UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LEANDRO MARTINEZ VIEIRA

DESENVOLVIMENTO DE CÓDIGO COMPUTACIONAL PARA ANÁLISE ELÁSTICA LINEAR DE CHAPAS UTILIZANDO O ELEMENTO FINITO CSQ

CAMPO MOURÃO

2019

LEANDRO MARTINEZ VIEIRA

DESENVOLVIMENTO DE CÓDIGO COMPUTACIONAL PARA ANÁLISE ELÁSTICA LINEAR DE CHAPAS UTILIZANDO O ELEMENTO FINITO 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

DESENVOLVIMENTO DE CÓDIGO COMPUTACIONAL PARA ANÁLISE ELÁSTICA LINEAR DE CHAPAS UTILIZANDO O ELEMENTO FINITO CSQ

por

Leandro Martinez Vieira

Este Trabalho de Conclusão de Curso foi apresentado às 09h00min do dia 04 de dezembro 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. Me. Angelo Giovanni Bonfim Corelhano
(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

Em primeiro lugar agradeço à Deus, pai celestial e redentor, meu abrigo nos momentos mais difíceis, aquele que sempre me deu suporte e não me deixou desistir quando pensei não ter mais forças. A ti, toda honra e glória, Senhor!

Aos meus queridos pais, Nelma Martinez Vieira e Selmo Francisco Vieira, o meu muito obrigado, sem vocês eu não seria nada, sem vocês nada disso teria acontecido. Obrigado por todo o suporte, todo carinho e amor que sempre me deram. Peço desculpas pelas muitas vezes que fui ausente, quando disse que não tinha tempo para conversar, quando não dei toda a atenção que vocês merecem. Pai, mãe, vocês são tudo para mim. Essa conquista não é minha, ela é de vocês.

Ao meu irmão Pedro Henrique, que junto com meus pais formam o meu elo mais forte, obrigado por ser quem você é. Obrigado por ser meu irmão. Eu amo nossa família!

Ao restantes dos meus familiares, em especial meus avós, obrigado por todo o amor que sempre me proporcionaram, obrigado pela educação que me deram e por tudo que vocês sempre fizeram por mim. Agradeço à Deus todos os dias por ter vocês em minha vida.

Ao meu orientador, Professor Dr. Leandro Waidemam, eu não tenho nem palavras que consigam demonstrar o quanto me sinto honrado de ter sido seu orientando. Muito obrigado por todo o apoio de sempre, por toda a disponibilidade e por toda a vontade que você sempre teve em me ajudar. Mais do que meu professor, se tornou um amigo e uma pessoa que vou levar para sempre com muito carinho.

Aos demais professores que tive a honra de ter durante minha jornada acadêmica, obrigado por serem exemplos para mim, exemplos de profissionais, de persistência, de sabedoria, de caráter e de seres humanos. Vocês cumprem com louvor o cargo mais importante que temos em nossa sociedade. Vou levar cada um em minhas melhores lembranças pelo resto da vida.

Aos meus companheiros de trabalhos acadêmicos, Vinicius Moura e Fernando Freire, o meu muito obrigado por toda a ajuda e por estarem junto comigo em muitos momentos durante o desenvolvimento das pesquisas e do código. Um ajudando o outro, sempre foi o nosso lema.

Aos meus outros inúmeros amigos que tive a honra de conhecer na faculdade, em especial Tamires, Jéssica e Nadine, que sempre estiveram mais próximas de mim e

sempre me ajudaram, seja com questões acadêmicas, seja com questões pessoais. Muito obrigado por cada momento com vocês, por cada conversa amiga, por cada conselho, por cada sorriso que vocês me proporcionaram ao longo desses anos.

Aos meus demais amigos fora da faculdade, o meu muito obrigado por cada momento de diversão, por cada conversa que tivemos. Vou levar cada um dos meus amigos no peito pelo resto da minha vida.

Aos demais servidores e funcionários da UTFPR, pois todos foram muito importantes durante essa jornada.

RESUMO

Com o advento dos computadores e o avanço tecnológico surgiram novas técnicas que revolucionaram a análise de estruturas por parte dos engenheiros. Uma dessas técnicas foi o Método dos Elementos Finitos (MEF), que é um método numérico utilizado para resolver estruturas no qual as soluções analíticas se tornam inviáveis devido a sua complexibilidade. O método consiste em dividir as estruturas em pequenas partes, denominadas elementos finitos, que se unem por pontos específicos denominados por nós. A partir da modelagem numérica e consequente desenvolvimento das equações de equilíbrio que regem o problema, é possível implementar o MEF na forma computacional para se realizar a análise estrutural. Quanto maior a quantidade de elementos, melhor será a aproximação dos resultados e maior será o esforço computacional. Dentro da análise elástica linear bidimensional há o elemento conhecido como *Constant Strain Quadrilateral (CSQ)*, que é caracterizado pelo formato retangular e por possuir 4 pontos nodais e 2 graus de liberdade por nó, sendo estes dois deslocamentos coplanares perpendiculares entre si e na direção dos eixos coordenados. O elemento *CSQ* é próprio para a análise de chapas, que são caracterizadas por serem elementos planos e sujeitos a carregamento contido em seu próprio plano. Nesse contexto, este trabalho tem como objetivo desenvolver um código computacional em linguagem *Python* que realize a análise elástica linear de chapas utilizando elementos finitos *CSQ*. Ao longo do trabalho é apresentada a formulação matemática do método e a dedução da matriz de rigidez do elemento *CSQ*. Por último, o código desenvolvido é utilizado para simular exemplos de chapas, sendo os resultados obtidos comparados com os fornecidos pela literatura ou com os obtidos em outros softwares já renomados na área de análise estrutural. A comparação de resultados mostra que o código desenvolvido fornece resultados adequados e satisfatórios.

Palavras-chave: Análise Estrutural de Chapas. Método dos Elementos Finitos. Elemento *CSQ*.

ABSTRACT

With the advent of computers and technological advancement new techniques emerged that revolutionized the analysis of structures by engineers. One such technique was the Finite Element Method (FEM), which is a numerical method used to solve structures in which analytical solutions become unviable due to their complexity. The method consists of dividing the structures into small parts, called finite elements, which are joined by specific points called nodes. From the numerical modeling and consequent development of the equilibrium equations that govern the problem, it is possible to implement the FEM in computational form to perform the structural analysis. The greater the number of elements, the better the approximation of the results and the greater the computational effort. Within the two-dimensional linear elastic analysis is the element known as *Constant Strain Quadrilateral* (*CSQ*), which is characterized by rectangular shape and 4 nodal points and 2 degrees of freedom per node, being these two coplanar displacements perpendicular to each other and towards the coordinate axes. The *CSQ* element is suitable for the analysis of plane stress problems, which are characterized by being flat elements and subject to loading contained in their own plane. In this context, this work aims to develop a computational code in *Python* language that performs linear elastic of plane stress problems using finite element *CSQ*. Throughout the work is presented the mathematical formulation of the method and the deduction of the stiffness matrix of the *CSQ* element. Finally, the developed code is used to simulate various examples, and the results obtained are compared with those provided in the literature or with those obtained in other software already renowned in the area of structural analysis. The comparison of results shows that the developed code provides adequate and satisfactory results.

Keywords: Plane Stress Structural Analysis. Finite Element Method. *CSQ* element.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE SÍMBOLOS E SIGLAS

MEF	Método dos Elementos Finitos
CSQ	Constant Strain Quadrilateral
PTV	Princípio dos Trabalhos Virtuais
CST	Constant Strain Triangle
σ	Tensão Normal
τ	Tensão de Cisalhamento
u	Deslocamento Nodal na Direção do Eixo x
v	Deslocamento Nodal na Direção do Eixo y
ε	Deformação Normal
γ	Deformação Angular
$[L]$	Matriz de Operadores Diferenciais
E	Módulo de Elasticidade (Módulo de Young)
ν	Coeficiente de Poisson
G	Módulo de Elasticidade Transversal
$[C]$	Matriz de Relação Deformação x Tensão
$[E]$	Matriz de Elasticidade
We	Trabalho Externo
U	Energia de Deformação Interna
uo	Infinitesimal de Energia de Deformação Interna
du	Infinitesimal de Deslocamento na Direção do Eixo x
$d\theta$	Infinitesimal de Giro em Torno do Eixo z
dv	Infinitesimal de Deslocamento na Direção do Eixo y

$d\varphi$	Infinitesimal de Giro em Torno do Eixo x
N	Esforço Normal
M	Momento Fletor
Q	Esforço Cortante
T	Momento Torsor
F_a	Campo das Forças Externas
σ_a	Campo das Tensões Internas
D_b	Campo dos Deslocamentos Externos
ε_b	Campo das Deformações Internas
f_a	Campo dos Esforços Internos
d_b	Campo dos Deslocamentos Relativos Internos
V	Volume da Chapa
t	Espessura da Chapa
$\{F_i\}$	Vetor de Forças Externas Reais
$\{u_i\}^*$	Vetor de Deslocamentos Externos Virtuais
$\{\varepsilon\}^*$	Vetor Deformações Internas Virtuais
$\{\sigma\}$	Vetor Tensões Internas Reais
$\{\varepsilon\}$	Vetor de Deformações Nodais
$[\phi]$	Matriz das Funções de Forma
$\{u_i\}$	Vetor de Deslocamentos Nodais
$[B]$	Matriz de Deformação
$[k]$	Matriz de Rigidez Elementar

$[K]$	Matriz de Rigidez Global
ξ	Coordenada Baricêntrica na Direção do Eixo x
η	Coordenada Baricêntrica na Direção do Eixo y
a	Metade da Dimensão Horizontal do Elemento CSQ
b	Metade da Dimensão Vertical do Elemento CSQ
x_c	Coordenada Cartesiana em x do Centro do Elemento
y_c	Coordenada Cartesiana em y do Centro do Elemento
$\{\alpha\}$	Vetor de Interpolação dos Deslocamentos
$[\omega]$	Matriz de Coordenadas Baricêntricas
$[\omega_i]$	Matriz de Coordenadas Baricêntricas dos Pontos Nodais
$[k]^e$	Matriz de Rigidez Elementar do Elementos CSQ

SUMÁRIO

1 INTRODUÇÃO

Desde o princípio da humanidade o ser humano sempre teve a necessidade de ter o seu abrigo, de ter aquele local que fosse somente seu, onde ele pudesse se sentir confortável e seguro. Foi assim que o homem encontrou nas cavernas sua primeira forma de abrigo. Ao longo do tempo, foi se descobrindo que esses abrigos poderiam ser melhorados com o uso de materiais como pedras, galhos e folhas de árvores, e até mesmo que outros tipos de abrigos poderiam ser criados utilizando o barro por exemplo.

Durante toda a história, a inteligência e a capacidade do ser humano foram se desenvolvendo, possibilitando cada vez mais o surgimento de novas criações. Os abrigos então deixaram de ser cavernas e construções de barro e começaram a surgir as primeiras residências de madeira e posteriormente de alvenaria e concreto. Hoje em dia, o grande desenvolvimento tecnológico permite cada vez mais a criação de moradias mais modernas e com mais recursos, que vão desde pequenas casas até grandes conjuntos habitacionais ou edifícios altos, os chamados arranha-céu.

Ao longo de toda essa história existe o fato em comum de que o homem sempre esteve em busca de construções mais seguras, mas que não deixasse de lado os aspectos de conforto, economia e estética. Em meio a esse contexto surge a figura de um profissional responsável por juntar todos esses aspectos em busca de uma solução que seja a mais eficiente possível, este profissional é então denominado engenheiro civil.

