO_i[i]*u_i)+(Beta_NO_int[i]*u_int)+(Beta_NO_f[i]*u_f))
            Deform_Ele_CST[1]                                                    =
(1/(2*A[i]))*((Gama_NO_i[i]*v_i)+(Gama_NO_int[i]*v_int)+(Gama_NO_f[i]*v_f))
            Deform_Ele_CST[2]                                                    =
(1/(2*A[i]))*((Gama_NO_i[i]*u_i)+(Beta_NO_i[i]*v_i)+(Gama_NO_int[i]*u_int)+
(Beta_NO_int[i]*v_int)+(Gama_NO_f[i]*u_f)+(Beta_NO_f[i]*v_f))

            MD[0][0] = E/(1-(Poisson**2))
            MD[0][1] = (E*Poisson)/(1-(Poisson**2))
            MD[1][0] = (E*Poisson)/(1-(Poisson**2))
            MD[1][1] = E/(1-(Poisson**2))
            MD[2][2] = (E/(1-(Poisson**2)))*((1-Poisson)/2)

            Tensao_Ele_CST = np.matmul(MD, Deform_Ele_CST)

#Matriz da somatória das deformacoes nodais
            Soma_Deform_Nod[(No_i)-1][0]  =  Soma_Deform_Nod[(No_i)-1][0]  +
Deform_Ele_CST[0]
```

```
            Soma_Deform_Nod[(No_i)-1][1]  =  Soma_Deform_Nod[(No_i)-1][1]  +
Deform_Ele_CST[1]
            Soma_Deform_Nod[(No_i)-1][2]  =  Soma_Deform_Nod[(No_i)-1][2]  +
Deform_Ele_CST[2]

            Soma_Deform_Nod[(No_int)-1][0] = Soma_Deform_Nod[(No_int)-1][0]
+ Deform_Ele_CST[0]
            Soma_Deform_Nod[(No_int)-1][1] = Soma_Deform_Nod[(No_int)-1][1]
+ Deform_Ele_CST[1]
            Soma_Deform_Nod[(No_int)-1][2] = Soma_Deform_Nod[(No_int)-1][2]
+ Deform_Ele_CST[2]

            Soma_Deform_Nod[(No_f)-1][0]  =  Soma_Deform_Nod[(No_f)-1][0]  +
Deform_Ele_CST[0]
            Soma_Deform_Nod[(No_f)-1][1]  =  Soma_Deform_Nod[(No_f)-1][1]  +
Deform_Ele_CST[1]
            Soma_Deform_Nod[(No_f)-1][2]  =  Soma_Deform_Nod[(No_f)-1][2]  +
Deform_Ele_CST[2]

#Matriz da somatória das tensoes nodais
            Soma_Tensao_Nod[(No_i)-1][0]  =  Soma_Tensao_Nod[(No_i)-1][0]  +
Tensao_Ele_CST[0]
            Soma_Tensao_Nod[(No_i)-1][1]  =  Soma_Tensao_Nod[(No_i)-1][1]  +
Tensao_Ele_CST[1]
            Soma_Tensao_Nod[(No_i)-1][2]  =  Soma_Tensao_Nod[(No_i)-1][2]  +
Tensao_Ele_CST[2]

            Soma_Tensao_Nod[(No_int)-1][0] = Soma_Tensao_Nod[(No_int)-1][0]
+ Tensao_Ele_CST[0]
            Soma_Tensao_Nod[(No_int)-1][1] = Soma_Tensao_Nod[(No_int)-1][1]
+ Tensao_Ele_CST[1]
            Soma_Tensao_Nod[(No_int)-1][2] = Soma_Tensao_Nod[(No_int)-1][2]
+ Tensao_Ele_CST[2]

            Soma_Tensao_Nod[(No_f)-1][0]  =  Soma_Tensao_Nod[(No_f)-1][0]  +
Tensao_Ele_CST[0]
            Soma_Tensao_Nod[(No_f)-1][1]  =  Soma_Tensao_Nod[(No_f)-1][1]  +
Tensao_Ele_CST[1]
            Soma_Tensao_Nod[(No_f)-1][2]  =  Soma_Tensao_Nod[(No_f)-1][2]  +
Tensao_Ele_CST[2]

#total de valores somados
            n[(No_i)-1] = n[(No_i)-1] + 1
            n[(No_int)-1] = n[(No_int)-1] + 1
            n[(No_f)-1] = n[(No_f)-1] + 1

    if (Num_Ele_CSQ != 0):

        for i in range(0, Num_Ele_CSQ):