No campo da engenharia a evolução se deu principalmente com o grande avanço tecnológico ocorrido em meados do século XX, que foi quando ocorreu o advento dos computadores e se começou a utilizar softwares para realizar trabalhos que antes eram feitos de forma manual. A utilização de softwares proporcionou aos engenheiros a criação de projetos mais elaborados, com maiores graus de dificuldade e em um menor espaço de tempo, otimizando o trabalho do profissional.

Hoje em dia, a maioria dos projetos elaborados por um engenheiro civil são realizado por meio de softwares. Assim, é de extrema importância que o profissional ao trabalhar com estes softwares tenha conhecimento das ferramentas que estão sendo utilizadas e saiba interpretar os resultados obtidos, analisando com sua experiência e bom senso a coerência e a compatibilidade dos dados, uma vez que não há software no mundo mais potente que a mente de um bom engenheiro. Na área de engenharia de estruturas isso faz ainda mais sentido, pois resultados obtidos de maneira equivocada dos softwares

podem gerar consequências catastróficas na edificação, colocando em risco a vida de pessoas.

Em meio a essa necessidade de uso de softwares que sejam eficientes e seguros aparece a importância de o engenheiro entender também como foram desenvolvidos esses programas e não somente como utiliza-los. Atualmente, muitos softwares da área de estruturas utilizam o Método dos Elementos Finitos (MEF) no seu código operacional, isto porque este é um método numérico de fácil implementação computacional e que consegue representar de maneira muito satisfatória o comportamento das estruturas quanto aos parâmetros de deslocamentos, deformações, esforços internos e tensões.

O MEF consiste em um método numérico utilizado para análise de estruturas em que a solução analítica fica inviável. O método cria um modelo físico, através da divisão da estrutura em um número finito de elementos, que representa o modelo matemático que seria utilizado pela solução analítica. Dessa forma, consegue obter resultados satisfatórios ao analisar o comportamento da estrutura quando esta estiver sendo solicitada por ações externas.

Dentro do contexto do método, existem diversos tipos de elementos que se diferenciam basicamente pelo seu formato, pela quantidade de nós e graus de liberdade que apresentam. Na análise bidimensional de estruturas existe o elemento conhecido como *Constant Strain Quadrilateral* (*CSQ*), que é caracterizado pelo seu formato retangular e por apresentar 4 pontos nodais, posicionados nos seus vértices, e 2 graus de liberdade por nó, sendo estes dois deslocamentos coplanares e perpendiculares entre si na direção dos eixos coordenados.

A engenharia de estruturas é possivelmente a área mais complexa e que envolva uma maior habilidade de raciocínio por parte do engenheiro. Isto porque existem diversos tipos de elementos estruturais e cada um deles possui um diferente tipo de comportamento. Neste trabalho, o elemento estrutural estudado será a chapa.

Uma chapa é um tipo de elemento pertencente ao grupo dos chamados sistemas estruturais planos, isto é, que possuem uma dimensão muito menor em relação as outras duas dimensões. Se enquadram nesse grupo os elementos de placa e de chapa, sendo a diferença entre estes é que o primeiro admite carregamento perpendicular ao seu plano, enquanto que o segundo admite carregamento contido no próprio plano.

Dessa maneira, o presente trabalho consiste na formulação do MEF aplicado em estruturas de chapa, bem como a criação de um código computacional para análise dessas estruturas.

1.1 Objetivos

1.1.1 Objetivo Geral

Apresentar e implementar um código computacional, em linguagem Python, formulado com base no Método dos Elementos Finitos utilizando o elemento *CSQ*, e que represente o comportamento elástico linear de estruturas de chapas estáticas.

1.1.2 Objetivos Específicos

- Apresentar de forma teórica e numérica a formulação do Método dos Elementos Finitos para o elemento *CSQ* (*Constant Strain Quadrilateral*);
- Apresentar a aplicabilidade prática dos problemas de chapas dentro da engenharia de estruturas;
- Desenvolver um código computacional para avaliar o comportamento das chapas, fornecendo resultados de reações de apoio, deslocamentos, tensões e deformações;
- Analisar exemplos no programa desenvolvido, bem como sua aplicabilidade na engenharia de estruturas;
- Comparar os resultados obtidos pelo programa desenvolvido com os obtidos em outros softwares renomados da área e/ou fornecidos por outros pesquisadores;
- Disponibilizar o programa para outros usuários do meio acadêmico que por algum motivo necessitem utiliza-lo.

1.2 Justificativa

A escolha do elemento de chapa se deve ao fato dele ter um papel importante no âmbito da construção civil, como no caso das vigas-parede e pilares-parede, alvenaria estrutural, paredes de concreto e também o seu amplo uso na área de estruturas metálicas,

por exemplo.

Dessa forma é intuitivo compreender a importância de se estudar esses elementos, para que assim o engenheiro consiga utiliza-los de maneira eficiente e segura nos projetos. Na hora de elaborar o projeto o engenheiro precisa saber as características da chapa utilizada, tais como suas propriedades físicas e geométricas, bem como os esforços solicitantes e resistentes. Além disso, é necessário conhecer quais serão os deslocamentos da estrutura, a quais esforços internos ela estará submetida, quais serão seus pontos críticos de tensão e se as deformações estão dentro dos padrões aceitáveis.

Segundo Martha (2010) o projeto estrutural tem como objetivo a concepção de uma estrutura que atenda requisitos para os quais ela está sendo construída, satisfazendo condições de segurança, utilização, economia, estética e legislação. A análise estrutural é então a etapa do projeto estrutural na qual é realizada a idealização do comportamento da estrutura, analisando parâmetros como tensões, deformações, esforços internos, reações de apoio e deslocamentos.

Portanto, ter em mãos uma ferramenta que consiga processar todas essas informações é de fundamental importância. Surge então a ideia de desenvolver um programa próprio, que consiga executar, a partir dos dados de entrada, a análise estrutural e fornecer ao usuário um arquivo de saída contendo as informações desejadas. O fato de usar um software próprio contribui muito para o desenvolvimento pessoal do engenheiro, pois dessa forma ele não vai estar apenas preenchendo campos de um programa já criado por um terceiro, mas sim vai estar utilizando algo que ele próprio desenvolveu, o que lhe faz ter muito mais conhecimento de todo o processo, aprimorando sua visão crítica na hora de analisar os resultados.

Além disso, optou-se por desenvolver o código com base no Método dos Elementos Finitos pois esse método se mostra como um dos mais avançados no campo de análise estrutural atualmente. Sua ampla utilização em diversas áreas, tais como em estruturas civis, mecânicas e aeronáuticas, transferência de calor e massa, entre outras, comprova que este é um método amplamente aceito no campo de pesquisas. Especialmente na área de estruturas civis, o MEF é o método base de diversos programas conhecidos, justamente por sua eficiência e capacidade de aproximar os resultados obtidos do comportamento real das estruturas.

Fish (2009) mostra a popularidade do MEF entre os cientistas e engenheiros do mundo todo. Segundo ele, nos Estados Unidos mais de U$$ 1 bilhão de dólares são gastos por ano no desenvolvimento de programas baseados no método. Além disso, uma

pesquisa realizada em 2006 no Google através da frase “elementos finitos” encontrou mais de 14 milhões de páginas tratando do assunto. Esses dados, bem como a vasta quantidade de livros acerca do tema mostra que o MEF é um método já consolidado na análise de estruturas.

Para o desenvolvimento do método, o elemento *CSQ* foi escolhido para a elaboração da malha devido a sua facilidade de uso e precisão nos resultados. A geometria retangular deste elemento permite a discretização da estrutura de forma simples e intuitiva, o que o torna um elemento de aplicação viável dentro do método, principalmente durante o desenvolvimento do código computacional. E mesmo com essa facilidade de aplicação, o elemento *CSQ*, quando utilizado em uma malha com refinamento adequado, consegue fornecer resultados muito satisfatórios e próximos do comportamento real da estrutura.

Por fim é importante salientar que o programa desenvolvido ao longo dessa pesquisa ficará disponível para a comunidade acadêmica, com o código sendo disponibilizado no trabalho, podendo servir de base para discentes que necessitem analisar o comportamento elástico linear de estruturas de chapa ou então realizar futuros trabalhos na área. Além disso, docentes da área terão a possibilidade de utilizar esse programa em suas aulas caso seja necessário, contribuindo para o aprendizado dos alunos.

1.3 Apresentação

No capítulo 1 é apresentada uma visão geral sobre o trabalho, mostrando a importância de uma eficiente análise estrutural de chapas e a aplicabilidade do MEF como método numérico para a resolução de problemas do tipo através do desenvolvimento de códigos computacionais.

No capítulo 2 é apresentada uma visão mais detalhada sobre o método através de uma revisão bibliográfica. São apresentadas as relações entre deslocamentos, deformações e tensões, e toda a formulação matemática do MEF aplicado aos problemas de chapas. A partir dos conceitos estabelecidos e do Princípio dos Trabalhos Virtuais (PTV) é deduzida analiticamente e apresentada a matriz de rigidez do elemento *CSQ*. Além disso, é mostrado como o método pode ajudar no dimensionamento de estruturas de vigas-parede de concreto armado.

No capítulo 3 é apresentado todo o desenvolvimento do código computacional objeto deste estudo. Primeiramente é mostrado um aspecto geral do código, com suas principais etapas de execução. Então, um fluxograma com cada sub-rotina é apresentado para facilitar a visualização do funcionamento do código. Posteriormente, cada sub-rotina é detalhada de maneira individual, de forma a mostrar o seu uso e a sua função durante a execução do código.

No capítulo 4 são simulados exemplos de chapas submetidas à diversas condições de vinculação e carregamento. Os exemplos foram executados no código desenvolvido e os resultados obtidos foram comparados com os fornecidos pela literatura ou por outros softwares já renomados. Em cada um dos exemplos é feita uma discussão sobre os resultados obtidos e sobre o uso do código.

No capítulo 5 são apresentadas as considerações finais sobre o trabalho e as conclusões que foram obtidas após a comparação dos resultados. São apresentadas também sugestões para o desenvolvimento de trabalhos futuros.

Nos apêndices são apresentados o código fonte do programa desenvolvido, bem como um exemplo do arquivo de saída que é fornecido por ele após a simulação numérica de um problema.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo é apresentada toda a fundamentação teórica do MEF aplicada aos problemas de chapas. São apresentados os conceitos básicos que fundamentam o equacionamento e também toda a dedução da matriz de rigidez do elemento CSQ. Além disso, são apresentados estudos sobre a utilização das chapas dentro da engenharia de estruturas.

2.1 Análise Estrutural

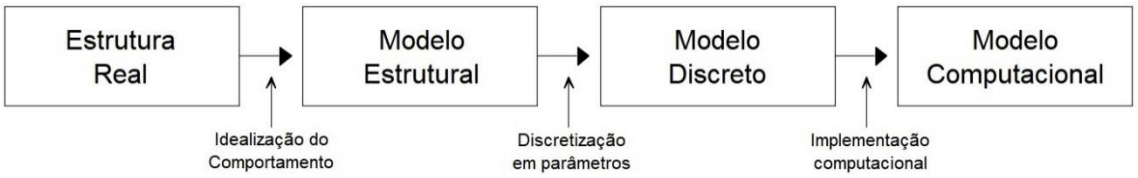
Para Sussekind (1981) a análise estrutural é a parte da mecânica que estuda as estruturas, com o objetivo de determinar os esforços e as deformações que elas apresentam quando sujeitas à ação de agentes externos, como por exemplo a aplicação de carregamentos. Ainda segundo o autor, uma estrutura é um conjunto de peças ligadas entre si de forma estável, isto é, capaz de receber solicitações externas, absorve-las e transmiti-las até os seus apoios.