            E = Prop_Ele_CSQ[i][2]
            Poisson = Prop_Ele_CSQ[i][3]

            No_i = Lig_Ele_CSQ[i][1]
            No_int_1 = Lig_Ele_CSQ[i][2]
            No_int_2 = Lig_Ele_CSQ[i][3]
            No_f = Lig_Ele_CSQ[i][4]

            Csi_NO_i = ((0-a[i])/a[i])
            Csi_NO_int_1 = (((2*a[i])-a[i])/a[i])
```

```
            Eta_NO_i = ((0-b[i])/b[i])
            Eta_NO_f = (((2*b[i])-b[i])/b[i])

            u_i = Desloc_Nod[(2*No_i)-2]
            u_int_1 = Desloc_Nod[(2*No_int_1)-2]
            u_int_2 = Desloc_Nod[(2*No_int_2)-2]
            u_f = Desloc_Nod[(2*No_f)-2]

            v_i = Desloc_Nod[(2*No_i)-1]
            v_int_1 = Desloc_Nod[(2*No_int_1)-1]
            v_int_2 = Desloc_Nod[(2*No_int_2)-1]
            v_f = Desloc_Nod[(2*No_f)-1]

            Deform_Nod_1[0]            =            (((1-Eta_NO_i)*(u_int_1-
u_i))+((1+Eta_NO_i)*(u_int_2-u_f)))/(4*a[i])
            Deform_Nod_1[1]     =     (((1+Csi_NO_i)*(v_int_2-v_int_1))+((1-
Csi_NO_i)*(v_f-v_i)))/(4*b[i])
            Deform_Nod_1[2]    =    ((b[i]*(((1+Eta_NO_i)*(v_int_2-v_f))+((1-
Eta_NO_i)*(v_int_1-v_i))))+(a[i]*(((1+Csi_NO_i)*(u_int_2-u_int_1))+((1-
Csi_NO_i)*(u_f-u_i)))))/(4*a[i]*b[i])

            Deform_Nod_2[0]            =            (((1-Eta_NO_i)*(u_int_1-
u_i))+((1+Eta_NO_i)*(u_int_2-u_f)))/(4*a[i])
            Deform_Nod_2[1]   =   (((1+Csi_NO_int_1)*(v_int_2-v_int_1))+((1-
Csi_NO_int_1)*(v_f-v_i)))/(4*b[i])
            Deform_Nod_2[2]    =    ((b[i]*(((1+Eta_NO_i)*(v_int_2-v_f))+((1-
Eta_NO_i)*(v_int_1-v_i))))+(a[i]*(((1+Csi_NO_int_1)*(u_int_2-u_int_1))+((1-
Csi_NO_int_1)*(u_f-u_i)))))/(4*a[i]*b[i])

            Deform_Nod_3[0]            =            (((1-Eta_NO_f)*(u_int_1-
u_i))+((1+Eta_NO_f)*(u_int_2-u_f)))/(4*a[i])
            Deform_Nod_3[1]   =   (((1+Csi_NO_int_1)*(v_int_2-v_int_1))+((1-
Csi_NO_int_1)*(v_f-v_i)))/(4*b[i])
            Deform_Nod_3[2]    =    ((b[i]*(((1+Eta_NO_f)*(v_int_2-v_f))+((1-
Eta_NO_f)*(v_int_1-v_i))))+(a[i]*(((1+Csi_NO_int_1)*(u_int_2-u_int_1))+((1-
Csi_NO_int_1)*(u_f-u_i)))))/(4*a[i]*b[i])

            Deform_Nod_4[0]            =            (((1-Eta_NO_f)*(u_int_1-
u_i))+((1+Eta_NO_f)*(u_int_2-u_f)))/(4*a[i])
            Deform_Nod_4[1]     =     (((1+Csi_NO_i)*(v_int_2-v_int_1))+((1-
Csi_NO_i)*(v_f-v_i)))/(4*b[i])
            Deform_Nod_4[2]    =    ((b[i]*(((1+Eta_NO_f)*(v_int_2-v_f))+((1-
Eta_NO_f)*(v_int_1-v_i))))+(a[i]*(((1+Csi_NO_i)*(u_int_2-u_int_1))+((1-
Csi_NO_i)*(u_f-u_i)))))/(4*a[i]*b[i])

            MD[0][0] = E/(1-(Poisson**2))
            MD[0][1] = (E*Poisson)/(1-(Poisson**2))
            MD[1][0] = (E*Poisson)/(1-(Poisson**2))
            MD[1][1] = E/(1-(Poisson**2))
            MD[2][2] = (E/(1-(Poisson**2)))*((1-Poisson)/2)

            Tensao_Nod_1 = np.matmul(MD, Deform_Nod_1)
            Tensao_Nod_2 = np.matmul(MD, Deform_Nod_2)
            Tensao_Nod_3 = np.matmul(MD, Deform_Nod_3)
            Tensao_Nod_4 = np.matmul(MD, Deform_Nod_4)

#Matriz da somatória das deformacoes nodais
            Soma_Deform_Nod[(No_i)-1][0]  =  Soma_Deform_Nod[(No_i)-1][0]  +
Deform_Nod_1[0]
```

```
            Soma_Deform_Nod[(No_i)-1][1]  =  Soma_Deform_Nod[(No_i)-1][1]  +
Deform_Nod_1[1]
            Soma_Deform_Nod[(No_i)-1][2]  =  Soma_Deform_Nod[(No_i)-1][2]  +
Deform_Nod_1[2]

            Soma_Deform_Nod[(No_int_1)-1][0]  =  Soma_Deform_Nod[(No_int_1)-
1][0] + Deform_Nod_2[0]
            Soma_Deform_Nod[(No_int_1)-1][1]  =  Soma_Deform_Nod[(No_int_1)-
1][1] + Deform_Nod_2[1]
            Soma_Deform_Nod[(No_int_1)-1][2]  =  Soma_Deform_Nod[(No_int_1)-
1][2] + Deform_Nod_2[2]

            Soma_Deform_Nod[(No_int_2)-1][0]  =  Soma_Deform_Nod[(No_int_2)-
1][0] + Deform_Nod_3[0]
            Soma_Deform_Nod[(No_int_2)-1][1]  =  Soma_Deform_Nod[(No_int_2)-
1][1] + Deform_Nod_3[1]
            Soma_Deform_Nod[(No_int_2)-1][2]  =  Soma_Deform_Nod[(No_int_2)-
1][2] + Deform_Nod_3[2]

            Soma_Deform_Nod[(No_f)-1][0]  =  Soma_Deform_Nod[(No_f)-1][0]  +
Deform_Nod_4[0]
            Soma_Deform_Nod[(No_f)-1][1]  =  Soma_Deform_Nod[(No_f)-1][1]  +
Deform_Nod_4[1]
            Soma_Deform_Nod[(No_f)-1][2]  =  Soma_Deform_Nod[(No_f)-1][2]  +
Deform_Nod_4[2]

#Matriz da somatória das tensoes nodais
            Soma_Tensao_Nod[(No_i)-1][0]  =  Soma_Tensao_Nod[(No_i)-1][0]  +
Tensao_Nod_1[0]
            Soma_Tensao_Nod[(No_i)-1][1]  =  Soma_Tensao_Nod[(No_i)-1][1]  +
Tensao_Nod_1[1]
            Soma_Tensao_Nod[(No_i)-1][2]  =  Soma_Tensao_Nod[(No_i)-1][2]  +
Tensao_Nod_1[2]

            Soma_Tensao_Nod[(No_int_1)-1][0]  =  Soma_Tensao_Nod[(No_int_1)-
1][0] + Tensao_Nod_2[0]
            Soma_Tensao_Nod[(No_int_1)-1][1]  =  Soma_Tensao_Nod[(No_int_1)-
1][1] + Tensao_Nod_2[1]
            Soma_Tensao_Nod[(No_int_1)-1][2]  =  Soma_Tensao_Nod[(No_int_1)-
1][2] + Tensao_Nod_2[2]

            Soma_Tensao_Nod[(No_int_2)-1][0]  =  Soma_Tensao_Nod[(No_int_2)-
1][0] + Tensao_Nod_3[0]
            Soma_Tensao_Nod[(No_int_2)-1][1]  =  Soma_Tensao_Nod[(No_int_2)-
1][1] + Tensao_Nod_3[1]
            Soma_Tensao_Nod[(No_int_2)-1][2]  =  Soma_Tensao_Nod[(No_int_2)-
1][2] + Tensao_Nod_3[2]

            Soma_Tensao_Nod[(No_f)-1][0]  =  Soma_Tensao_Nod[(No_f)-1][0]  +
Tensao_Nod_4[0]
            Soma_Tensao_Nod[(No_f)-1][1]  =  Soma_Tensao_Nod[(No_f)-1][1]  +
Tensao_Nod_4[1]
            Soma_Tensao_Nod[(No_f)-1][2]  =  Soma_Tensao_Nod[(No_f)-1][2]  +
Tensao_Nod_4[2]