De acordo com Timoshenko (1983) *apud* Martha (2010), a engenharia estrutural teve seu início na antiguidade, ainda que de uma forma empírica, na construção dos grandes monumentos e pirâmides do antigo Egito, e também nas construções, como pontes e templos, das antigas Grécia e Roma. Os primeiros registros de estudos sobre estática e mecânica dos sólidos foram atribuídos a Leonardo da Vinci (1452-1519), sendo considerado o primeiro grande estudo na área a publicação do livro *Duas Ciências*, de Galileu, no ano de 1638. Após isso, diversos outros autores tiveram influência na área, tais como Jacob Bernoulli (1654-1705), Euler (1707-1783), Lagrange (1736-1813), Kirchhoff (1824-1887), Mohr (1835-1918), entre outros.

Ainda para Martha (2010) a engenharia estrutural teve um grande avanço no final do século XIX, com a Revolução Industrial. Nesse período, novos materiais surgiram para uso nas construções, tais como concreto armado, ferro fundido e aço. Já durante o século XX os principais avanços se deram em relação aos processos construtivos e nos procedimentos de cálculo. A engenharia estrutural através de suas teorias científicas permite aos engenheiros estabelecer com segurança as solicitações que atuam na estrutura, definindo de maneira adequada o material a ser utilizado e as dimensões de cada componente.

Martha (2010) trata a análise estrutural moderna em 4 níveis de abstração com relação a estrutura analisada, sendo o primeiro nível aquele que representa a estrutura real tal como ela é. A Figura 1 abaixo exemplifica a distribuição de níveis.

Figura 1 - Níveis da análise estrutural.



Fonte: Martha (2010).

O segundo nível trata do modelo estrutural, isto é, o modelo analítico utilizado para representar matematicamente a estrutura. Esse modelo incorpora teorias e hipóteses baseadas em leis físicas, tais como o equilíbrio entre forças e tensões, as relações de compatibilidade de deslocamentos e deformações, e as leis constitutivas dos materiais que compõem a estrutura. Durante a concepção desse modelo são adotadas diversas hipóteses simplificadoras, que são baseadas em teorias físicas e resultados experimentais e estatísticos. Essas hipóteses são referentes a geometria do modelo, as condições de suporte (ligação com o meio externo), o comportamento dos materiais e as solicitações que atuam na estrutura.

O terceiro nível se refere ao modelo discreto, que é o modelo concebido dentro das metodologias de cálculo dos métodos de análise. De maneira geral, os métodos de análise utilizam parâmetros para representar o comportamento da estrutura. Os tipos de parâmetros adotados dependem do método utilizado, como por exemplo nos casos de elementos reticulados, as forças e os deslocamentos são parâmetros para o método das forças e o método dos deslocamentos, respectivamente. Já no caso do Método dos Elementos Finitos os parâmetros são os deslocamentos dos nós de cada elemento.

Por fim o quarto e último nível se refere ao modelo computacional, que abrange a simulação do comportamento da estrutura por meio de um computador. Nesse contexto, torna-se importante conceitos como a estrutura de dados e os procedimentos para a criação dos modelos anteriores, bem como o atributo de dados e análise dos resultados obtidos.

2.2 História do Método dos Elementos Finitos

De acordo com Vaz (2011) o Método dos Elementos Finitos ganhou projeção a partir de meados dos anos 50 do século XX, principalmente devido aos trabalhos de John Argyris, professor no *Imperial College* em Londres, e também de um grupo de engenheiros da Boeing, liderados pelo professor Ray W. Clough. Contudo, o trabalho considerado como pioneiro do método foi realizado em 1943 pelo matemático alemão Richard Courant e tratava do problema de torção de Saint-Venant. Na época, esse trabalho não teve muita repercussão, principalmente devido aos fatos dos métodos numéricos não serem ainda muito reconhecidos e do uso dos computadores ainda ser algo raro.

Anos mais tarde, em 1967, o professor O.C. Zienkiewicz publicou seu livro intitulado "*The Finite Elements Methods for Engineering*", que ficou conhecido no meio acadêmico como "*The Book*" e trouxe uma grande visibilidade para o método em todo o mundo.

No Brasil, o primeiro trabalho surgiu em 1970 na Coppe – UFRJ e foi uma tese, apresentada pelo engenheiro Alcebíades Vasconcelos, na qual ele desenvolveu um programa para análise de estruturas de estado plano com o uso do elemento triangular *CST* e usou este programa para comparar os resultados obtidos com a Teoria da Elasticidade.

Segundo Fish (2009) os primeiros programas em elementos finitos foram desenvolvidos na década de 1960. Um dos mais conhecidos foi o programa desenvolvido pela NASA, no ano de 1965, que ficou conhecido como NASTRAN. Esse programa inclui análise de tensões em duas e três dimensões em vigas, elementos de casca e estruturas complexas, como fuselagem de aviões, além de análise de vibrações e respostas as cargas dinâmicas. Para o desenvolvimento desse projeto, a NASA gastou cerca de U$$ 3 milhões. Anos mais tarde, o NASTRAN foi disponibilizado para comercialização, gerando patrimônio financeiro exorbitante para as empresas detentoras dos seus direitos.

No ano de 1969, John Swanson lançou no mercado o ANSYS, programa que ele tinha desenvolvido anos anteriores para a empresa em que trabalhava. Esse programa era capaz de resolver problemas lineares e problemas não-lineares em elementos finitos. Em 2006, o ANSYS chegou a uma capitalização de U$$ 1,8 bilhão.

Outro conhecido programa em elementos finitos é o ABAQUS. Ele foi desenvolvido por uma companhia chamada HKS, e inicialmente tinha proposta de

resolver apenas problemas não lineares. Conforme foi se desenvolvendo, ganhou capacidade de resolver problemas lineares e o diferencial de permitir ao usuário adicionar novos modelos e tipos de elementos. Em 2005, a companhia detentora do programa foi vendida pelo montante de U$$ 413 milhões de dólares, gerando um bom retorno aos seus investidores.

Vaz (2011) diz que o MEF é um desenvolvimento do método de análise matricial de estruturas reticuladas, impulsionado principalmente devido ao crescimento exponencial do uso dos computadores. A semelhança entre os dois métodos consiste na montagem da matriz de rigidez da estrutura que é formada pela junção das matrizes de rigidez de cada elemento. A diferença é que o MEF abrange não somente as estruturas reticuladas, como vigas e treliças, mas também estruturas contínuas bi e tridimensionais. Além disso, destaca que o MEF pode ser utilizado na análise de estruturas lineares e também nas estruturas que apresentam não linearidade física e geométrica, bem como na análise dinâmica de estruturas.

Para Ribeiro (2004) a primeira etapa da modelagem computacional de um problema físico é identificar os princípios físicos e as variáveis dependentes e independentes que influenciam o problema, resultando em um modelo matemático constituído por um conjunto de equações diferenciais. A segunda etapa consiste na resolução desse modelo matemático através do uso de um método numérico. Ainda segundo o autor, o MEF teve suas origens com o surgimento dos computadores, pois foi quando os métodos matriciais tiveram um grande desenvolvimento dentro da análise estrutural.

Segundo Vaz (2011), o MEF pode ser embasado em diversas teorias existentes. O mesmo autor afirma que sua formulação pode ser baseada no Princípio da Mínima Energia Potencial, no Método dos Resíduos Ponderados ou ainda no Princípio dos Trabalhos Virtuais.

Vaz (2011) explica que o MEF usa o conceito de discretização do contínuo, isto é, utilizar um modelo com um número finito de elementos nos quais as incógnitas são os deslocamentos de seus nós. Dessa forma, se contrapõe com a Teoria da Elasticidade, que utiliza uma quantidade infinitas de incógnitas (funções contínuas) como solução.

Para Soriano (2009) o MEF se popularizou na década de 1990 devido à ampla disponibilidade de computadores e programas de baixo custo. Isto facilitou a resolução de modelos com expressivos números de graus de liberdade.

Waidemam (2004) também destaca que com a disseminação da informática houve um aumento na capacidade de armazenamento, gerenciamento e processamento de dados, fazendo com que o engenheiro de estruturas tivesse acesso a equipamentos e programas que possibilitassem uma análise estrutural baseada em modelos mais refinados, aumentando assim a confiabilidade e diminuindo os custos dos projetos.

Pereira (2005) diz que a generalização de meios de cálculo automático cada vez mais potentes tem difundido a utilização do MEF. Porém, dado o caráter aproximado das soluções fornecidas por este método, o desconhecimento dos seus fundamentos pode levar a resultados catastróficos na sua utilização. Dessa forma, o licenciado em engenharia civil que deseja trabalhar com estruturas tem a necessidade de aprender sobre o método.

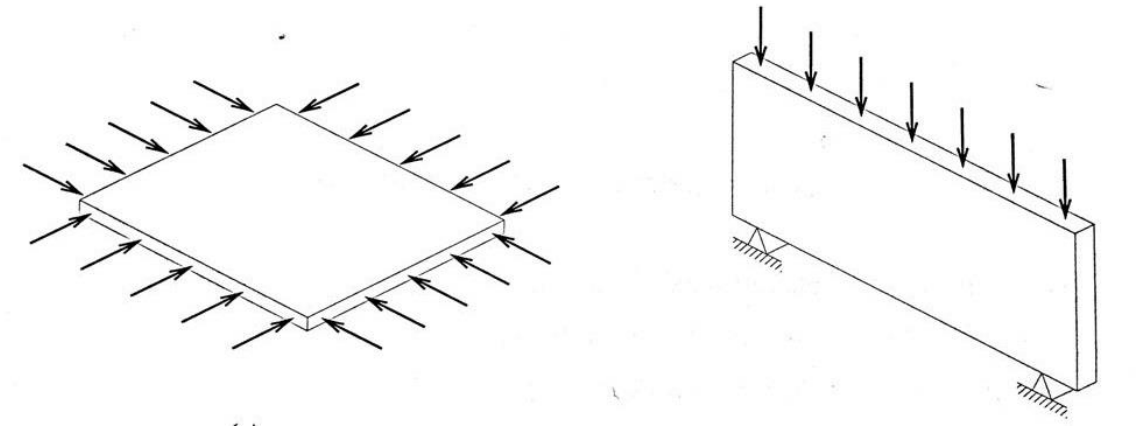
Atualmente, o uso difundido de computadores torna o MEF um dos métodos mais viáveis. Muitos programas de análise estrutural têm seu código de programação baseado nesse método. Vaz (2011) cita como exemplo o SAP2000, e os programas já citados anteriormente, como o ANSYS, o ABAQUS e o NASTRAN.

2.3 Elemento Estrutural de Chapa

Para Sales (2005) as estruturas devem ser entendidas como disposições racionais e adequadas de diversos elementos estruturais. Entende-se por elementos todo corpo sólido deformável com capacidade de resistir e transmitir solicitações.

Segundo Azevedo (2003), a chapa pertence ao grupo das chamadas estruturas laminares. Sales (2005) afirma que lâmina é o nome atribuído à um corpo que possua uma de suas dimensões muito menor que as outras duas. Além disso, atribui o nome de folha a uma estrutura constituída por uma ou mais lâminas. Dessa maneira, classifica chapa como sendo uma folha plana sujeita à carregamento aplicado apenas em seu plano médio. A Figura 2 exemplifica o elemento de chapa.