#total de valores somados
            n[(No_i)-1] = n[(No_i)-1] + 1
            n[(No_int_1)-1] = n[(No_int_1)-1] + 1
            n[(No_int_2)-1] = n[(No_int_2)-1] + 1
            n[(No_f)-1] = n[(No_f)-1] + 1
```

```
    for i in range(0, Num_NO):

        for j in range(0, 3):

            Deform_Nod[i][j] = Soma_Deform_Nod[i][j] / n[i]
            Tensao_Nod[i][j] = Soma_Tensao_Nod[i][j] / n[i]

    return Deform_Nod, Tensao_Nod

def Tensoes_Principais():

    Tensoes_Principais = np.zeros((Num_NO,3))
    Angulos_Principais = np.zeros((Num_NO,2))

    for i in range(0, Num_NO):

        Tensao_x = Tensao_Nod[i][0]
        Tensao_y = Tensao_Nod[i][1]
        Tensao_xy = Tensao_Nod[i][2]

        Tensao_1          =          ((Tensao_x+Tensao_y)/2)+(((((Tensao_x-
Tensao_y)/2)**2)+(Tensao_xy**2))**(1/2))
        Tensao_2          =          ((Tensao_x+Tensao_y)/2)-(((((Tensao_x-
Tensao_y)/2)**2)+(Tensao_xy**2))**(1/2))
        Tensao_cisal_max                    =                   (((((Tensao_x-
Tensao_y)/2)**2)+(Tensao_xy**2))**(1/2))

        Teta_p_rad = (math.atan((2*Tensao_xy)/(Tensao_x-Tensao_y)))/2
        Teta_s_rad = (math.atan(-(Tensao_x-Tensao_y)/(2*Tensao_xy)))/2

        Teta_p_graus = math.degrees(Teta_p_rad)
        Teta_s_graus = math.degrees(Teta_s_rad)

        Tensoes_Principais[i][0] = Tensao_1
        Tensoes_Principais[i][1] = Tensao_2
        Tensoes_Principais[i][2] = Tensao_cisal_max

        Angulos_Principais[i][0] = Teta_p_graus
        Angulos_Principais[i][1] = Teta_s_graus


    return Tensoes_Principais, Angulos_Principais


def Deformacoes_Principais():

    Deformacoes_Principais = np.zeros((Num_NO,3))

    for i in range(0, Num_NO):

        Deform_x = Deform_Nod[i][0]
        Deform_y = Deform_Nod[i][1]
        Deform_xy = Deform_Nod[i][2]

        Deform_1          =          ((Deform_x+Deform_y)/2)+(((((Deform_x-
Deform_y)/2)**2)+((Deform_xy/2)**2))**(1/2))
        Deform_2          =          ((Deform_x+Deform_y)/2)-(((((Deform_x-
Deform_y)/2)**2)+((Deform_xy/2)**2))**(1/2))
        Deform_cisal_max                  =                 2*(((((Deform_x-
Deform_y)/2)**2)+((Deform_xy/2)**2))**(1/2))
```

```
        Deformacoes_Principais[i][0] = Deform_1
        Deformacoes_Principais[i][1] = Deform_2
        Deformacoes_Principais[i][2] = Deform_cisal_max

    return Deformacoes_Principais


def Saida_de_Dados(Desloc_Nod):

    with open('Saida.txt', 'w') as saida:

        saida.write('DESLOCAMENTOS NODAIS\n')
        saida.write('Nó u v\n')

        for i in range(0, len(desloc_nod)):

            if (i % 2 == 0):
                saida.write(str('{0:.0f}'.format((i/2)+1))+'
'+str(desloc_nod[i]))
            else:
                saida.write(' '+str(desloc_nod[i])+'\n')


        saida.write('\nREACOES DE APOIO\n')
        saida.write('Nó u v\n')


        for i in range(0, len(Reacoes_AP)):

            if (i % 2 == 0):
                saida.write(str('{0:.0f}'.format((i/2)+1))+'
'+str(Reacoes_AP[i]))

            else:
                saida.write(' '+str(Reacoes_AP[i])+'\n')


        saida.write('\nDEFORMAÇÕES MÉDIAS NODAIS\n')
        saida.write('Nó Def.x Def.y Def.xy\n')
        for i in range(0, len(Deform_Nod)):
            saida.write(str(i+1)+' ')
            for j in range(0, len(Deform_Nod[0])):
                saida.write(str(Deform_Nod[i][j])+' ')
            saida.write('\n')

        saida.write('\nTENSÕES MÉDIAS NODAIS\n')
        saida.write('Nó Tx Ty Txy\n')
        for i in range(0, len(Tensao_Nod)):
            saida.write(str(i+1)+' ')
            for j in range(0, len(Tensao_Nod[0])):
                saida.write(str(Tensao_Nod[i][j])+' ')
            saida.write('\n')

        saida.write('\nTensoes principais e de cisalhamento maxima\n')
        saida.write('Nó T1 T2 TCM\n')
        for i in range(0, len(Tensoes_Principais)):
            saida.write(str(i+1)+' ')
            for j in range(0, len(Tensoes_Principais[0])):
                saida.write(str(Tensoes_Principais[i][j])+' ')
            saida.write('\n')
```

```
        saida.write('\nANGULO PRINCIPAIS\n')
        saida.write('Nó  Teta p   Teta s\n')
        for i in range(0, len(Angulos_Principais)):
            saida.write(str(i+1)+' ')
            for j in range(0, len(Angulos_Principais[0])):
                saida.write(str(Angulos_Principais[i][j])+' ')
            saida.write('\n')

        saida.write('\nDEFORCOES PRINCIPAIS E DE CISALHAMENTO MAXIMA\n')
        saida.write('Nó Def.1 Def.2 DCM\n')
        for i in range(0, len(Deformacoes_Principais)):
            saida.write(str(i+1)+' ')
            for j in range(0, len(Deformacoes_Principais[0])):
                saida.write(str(Deformacoes_Principais[i][j])+' ')
            saida.write('\n')


#Programa Principal

Num_NO, Num_Ele_CST, Num_Ele_CSQ, Num_AP, Num_For_x, Num_For_y, Coord_NO,
Lig_Ele_CST,   Lig_Ele_CSQ,   Tipo_AP,   Forca_x,   Forca_y,   Prop_Ele_CST,
Prop_Ele_CSQ = abertura_de_arquivos()

if (Num_Ele_CST != 0):
    Beta_NO_i, Beta_NO_int, Beta_NO_f, Gama_NO_i, Gama_NO_int, Gama_NO_f, A
= Propriedades_Geometricas_CST(Num_Ele_CST)

if (Num_Ele_CSQ != 0):
    a, b = Propriedades_Geometricas_CSQ(Num_Ele_CSQ)

MR = Matriz_de_Rigidez(Num_NO, Num_Ele_CST, Num_Ele_CSQ)

Vetor_Esf = Vetor_Esforcos()

MR1, Vetor_Esf = Condicoes_de_Contorno()

Desloc_Nod = Resolucao_Sistema_Linear()

Reacoes_AP = Reacoes_de_Apoio()

Deform_Nod, Tensao_Nod = Esforcos_Internos(Num_NO)

Tensoes_Principais, Angulos_Principais = Tensoes_Principais()

Deformacoes_Principais = Deformacoes_Principais()