Figura 2 - Elemento estrutural de chapa.



Fonte: Sales (2005).

Nesse momento se faz necessária a correta diferenciação entre o que são elementos de chapa e o que são elementos de placa. Sales (2005) afirma que as placas diferem das chapas não só pela espessura, que costuma ser menor nas chapas, mas principalmente pela forma de carregamento, sendo as placas solicitadas por carregamento perpendicular ao seu plano médio, enquanto que nas chapas o carregamento está contido neste plano médio. Além disso, é importante salientar que no mercado siderúrgico em geral a classificação é diferente, sendo que as placas se diferenciam das chapas apenas pela espessura, uma vez que as placas são elementos com espessura maior ou igual a 4”(100mm), e as chapas são elementos mais finos.

2.4 Estado Plano de Tensão e Estado Plano de Deformação

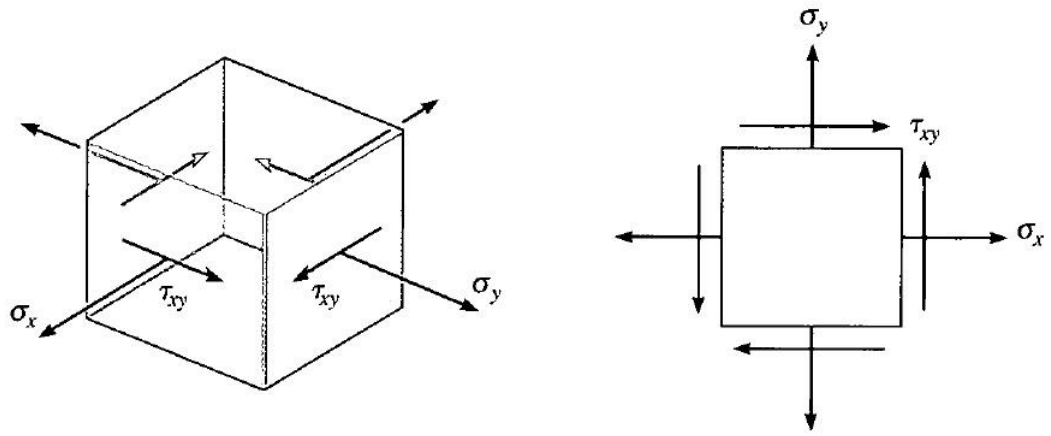
Para Waidemam (2008) os estados planos de tensão e de deformação surgiram de simplificações impostas ao equacionamento tridimensional da teoria da elasticidade, sendo que estes podem ser aplicados aos problemas sob determinadas restrições.

Ainda segundo Waidemam (2008) para os problemas do estado plano de deformação, uma componente de deformação normal e duas componentes de deformação por cisalhamento são nulas. Já os problemas do estado plano de tensão são caracterizados por distribuições planas das tensões no corpo, isto é, uma componente de tensão normal e duas componentes de tensões cisalhantes são nulas.

As tensões atuantes em um elemento de chapa podem ser representadas por meio do estado plano de tensões. Segundo Hibbeler (2010) na prática da engenharia é comum apresentar as tensões atuantes em um corpo por meio do estado plano. Nesse estado, as tensões atuantes são representadas pela combinação de duas componentes de tensões normais e uma componente de tensão de cisalhamento, orientadas de acordo com o sistema de coordenadas escolhido.

Dessa maneira, se não houver carregamento atuando na superfície do corpo, as componentes de tensões serão nulas nas faces dos elementos que estiverem orientadas de acordo com essa superfície. Por consequência, as faces opostas a essa superfície também terão tensões nulas para garantir o equilíbrio, caracterizando assim a atuação de tensões em apenas um plano, como pode ser observado na Figura 3.

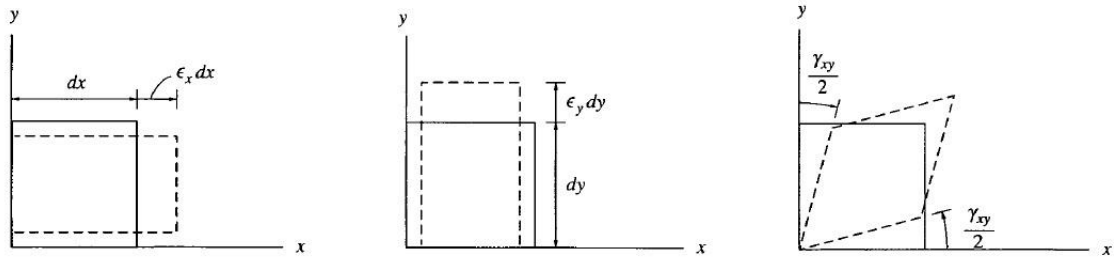
Figura 3 - Caracterização do estado plano de tensão atuando em um elemento infinitesimal de volume.



Fonte: Hibbeler (2010).

Hibbeler (2010) ainda destaca que, da mesma forma que ocorre com as tensões, para as deformações também é comum na engenharia se utilizar o estado plano. Dessa maneira, um determinado elemento estará sujeito somente a duas componentes de deformação normal e uma componente de deformação por cisalhamento. As deformações normais são caracterizadas por mudanças no comprimento do elemento, enquanto que a deformação de cisalhamento é caracterizada por mudanças na angulação entre as faces do elemento. A Figura 4 mostra um elemento sujeito a um estado plano de deformações.

Figura 4 - Caracterização do estado plano de deformação atuando em um elemento infinitesimal de volume.



Fonte: Hibbeler (2010).

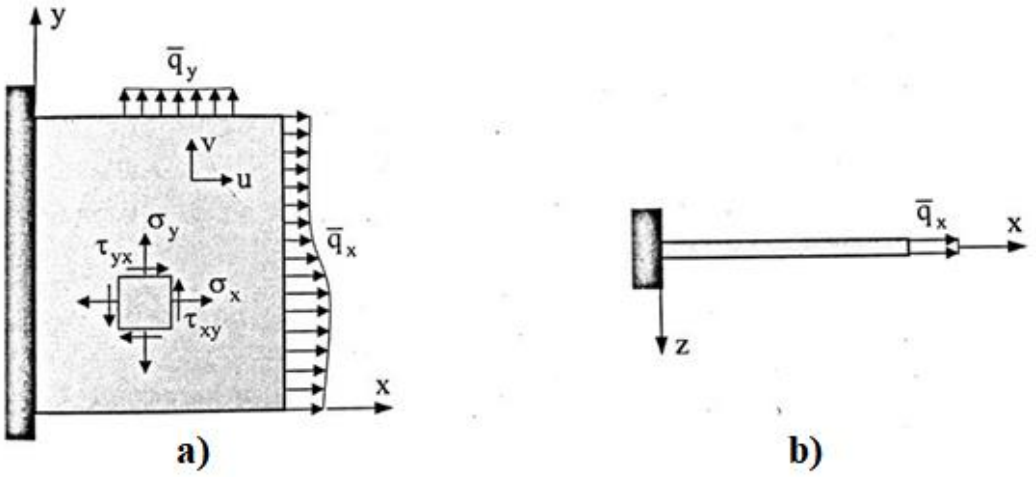
Soriano (2009) diz que o estado plano de tensão se caracteriza em um corpo de espessura pequena em relação ao seu plano médio, de referencial *xy*, quando as ações externas atuam somente nesse plano, gerando o seguinte estado de tensão:

$$\begin{cases} \sigma_x \neq 0\ , \quad \sigma_y \neq 0\ , \quad \sigma_z = 0 \\ \tau_{xy} \neq 0\ , \quad \tau_{xz} = 0\ , \quad \tau_{yz} = 0 \end{cases} \tag{1}$$

Savassi (1996) destaca que o estado de tensão apresentado em (1) é típico de elementos estruturais de superfície, tais como as chapas (vigas-parede). Além disso, o autor ainda destaca a utilização desse estado no caso de vigas comuns, dizendo que estas podem ser estudadas como um caso particularizado dentro do campo das chapas. Ribeiro (2004) também afirma que os problemas do estado plano de tensões são característicos de estruturas de chapas planas carregadas em seu próprio plano.

No caso das chapas, objeto de estudo deste trabalho, o carregamento atua de forma contida no plano do referencial *xy*, como mostra a Figura 5. No caso, os deslocamentos nas direções *x* e *y*, respectivamente chamados *u* e *v*, são os graus de liberdade da estrutura, sendo chamados dentro do MEF de variáveis primárias, enquanto que as componentes de tensão e deformação são as variáveis dependentes secundárias.

Figura 5 - Chapa tracionada caracterizando o estado plano de tensões: (a) vista frontal; (b) vista superior.

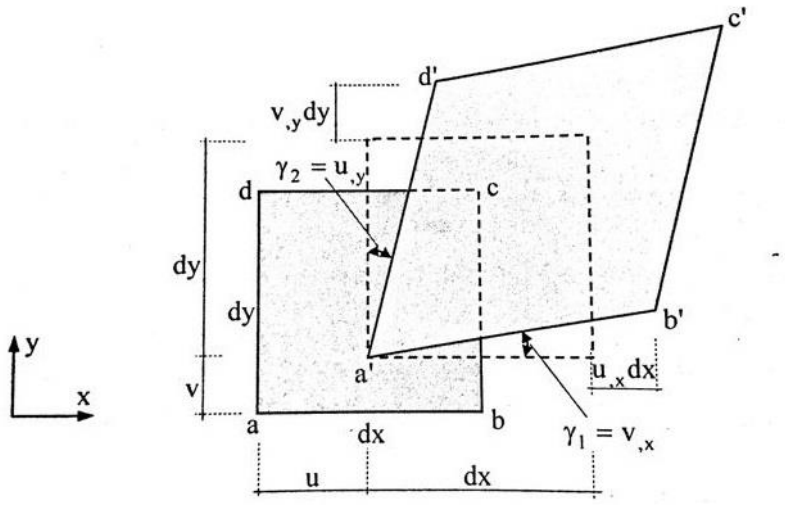


Fonte: Soriano (2009).

2.4.1 Relações entre Deslocamentos e Deformações

Soriano (2009) mostra como os deslocamentos são relacionados com as deformações existentes utilizando o exemplo de um elemento infinitesimal retangular. Como mostrado na Figura 6, o elemento se desloca de um estado *abcd* para um estado *a'b'c'd'*, sofrendo deformações normais e angular no plano *xy*, sendo que os deslocamentos *u* e *v* são os deslocamentos do ponto *a* nas direções *x* e *y*, respectivamente.

Figura 6 - Estado plano de deformação em um elemento infinitesimal.



Fonte: Soriano (2009).

Os incrementos infinitesimais dos deslocamentos *u* e *v*, de acordo com a Figura 6, são dados pelas seguintes equações:

$$du = u,_x \cdot dx \qquad dv = v,_y \cdot dy \tag{2}$$

Desta forma, utilizando as definições de deformações normais e angular, e as relações indicadas acima, podem ser escritas as seguintes equações:

$$\varepsilon_x = \frac{(dx + du) - dx}{dx} \qquad \varepsilon_y = \frac{(dy + dv) - dy}{dy} \tag{3}$$

$$\varepsilon_x = \frac{du}{dx} \qquad \varepsilon_y = \frac{dv}{dy} \tag{4}$$

$$\gamma_1 = Tg(\gamma_1) = \frac{dv}{dx} \quad \gamma_2 = Tg(\gamma_2) = \frac{du}{dy} \tag{5}$$

$$\gamma_{xy} = \gamma_{yx} = \gamma_1 + \gamma_2 = \frac{dv}{dx} + \frac{du}{dy} \tag{6}$$

$$\gamma_{xy} = \frac{dv}{dx} + \frac{du}{dy} \tag{7}$$

A partir de então as equações (4) e (7) podem ser colocadas na forma matricial, relacionando as deformações do elemento com os respectivos deslocamentos nodais, obtendo-se:

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \cdot \begin{Bmatrix} u \\ v \end{Bmatrix} \tag{8}$$

$$\{\varepsilon\} = [L] \cdot \{u\} \tag{9}$$

Onde, $\{\varepsilon\}$ = vetor de deformações nodais;

$[L]$ = matriz de operadores diferenciais;

$\{u\}$ = vetor de deslocamentos nodais;

2.5 Lei de Hooke Generalizada

A Lei de Hooke é uma importante relação da resistência dos materiais que estabelece a linearidade entre as tensões e as deformações atuantes em um corpo, quando o material que o compõe trabalha em regime elástico. Essa linearidade entre tensão e deformação ocorre em função do módulo de elasticidade do material (E), também chamado de Módulo de Young, sendo estabelecida pela equação (10).

$$\sigma = E \cdot \varepsilon \tag{10}$$

Hibbeler (2010) mostra que, uma vez estabelecido os princípios gerais de tensão e deformação no plano, é possível relaciona-los com propriedades dos materiais através da chamada Lei de Hooke generalizada. Para isso, é necessário considerar o material homogêneo, isotrópico e com comportamento linear elástico.

Dessa maneira, para um estado plano de tensão, utilizando-se do princípio da superposição dos efeitos, da definição do coeficiente de Poisson e da Lei de Hooke, é possível determinar as relações dadas pelas equações (11), (12) e (13).

$$\varepsilon_x = \frac{1}{E} \cdot \left[\sigma_x - \nu(\sigma_y) \right] \tag{11}$$

$$\varepsilon_y = \frac{1}{E} \cdot \left[\sigma_y - \nu(\sigma_x) \right] \tag{12}$$

$$\gamma_{xy} = \frac{1}{G} \cdot \tau_{xy} \tag{13}$$

Além disso, Hibbeler (2010) mostra que para materiais isotrópicos, o módulo de elasticidade (E) se relaciona ao módulo de elasticidade ao cisalhamento, também chamado módulo de elasticidade transversal (G). Essa relação é dada pela equação (14):

$$G = \frac{E}{2 \cdot (1 + \nu)} \tag{14}$$

Assim sendo, todas essas relações podem ser utilizadas de modo a caracterizar os efeitos ocorrentes devido ao carregamento em uma estrutura de chapa, uma vez que esta pode ser entendida como um elemento sujeito a um estado plano de tensão.

2.5.1 Relações entre Deformações e Tensões

Soriano (2009) diz que um material é homogêneo quando suas propriedades são as mesmas independente do ponto considerado no sólido e é isótropo quando essas propriedades não dependem da direção considerada. Além disso, o material pode ser elástico ou não e pode ser linear ou não. Ele ainda destaca que um material elástico linear é aquele, que quando submetido à um estado uniaxial de tensão, tem relação tensão x deformação definido pela equação (10).

Porém, mesmo que um corpo esteja submetido a tensão normal apenas em uma direção, ele vai sofrer deformações normais na direção de atuação da tensão e nas direções perpendiculares à esta. Isto ocorre devido ao chamado efeito de Poisson. Portanto, um corpo que esteja submetido a um estado plano de tensão estará sujeito a deformações normais ε_x e ε_y, além da deformação angular γ_{xy}. E assim como já foi mostrado anteriormente, essas deformações podem ser relacionadas com as tensões atuantes através da Lei de Hooke Generalizada. Dessa forma, utilizando a equação (14), é possível expressar as equações (11), (12) e (13) na forma matricial mostrada nas equações (15) e (16):

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \frac{1}{E} \cdot \begin{bmatrix} 1 & -\nu & 0 \\ -\nu & 1 & 0 \\ 0 & 0 & 2(1+\nu) \end{bmatrix} \cdot \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} \tag{15}$$

$$\{\varepsilon\} = [C] \cdot \{\sigma\} \tag{16}$$

Sendo $[C]^{-1}$ a matriz inversa da matriz $[C]$, definida pela equação (17):

$$[C]^{-1} = \frac{E}{(1-\nu^2)} \cdot \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \dfrac{(1-\nu)}{2} \end{bmatrix} \tag{17}$$

Pré-multiplicando os dois membros da equação (16) por $[C]^{-1}$ e definindo-se a matriz $[E]$, tem-se:

$$[C]^{-1} \cdot \{\varepsilon\} = [C]^{-1} \cdot [C] \cdot \{\sigma\} \tag{18}$$

$$[C]^{-1} = [E] \tag{19}$$

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = \frac{E}{(1-\nu^2)} \cdot \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{(1-\nu)}{2} \end{bmatrix} \cdot \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \tag{20}$$

$$\{\sigma\} = [E] \cdot \{\varepsilon\} \tag{21}$$

2.6 Princípio dos Trabalhos Virtuais

Os métodos existentes de análise estrutural sempre têm como objetivo a determinação de deslocamentos e de esforços internos atuantes na estrutura. Em alguns desses métodos o princípio da conservação de energia aparece como formulação básica para equacionar o equilíbrio da estrutura.

Martha (2010) mostra que o princípio da conservação de energia pode ser entendido como um balanço de energia, no qual a soma algébrica de todo o trabalho produzido pelas forças aplicadas com a energia de deformação armazenada na estrutura deve resultar em um valor nulo. Considerando algumas hipóteses, tais como que o único tipo de energia armazenado na estrutura é a energia de deformação elástica, que não há perda de energia e que a estrutura tem comportamento elástico linear, o princípio de conservação de energia pode ser equacionado como segue:

$$We = U \tag{22}$$

Onde, We = trabalho realizado pelas forças externas;

U = energia de deformação interna da estrutura;

A energia de deformação interna pode ser entendida como o trabalho realizado pelas tensões internas enquanto o corpo se deforma. Assim, um elemento infinitesimal de volume de um componente estrutural, que está submetido à diferentes parcelas de tensões e deformações, armazena uma quantidade energia de deformação interna, chamada uo. Matematicamente essa energia pode ser escrita em função das tensões $(\sigma,\ \tau)$ e deformações $(\varepsilon,\ \gamma)$. Porém, utilizando as relações já conhecidas entre as deformações e deslocamentos $(du, d\theta, dv, d\varphi)$, bem como entre as tensões e esforços internos $(N,\ M,\ Q,\ T)$, é possível determinar uo em função destes deslocamentos e esforços internos.

A partir disso é possível integrar a equação de energia uo no volume da estrutura para obter a energia de deformação interna total, chamada U. Por fim, utilizando a equação (22) é possível determinar os deslocamentos estruturais provocados pela ação do carregamento externo. Martha (2010) afirma ainda que o princípio da conservação de energia acaba se tornando limitado pois só consegue fornecer o valor do deslocamento de um ponto da estrutura e esse ponto necessariamente é o ponto em que a força está aplicada. Além disso, para estruturas que tenham mais de uma força aplicada o princípio não pode ser utilizado, pois haveria somente uma equação para se determinar mais de um deslocamento (incógnita), o que torna o problema matematicamente indeterminado. A mesma limitação ocorre para se descobrir o giro em estruturas com momentos aplicados. A solução para isso é a generalização do princípio para o chamado princípio dos trabalhos virtuais (PTV).

Martha (2010) destaca que para a formulação do PTV é necessário a definição de dois tipos de sistema. O primeiro sistema é formado por $(F_a,\ \sigma_a)$, sendo F_a o campo de forças externas e σ_a o campo de tensões internas, sendo necessário que haja equilíbrio entre eles. O segundo sistema é formado por $(D_b,\ \varepsilon_b)$, onde D_b é o campo de deslocamentos externos e ε_b é o campo de deformações internas, sendo necessário que haja compatibilidade entre eles.

Esses dois sistemas são independentes entre si, isto é, não existe uma relação de causa-efeito entre eles. O balanço entre trabalho externo e energia de deformação interna, combinado com essa relação de independência entre os sistemas, tem como resultado a formulação do PTV, estabelecida pela equação (23):

$$We = U \rightarrow \sum F_a \cdot D_b = \int \sigma_a \cdot \varepsilon_b \tag{23}$$

O lado esquerdo da equação (23) representa o trabalho virtual externo, enquanto que o lado direito representa a energia de deformação interna virtual. O termo virtual se aplica justamente pela relação de independência entre forças e deslocamentos externos e entre tensões e deformações internas.

Da mesma forma que ocorre para o princípio da conservação de energia, no princípio dos trabalhos virtuais a energia de deformação interna virtual pode ser escrita em função esforços internos virtuais (f_a) e seus respectivos deslocamentos internos virtuais (d_b), sendo então estabelecida a equação (24):

$$We = U \rightarrow \sum F_a \cdot D_b = \int f_a \cdot d_b \tag{24}$$

Assim sendo, (F_a) é o campo de forças externas (solicitações e reações) que deve estar em equilíbrio com o campo (f_a) de esforços internos $(Na,\ Ma,\ Qa,\ Ta)$. Enquanto que (D_b) é o campo de deslocamentos externos que deve estar compatível com o campo (d_b) de deslocamentos relativos internos $(dub,\ d\theta b,\ dvb,\ d\varphi b)$.

Portanto, Martha (2010) afirma que esse princípio pode ser utilizado para impor condições de compatibilidade. Nesse momento, o PTV se divide em duas vertentes, o Princípio das Forças Virtuais e o Princípio dos Deslocamentos Virtuais.

No princípio das forças virtuais se escolhe arbitrariamente um sistema de forças $(F_a,\ f_a)$ que esteja em equilíbrio, denominado virtual, para então determinar uma configuração deformada real qualquer $(D_b,\ d_b)$.

No princípio dos deslocamentos virtuais se escolhe arbitrariamente um sistema de deslocamentos $(D_b,\ d_b)$ que seja compatível, denominado virtual, para então determinar um sistema de forças real qualquer $(F_a,\ f_a)$.

2.7 Método dos Elementos Finitos

2.7.1 Introdução ao Método dos Elementos Finitos

Logan (2007) diz que o MEF é um método numérico voltado para resolução de problemas de engenharia e física onde as soluções analíticas são complexas. Ele destaca que as soluções analíticas são aquelas dadas por expressões matemáticas que na maioria das vezes requerem a solução de equações diferenciais parciais ou ordinárias. Essas soluções são capazes de fornecer valores das variáveis estudadas em qualquer parte do corpo, ou seja, em um número infinito de pontos.

Soriano (2009) diz que métodos aproximados de resolução de modelos matemáticos se tornam extremamente importantes na engenharia a partir do instante em que as soluções analíticas ficam inviáveis na hora de analisar o comportamento de sistemas físicos. A ideia é sempre utilizar um método aproximado que substitua os infinitos graus de liberdade do modelo contínuo por um número finito de graus de liberdade em um modelo discreto, tendo dessa forma a troca de um sistema de equações diferenciais por um sistema de equações algébricas. Alguns desses métodos aproximados são Método das Diferenças Finitas, o Método dos Elementos de Contorno e o Método dos Elementos Finitos. Porém, o MEF se destaca dos demais devido as suas facilidades de generalização, programação e uso.