Saida_de_Dados(Desloc_Nod)
```

APÊNDICE B - ARQUIVO DE ENTRADA EXEMPLO 2 COM MALHA MISTA (COMENTADO)

!Na primeira linha entra-se com os valores referentes a quantidade de nós, elementos *CST*, elementos *CSQ*, apoios, força concentrada em x e força concentrada em y. A quantidade das outras linhas do arquivo dependerá das quantidades aqui estipuladas.

Linha 1: 6 2 1 2 2 0

!Da linha 2 até a linha 7 informa-se o número do nó e suas coordenadas x e y, respectivamente. As unidades são definidas pelo próprio usuário, no qual devem ser mantidas até o fim do arquivo.

Linha 2: 1 0 2

Linha 3: 2 0 0

Linha 4: 3 2 2

Linha 5: 4 2 0

Linha 6: 5 4 2

Linha 7: 6 4 0

!As linhas 8 e 9 contêm o número do elemento *CST*, seu nó inicial, seu nó intermediário e seu nó final. A convenção utilizada para a definição dos nós deve ser no sentido anti-horário partindo do nó inferior esquerdo.

Linha 8: 1 4 5 3

Linha 9: 2 4 6 5

!A linha 10 contêm o número do elemento *CSQ*, seu nó inicial, os dois nós intermediários e seu nó final. A convenção utilizada para a definição dos nós deve ser análoga ao elemento *CST*.

Linha 10: 1 2 4 3 1

!Nas linhas 11 e 12 deve-se informar o nó onde existe apoio, sua restrição ao deslocamento horizontal (eixo x) e restrição ao deslocamento vertical (eixo y), sendo que 1 representa restrito e 0 livre.

Linha 11: 1 1 0

Linha 12: 2 1 1

!Nas linhas 13 e 14 contém as informações pertinentes às forças em x, sendo o número do nó em que ela está aplicada e a intensidade da força. Lembrando que valores negativos significam força para esquerda (contrário ao eixo x positivo). Caso houvessem forças na direção y, as linhas seguintes deveriam conter as informações pertinentes a estas forças de forma análoga às forças em x, sendo que os valores positivos significam forças para cima e os valores negativos forças para baixo (de acordo com o eixo y).

Linha 13: 2 1 1

Linha 14: 2 1 1

!Nas linhas 15 e 16 informa-se o número do elemento *CST*, sua espessura, Módulo de Elasticidade Longitudinal e Coeficiente de Poisson.

Linha 15: 1 1 1.0 0.25

Linha 16: 2 1 1.0 0.25

!Na linha 17 informa-se o número do elemento *CSQ*, sua espessura, Módulo de Elasticidade Longitudinal e Coeficiente de Poisson.

Linha 17: 1 1 1.0 0.25

APÊNDICE C - ARQUIVO DE SAÍDA EXEMPLO 2 COM MALHA MISTA

DESLOCAMENTOS NODAIS

Nó	u	v
1	0.00000	-0.50000
2	0.00000	0.00000
3	2.00000	-0.50000
4	2.00000	-9.692150648618011e-16
5	4.00000	-0.5000000000000013
6	4.00000	-1.4288790717142585e-15

REACOES DE APOIO

Nó	Rx	Ry
1	-1.00000	-5.551115123125783e-17
2	-1.00000	1.6653345369377348e-16
3	-2.7755575615628914e-16	1.1102230246251565e-16
4	-2.1827278517753008e-16	-2.220446049250313e-16
5	1.00000	1.1102230246251565e-16
6	1.00000	-1.1102230246251565e-16

DEFORMAÇÕES MÉDIAS NODAIS

Nó	Def.x	Def.y	Def.xy
1	1.00000	-0.25000	-2.7755575615628914e-16
2	1.00000	-0.25000	-4.846075324309006e-16
3	1.00000	-0.25000	-2.7755575615628914e-17
4	1.00000	-0.25000	-2.355507124519773e-16
5	1.00000	-0.25000	-2.220446049250313e-16
6	1.00000	-0.25000	-4.440892098500626e-16

TENSÕES MÉDIAS NODAIS

Nó	Ten.x	Ten.y	Ten.xy
1	1.00000	-1.1102230246251565e-16	-1.1102230246251565e-16
2	1.00000	-1.6653345369377348e-16	-1.9384301297236023e-16
3	1.00000	1.1102230246251565e-16	-1.1102230246251566e-17
4	1.00000	1.1102230246251565e-16	-9.422028498079092e-17
5	1.00000	1.3877787807814457e-16	-8.881784197001253e-17
6	1.00000	1.6653345369377348e-16	-1.7763568394002506e-16

TENSOES PRINCIPAIS E DE CISALHAMENTO MAXIMA

Nó	Ten.1	Ten.2	Ten.Cisal.Max

1	1.00000	-2.220446049250313e-16	0.50000
2	1.00000	-2.220446049250313e-16	0.50000
3	1.00000	2.220446049250313e-16	0.50000
4	1.00000	2.220446049250313e-16	0.50000
5	1.00000	2.220446049250313e-16	0.50000
6	1.00000	2.220446049250313e-16	0.50000

ANGULO PRINCIPAIS

Nó	Teta p	Teta s
1	-6.36110936292703e-15	45.00000
2	-1.1106386531415906e-14	45.00000
3	-6.36110936292703e-16	45.00000
4	-5.398424673919178e-15	45.00000
5	-5.088887490341626e-15	45.00000
6	-1.0177774980683252e-14	45.00000

DEFORCOES PRINCIPAIS E DE CISALHAMENTO MAXIMA

Nó	Def.1	Def.2	DCM
1	1.00000	-0.25000	1.25000
2	1.00000	-0.25000	1.25000
3	1.00000	-0.25000	1.250000
4	1.00000	-0.25000	1.250000
5	1.00000	-0.25000	1.250000
6	1.00000	-0.25000	1.250000