Waidemam (2004) também afirma que os métodos discretos realizam a aproximação da solução exata dos sistemas contínuos através da utilização de sistemas discretos que contém um número finito de graus de liberdade.

Ainda de acordo com Soriano (2009) o MEF é um método genial, que consegue analisar o comportamento de qualquer sistema físico regido por equações diferenciais ou integrais, sejam estes sistemas físicos da mecânica dos sólidos deformáveis, da condução de calor e massa e até mesmo do eletromagnetismo.

Fish (2009) destaca que a análise de tensões em estruturas requer a solução de equações diferenciais parciais, que são complexas de serem determinadas pelos métodos clássicos, exceto para estruturas simples, o que nem sempre é o suficiente dentro da engenharia. Dessa forma, o MEF tem como objetivo determinar a solução aproximada dessas equações quando se trabalha com estruturas mais complexas.

Fish (2009) ainda afirma que dentro da análise de tensões, o uso de elementos finitos lineares requer o comportamento elástico-linear do material e requer que os deslocamentos sejam pequenos. Mas em geral, dentro da engenharia, essas condições são satisfeitas, uma vez que não é desejável trabalhar com carregamentos que possam levar o material ao comportamento não linear ou então ao surgimento de grandes deformações.

Para Waidemam (2004) o primeiro passo na aplicação do MEF é dividir a estrutura em um número finito de elementos com dimensões adequadas. Então, os deslocamentos dos pontos nodais desses elementos são generalizados em função das suas coordenadas. Posteriormente, os deslocamentos ao longo do elemento são expressos em função dos deslocamentos nodais através de funções específicas, chamadas de funções de forma.

Então, a partir dos deslocamentos são determinadas as deformações e tensões nos pontos nodais da estrutura. Gesualdo (2010) afirma que um modelo completo de elementos finitos corresponde a associação de diversos elementos. Assim, no processo de cálculo, se garante apenas que os deslocamentos relativos aos graus de liberdade dos nós de conectividade, entre elementos vizinhos, são iguais. Enquanto que as deformações são específicas para cada elemento e dependem do seu comportamento interno.

Então, o MEF é um método numérico cujo funcionamento é bem coerente com o seu nome. Na engenharia de estruturas, ele funciona dividindo uma determinada estrutura em um número finito de pequenas partes, denominadas elementos. Os elementos são então delimitados por pontos específicos, que são chamados de nós. Cada nó possui um número finito de graus de liberdade, isto é, de possíveis movimentos no espaço tridimensional. Após realizada a divisão, o método consiste em utilizar funções aproximadoras que interpolam os deslocamentos no elemento em função dos deslocamentos nodais. Por fim, a partir da determinação dos deslocamentos é possível determinar valores de forças nodais, tensões e deformações, e com isso analisar o comportamento da estrutura. Logan (2007) destaca que no Método dos Elementos Finitos, em vez de resolver o problema todo de uma só vez, as equações são formuladas para cada elemento, para depois combinar todos os elementos e obter a solução geral do problema.

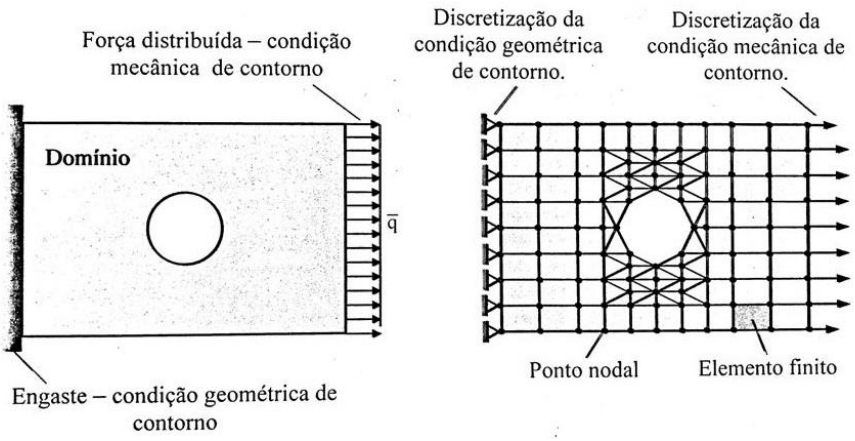
2.7.2 Definição do Modelo Discreto e da Malha de Elementos

O MEF parte do princípio de leis simples para as variáveis dependentes primárias em subdomínios chamados elemento finitos. Esses elementos são conectados entre si por pontos nodais, formando uma malha com um número finito de pontos, sendo esse processo conhecido como discretização do modelo contínuo, dando origem então à um modelo discreto de análise do sistema físico. A solução para o comportamento desse sistema será mais aproximada da solução exata à medida que se refina a malha utilizada.

Waidemam (2004) diz que a discretização é um dos conceitos necessários para a formulação do MEF. Para o autor, a discretização consiste em transformar um sistema estrutural contínuo em um sistema estrutural discreto, através da divisão do modelo em pequenas regiões, chamadas de elementos finitos.

Soriano (2009) destaca que os elementos finitos podem ser uni, bi ou tridimensionais, das mais variadas formas e tamanhos, e que também se diferenciam em quantidade de pontos nodais e faces. Além disso, existem diferentes números e tipos de grau de liberdade em cada um dos pontos nodais. No presente trabalho, os elementos serão do tipo *CSQ*, isto é, elementos bidimensionais de formato retangular, com pontos nodais apenas em seus vértices e possuindo dois graus de liberdade em cada um, sendo estes graus de liberdade o deslocamento horizontal e o deslocamento vertical do nó. A Figura 7 exemplifica uma chapa sendo submetida a carregamento de tração e com discretização do meio físico realizada por elementos finitos retangulares e triangulares, a chamada discretização mista.

Figura 7 - Discretização de uma chapa tracionada.

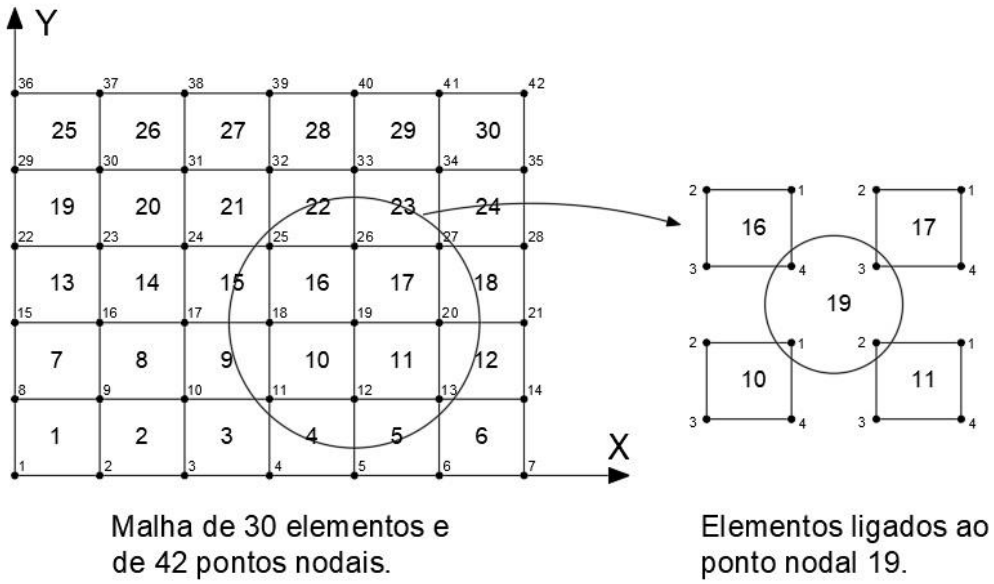


Fonte: Soriano (2009).

Soriano (2009) diz que os elementos interagem entre si por meio de leis que interpolam parâmetros nodais de variáveis primárias no domínio do elemento, como os deslocamentos por exemplo. Essas leis são chamadas funções de interpolação e deve haver uma função de interpolação para cada parâmetro nodal.

Para automatização do método, os pontos nodais e os elementos devem ter posições definidas em relação a um referencial, devendo ser numerados em forma sequencial como mostra a Figura 8:

Figura 8 - Numeração sequencial dos elementos e pontos nodais.



Fonte: Soriano (2009).

A resolução do modelo discreto é completada com a especificação das condições de contorno, das propriedades dos materiais, das características geométricas do elemento e das ações externas. Com essas definições é possível elaborar para cada elemento a sua matriz de rigidez e o vetor de forças nodais, para então impor a igualdade com os parâmetros nodais e montar o sistema de equações algébricas que regem o comportamento do sistema. Feito isso para cada elemento, é possível então montar as equações para toda a estrutura estudada.

Gesualdo (2010) destaca que a vantagem do MEF é justamente transformar um problema global (mais complexo) em um problema local. Estabelecendo-se um modelo numérico adequado, é possível chegar na matriz de rigidez global apenas fazendo o acoplamento das matrizes de rigidez individuais de cada elemento.

Soriano (2009) destaca que a escolha do modelo discreto à ser utilizado deve seguir algumas etapas. Para ele, a melhor estratégia é começar com um modelo simples e depois sofisticá-lo de acordo com as necessidades. Além disso, um modelo só fica definido com a especificação de dados como os pontos nodais (coordenadas, forças concentradas, restrições de graus de liberdade) e os elementos (tipo, propriedades geométricas e do material, ações externas e comportamentos das ligações).

Logan (2007) também afirma que ao utilizar o programa o usuário deve informar a malha que deverá ser utilizada, bem como o tipo de elemento finito e a forma como eles estão conectados entre si. Além disso, é necessário incluir informações como coordenadas dos pontos nodais, propriedades do material, forças aplicadas e condições de contorno do problema.

Os pontos nodais deverão estar em posições estratégicas da estrutura, como por exemplo, em pontos onde há força concentrada e em regiões onde há mudança de material ou de espessura, isto por que não é usual a não homogeneidade em um mesmo elemento finito. Também são definidos pontos nodais em locais onde se deseja conhecer deslocamentos e tensões. Os demais pontos nodais são definidos de forma a procurar manter uma densidade e uniformidade adequada na malha. Nos casos em que a geração da malha é feita de forma automática, é necessário ajusta-la manualmente de forma que os pontos nodais possam coincidir com os pontos estratégicos, como os que possuem forças concentradas.

Para modelagem bidimensional os elementos retangulares e triangulares se apresentam como os mais adequados, sendo que eles podem ter diferentes quantidades e posicionamentos dos pontos nodais. Inclusive, é muito usual combinar elementos retangulares e triangulares, formando uma malha mista. Quanto maior o número de elementos em uma malha, sejam eles de qualquer tipo, maior é o refinamento desta. Em regiões com elevadas tensões é adequado que se tenha um refinamento maior da malha, para que os resultados sejam mais próximos dos reais. O refinamento da malha pode ser feito aumentando o número de elementos ou aumentando o número de pontos nodais em cada elemento.

Mesmo com maior refinamento, o campo de tensões nunca satisfaz todas as condições de equilíbrio no interior e nas fronteiras dos elementos finitos. Além disso, como as tensões são obtidas a partir das derivadas dos deslocamentos, as aproximações obtidas para as tensões são sempre menos satisfatórias do que as aproximações obtidas para os deslocamentos. (PEREIRA, 2005)

Pontos onde há a aplicação de forças concentradas são considerados pontos de elevada concentração de tensão, bem como nas regiões próximas aos apoios. Para elementos bidimensionais é usual que as forças concentradas sempre estejam aplicadas em um ponto nodal. Além disso, essa força deve ter mesma direção de um dos graus de liberdade existente, isto é, na direção dos deslocamentos nodais.

As condições geométricas de contorno se referem ao comportamento dos pontos nodais que estão conectados de alguma forma em uma restrição de um, ou mais, dos seus graus de liberdade. Essas restrições de deslocamentos nodais e consequente redução no número de graus de liberdade auxilia na modelagem e no condicionamento da matriz de rigidez na resolução do sistema.

Soriano (2009) diz que para uma melhor organização e facilidade na execução computacional, tanto os pontos nodais quanto os elementos devem possuir uma numeração em forma sequencial partindo sempre do número 1. Sendo que os pontos nodais devem possuir uma numeração sequencial global em relação ao referencial *xy* e também uma numeração sequencial local, isto é, uma numeração dentro do próprio elemento em que ele se encontra. Essa numeração local também deve partir do número 1, em um vértice qualquer, e percorrer de forma sequencial os demais vértices do elemento no sentido anti-horário.

Soriano (2009) ainda diz que a validação dos resultados deve ser feita pelo engenheiro responsável pela análise via MEF, que deve ter bom senso, conhecimento e experiência para interpretar os dados obtidos, tendo em vista que esses resultados são apenas uma aproximação do modelo matemático contínuo. Ele diz que é preciso sempre ter uma desconfiança em relação aos resultados apresentados, pois pode haver erros tanto por parte do usuário quanto por parte da discretização ou do arredondamento de números. Por isso, comparar os resultados obtidos com outros programas deve ser uma prática frequente na hora de analisar as estruturas.

2.7.3 Formulação do Método dos Elementos Finitos Aplicado ao Problema de Chapas

Soriano (2009) diz que existem diferentes métodos que subsidiam a formulação do MEF, entre eles cita o Método de Rayleigh-Ritz, o Método de Galerkin e o Princípio dos Deslocamentos Virtuais. Este último por sua vez, que como já foi visto é uma derivação do Princípio dos Trabalhos Virtuais, será o método utilizado nas deduções do MEF neste trabalho.

Partindo então do Princípio dos Deslocamentos Virtuais, simulando uma chapa submetida à um estado plano de tensões, de volume V e espessura uniforme t, e com carregamento de forças concentradas Fi em um ou mais de seus pontos. Sendo $(u_i^*,\ \varepsilon^*)$ o sistema formado pelo campo de deslocamentos externos virtuais (u_i^*) e pelo campo de deformações internas virtuais (ε^*), compatíveis entre si e representados por (*) para denotar que se trata de valores virtuais.

De acordo com as definições já mostradas para o estado plano de tensão tem-se definido os seguintes vetores:

$$\{\sigma\} = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} \qquad \{\varepsilon\}^* = \begin{Bmatrix} \varepsilon_x^* \\ \varepsilon_y^* \\ \gamma_{xy}^* \end{Bmatrix} \tag{25}$$

Então, a partir da equação (23), que define o Princípio dos Deslocamentos Virtuais, é possível igualar o trabalho gerado pelas foças externas aplicadas com a energia interna ao longo de todo o volume do corpo, estabelecendo-se a equação (26):

$$\sum \{F_i\} \cdot \{u_i\}^* = \int_V \{\varepsilon\}^{*T} \cdot \{\sigma\} \cdot dV \tag{26}$$

Onde, $\{F_i\}$ = vetor de forças externas reais aplicadas nos pontos nodais *i*;

$\{u_i\}^*$ = vetor de deslocamentos externos virtuais dos respectivos pontos nodais *i*;

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

$\{\sigma\}$ = vetor de tensões internas reais;

Como a espessura *t* da chapa se mantém constante ao longo de todo o seu comprimento, o infinitesimal de volume (dV) pode ser escrito em função do infinitesimal de área (dA) do elemento, de acordo com a equação (27):

$$\int_V dV = t \cdot \int_A dA \tag{27}$$

Na formulação do Método dos Elementos Finitos, os deslocamentos de cada elemento (u, v) podem ser relacionados com os deslocamentos nodais (u_i, v_i) deste elemento através de funções de aproximação, chamadas funções de forma (ϕ). No caso, o elemento *CSQ* utilizado nesse trabalho possui 4 pontos nodais (um em cada vértice), de tal forma que é possível escrever:

$$\{u\} = \begin{Bmatrix} u \\ v \end{Bmatrix} \tag{28}$$

$$\{u_i\} = \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{Bmatrix} \tag{29}$$

$$[\phi] = \begin{bmatrix} \phi_1 & 0 & \phi_2 & 0 & \phi_3 & 0 & \phi_4 & 0 \\ 0 & \phi_1 & 0 & \phi_2 & 0 & \phi_3 & 0 & \phi_4 \end{bmatrix} \tag{30}$$

$$\{u\} = [\phi] \cdot \{u_i\} \tag{31}$$

A partir de então, substituindo a equação (31) nas relações entre deslocamentos e deformações já demonstradas anteriormente através da equação (9), é possível escrever que:

$$\{\varepsilon\} = [L] \cdot [\phi] \cdot \{u_i\} \tag{32}$$

Nesse instante, é possível definir uma matriz, nesse caso chamada por $[B]$, formada pelo produto entre as matrizes de operadores diferenciais e de funções de forma, assim sendo:

$$[B] = [L] \cdot [\phi] \tag{33}$$

$$\{\varepsilon\} = [B] \cdot \{u_i\} \tag{34}$$

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

$$\{\varepsilon\}^* = [B] \cdot \{u_i\}^* \tag{35}$$

Assim, substituindo a equação (34) nas relações entre tensões e deformações já demonstradas na equação (21), é possível escrever a equação (36):

$$\{\sigma\} = [E] \cdot [B] \cdot \{u_i\} \tag{36}$$

Além disso, utilizando as definições de matriz transposta e as propriedades do produto de matrizes, é possível reescrever a equação (35) da seguinte maneira:

$$\{\varepsilon\}^{*T} = \{u_i\}^{*T} \cdot [B]^T \tag{37}$$

Como já foi dito anteriormente, o elemento *CSQ* utilizado na formulação do MEF no presente trabalho possui 4 pontos nodais, sendo cada um desses pontos um dos vértices do retângulo. Assim sendo, cada ponto nodal pode estar submetido ou não a forças, tanto na direção *x* quanto na direção *y*, que provocarão deslocamentos nas direções de *u* e *v*, respectivamente. Dessa forma, o vetor de forças nodais $\{F_i\}$ e o vetor de deslocamentos externos virtuais $\{u_i\}^{*T}$ podem ser escritos, respectivamente, da seguinte maneira:

$$\{F_i\} = \begin{Bmatrix} F_{1x} \\ F_{1y} \\ F_{2x} \\ F_{2y} \\ F_{3x} \\ F_{3y} \\ F_{4x} \\ F_{4y} \end{Bmatrix} \tag{38}$$

$$\{u_i\}^{*T} = \begin{Bmatrix} u_1^* & v_1^* & u_2^* & v_2^* & u_3^* & v_3^* & u_4^* & v_4^* \end{Bmatrix} \tag{39}$$

Portanto, substituindo as equações (27), (36), (37), (38) e (39) na equação (26) do PTV, e sabendo que os deslocamentos, tanto reais quanto virtuais são constantes durante o processo de integração, tem-se:

$$\{F_i\} \cdot \{u_i\}^{*T} = t \cdot \int_A \{u_i\}^{*T} \cdot [B]^T \cdot [E] \cdot [B] \cdot \{u_i\} \cdot dA \tag{40}$$

$$\{F_i\} \cdot \{u_i\}^{*T} = \{u_i\}^{*T} \cdot t \cdot \left[\int_A \cdot [B]^T \cdot [E] \cdot [B] \cdot dA \right] \cdot \{u_i\} \tag{41}$$

$$\{F_i\} = t \cdot \left[\int_A \cdot [B]^T \cdot [E] \cdot [B] \cdot dA \right] \cdot \{u_i\} \tag{42}$$

Assim sendo, define-se a matriz de rigidez $[k]$ para o elemento de acordo com a equação (43):

$$[k] = t \cdot \left[\int_A \cdot [B]^T \cdot [E] \cdot [B] \cdot dA \right] \tag{43}$$

Por fim, a equação de equilíbrio do PTV se reduz na equação (44):

$$\{F\} = [k] \cdot \{d\} \tag{44}$$

Onde, $\{F\}$ = vetor de forças reais nos pontos nodais do elemento;

$[k]$ = matriz de rigidez do elemento;

$\{d\}$ = deslocamentos reais nodais do elemento;

2.7.3.1 Elemento CSQ

O elemento *CSQ* (*Constant Strain Quadrilateral*) é um tipo de elemento utilizado na formulação do MEF, o qual pertence à classe dos chamados elementos bidimensionais. Esse elemento é caracterizado pelo seu formato retangular e pelo fato de possuir somente 4 pontos nodais, um em cada um dos seus vértices.

O elemento *CSQ* possui dois graus de liberdade em cada um dos nós, sendo esses graus os deslocamentos horizontais e verticais. Como já foi mostrado anteriormente, a formulação do MEF pode ser derivada da equação do PTV, relacionando então os deslocamentos nodais com as forças nodais através da chamada matriz de rigidez $[k]$. Essa matriz será sempre uma matriz simétrica e quadrada, e sua ordem é o produto entre a quantidade de nós do elemento e a quantidade de graus de liberdade existentes. No caso do elemento *CSQ*, como este possui 4 nós e 2 graus de liberdade por nó, a matriz de rigidez de cada elemento terá ordem oito.

Nesse instante, se faz necessária uma observação, pois a equação (44) foi deduzida para apenas um elemento finito. Então, para a obtenção do sistema global de equações é necessário que se tenha o sistema de equações locais de cada elemento para que então se faça a sobreposição das matrizes. Dessa forma, na interface dos elementos, isto é, nos pontos nodais em comum, é necessário que se faça a soma de cada coeficiente das matrizes, adequando-os na posição correta dentro da matriz do sistema global.

A matriz de rigidez global $[K]$ por sua vez também é uma matriz quadrada, sendo que sua ordem é dada pelo produto entre a quantidade de pontos nodais na malha de elementos e a quantidade de graus de liberdade existentes.

Soriano (2009) destaca que para a formulação do MEF utiliza-se a chamada formulação dos deslocamentos. Essa formulação é desenvolvida a partir de um campo de deslocamentos, o qual é expresso pela equação (31). Como pode ser observado nessa equação, os deslocamentos em qualquer ponto do elemento são escritos em função dos deslocamentos nodais desse mesmo elemento. As funções que realizam essa relação entre deslocamentos são chamadas funções de interpolação.

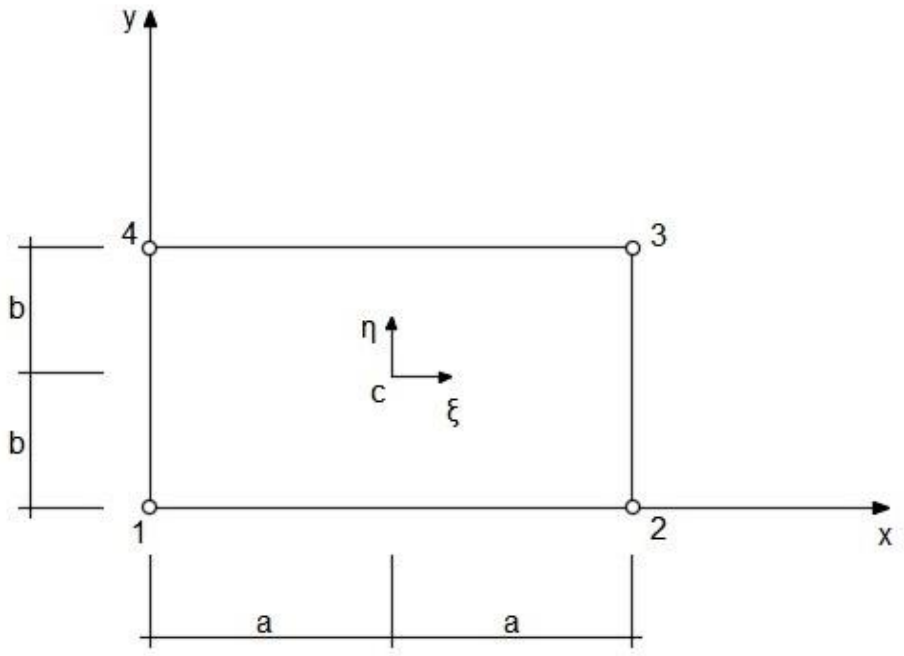
Soriano (2009) diz que para o caso dos elementos quadrilaterais (os quais o elemento retangular *CSQ* é um dos tipos) se torna mais prático construir diretamente as funções de interpolação através de coordenadas locais normalizadas ortogonais (ξ, η).

Savassi (1996) também afirma que no caso dos elementos retangulares as coordenadas adimensionais baricêntricas $(\xi,\ \eta)$ são mais cômodas de se trabalhar do que as coordenadas cartesianas $(x,\ y)$. Pois dessa forma, a integração de determinada função $f\ (x,y)$ será no próprio domínio do retângulo, assim como mostra a equação:

$$\iint_A f(x,y)dA = a \cdot b \cdot \int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta)\cdot d\xi \cdot d\eta \tag{45}$$

A comodidade oferecida por essas coordenadas é ilustrada a partir da Figura 9, a qual mostra um elemento *CSQ* e seus pontos nodais numerados a partir do número 1 e no sentido anti-horário.

Figura 9 - Elemento CSQ com coordenadas baricêntricas.



Fonte: Savassi (1996).

A partir da Figura 9 pode-se escrever as coordenadas baricêntricas em função das coordenadas cartesianas de acordo com as equações estabelecidas em (46):

$$\xi = \frac{(x - x_c)}{a} \qquad \eta = \frac{(y - y_c)}{b} \tag{46}$$

Onde, x_c = coordenada cartesiana em x do centro do elemento;

y_c = coordenada cartesiana em y do centro do elemento;

Utilizando as equações dadas em (46), de acordo com os referenciais, ficam determinadas as coordenadas baricêntricas. Uma vez mantida a origem do referencial de coordenadas baricêntricas no centro do retângulo e a numeração sequencial dos pontos nodais exatamente como o elemento da Figura 9, essas coordenadas irão se repetir individualmente para todos os elementos da malha. A Tabela 1 fornece os valores dessas coordenadas para cada um dos nós:

Tabela 1 - Valores das coordenadas baricêntricas.

Nó	ξ	η
1	-1	-1
2	1	-1
3	1	1
4	-1	1

Fonte: Autoria própria.

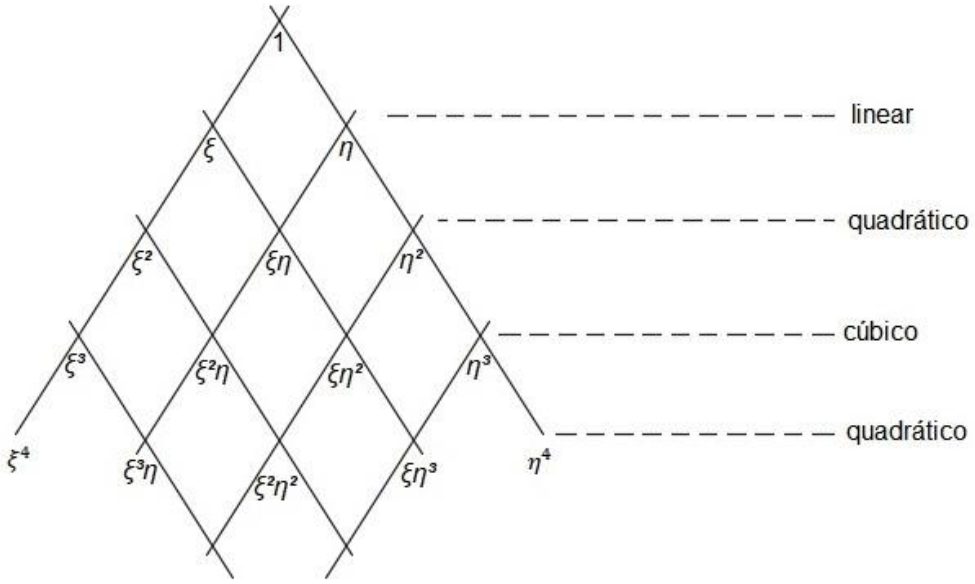
Como já foi dito anteriormente, as variáveis primárias (que no caso são os deslocamentos do elemento) podem ser determinadas fazendo a interpolação de parâmetros nodais (que no caso são os deslocamentos nodais) através de determinadas funções. Essas funções geralmente são obtidas por meio de um polinômio, de acordo com

o tipo de elemento utilizado.

Savassi (1996) diz que a escolha dos monômios que devem ser utilizados se inicia a partir da Figura 10, chamada de triângulo de monômios, também conhecida como triângulo de Pascal.

Soriano (2009) também destaca que os elementos bidimensionais devem possuir isotropia espacial, isto é, o seu comportamento numérico deve independer do seu referencial, que por sua vez depende da numeração dos seus pontos nodais. Para isso, é necessário que os monômios utilizados no desenvolvimento do elemento sejam obtidos através do triângulo de Pascal.

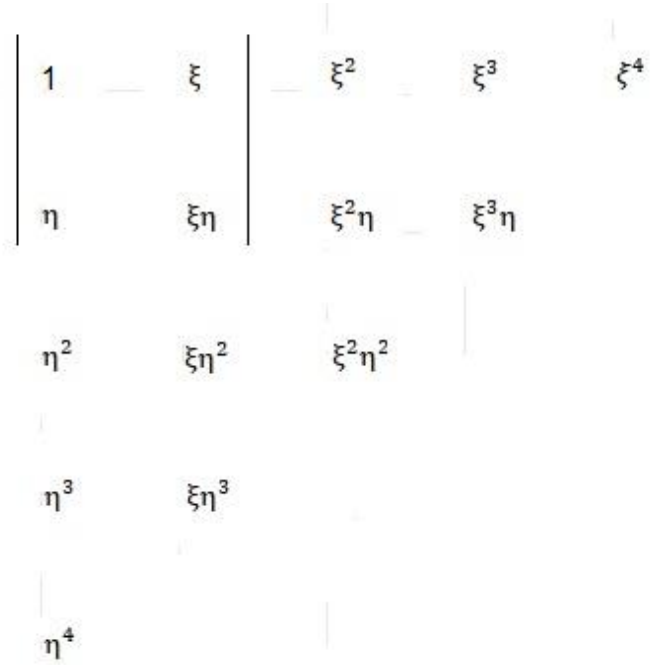
Figura 10 - Triângulo de Pascal.



Fonte: Savassi (1996).

Savassi (1996) ainda diz que no caso do elemento *CSQ*, existem pontos nodais apenas em seus vértices e os graus de liberdade existem em somente duas direções, o que implica em uma variação linear. Dessa forma, o triângulo de Pascal pode ser reescrito de acordo com a Figura 11:

Figura 11 - Triângulo de pascal para o caso de elemento CSQ.



Fonte: Savassi (1996).

Então, para o caso linear e considerando por enquanto somente o deslocamento horizontal *u*, a interpolação dos parâmetros nodais pode ser escrita de acordo com a equação (47):

$$u\ (\xi,\eta) = \alpha_1 + \xi \cdot \alpha_2 + \xi\eta \cdot \alpha_3 + \eta \cdot \alpha_4 \tag{47}$$

A equação (47) pode então ser escrita em forma matricial:

$$\{u\} = \begin{bmatrix} 1 & \xi & \xi\eta & \eta \end{bmatrix} \cdot \begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{Bmatrix} \tag{48}$$

$$\{u\} = [\omega] \cdot \{\alpha\} \tag{49}$$

A partir de então, utilizando a equação (47) e os valores das coordenadas baricêntricas fornecidos na Tabela 1, é possível montar o sistema linear mostrado na equação (50):

$$\begin{cases} u_1 = \alpha_1 - \alpha_2 + \alpha_3 - \alpha_4 \\ u_2 = \alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 \\ u_3 = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \\ u_4 = \alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 \end{cases} \tag{50}$$

Na forma matricial, tem-se o sistema estabelecido na equação (50):

$$\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \cdot \begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{Bmatrix} \tag{51}$$

$$\{u_i\} = [\omega_i] \cdot \{\alpha\} \tag{52}$$

A resolução do sistema linear da equação (52) é feita pelo método da inversão de matriz, de forma que:

$$\{\alpha\} = [\omega_i]^{-1} \cdot \{u_i\} \tag{53}$$

Assim, a equação (53) pode ser substituída na equação (49), resultando em:

$$\{u\} = [\omega] \cdot [\omega_i]^{-1} \cdot \{u_i\} \tag{54}$$

Ao se comparar a equação (54) com a equação (31) é possível estabelecer que:

$$[\phi] = [\omega] \cdot [\omega_i]^{-1} \tag{55}$$

Tem-se que a matriz $[\omega_i]^{-1}$ é a matriz inversa da matriz $[\omega_i]$ definida na equação (51), e de acordo com a definição de matriz inversa é possível escrevê-la:

$$[\omega_i]^{-1} = \left(\frac{1}{4}\right) \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 \end{bmatrix} \tag{56}$$

Dessa forma, utilizando as equações (48), (55) e (56), tem-se que:

$$[\phi] = [1 \quad \xi \quad \xi\eta \quad \eta] \cdot \left(\frac{1}{4}\right) \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 \end{bmatrix} \tag{57}$$

A partir de então, resolvendo a equação (57), é possível determinar a matriz $[\phi]$, que contém as funções de interpolação, de acordo com a equação (58):

$$[\phi] = \left[\frac{(1-\xi)\cdot(1-\eta)}{4} \quad \frac{(1+\xi)\cdot(1-\eta)}{4} \quad \frac{(1+\xi)\cdot(1+\eta)}{4} \quad \frac{(1-\xi)\cdot(1+\eta)}{4} \right] \tag{58}$$

$$[\phi] = [\phi_1 \quad \phi_2 \quad \phi_3 \quad \phi_4] \tag{59}$$

As deduções nos passos anteriores foram todas realizadas para os deslocamentos horizontais *u*, porém as mesmas deduções são válidas para os deslocamentos verticais *v*. Por fim, isso resulta no sistema matricial mostrado pela equação (60):

$$\begin{cases} \{u\} = [\phi] \cdot \{u_i\} \\ \{v\} = [\phi] \cdot \{v_i\} \end{cases} \tag{60}$$

A partir das formulações mostradas é possível reescrever a equação (33):

$$[B] = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \cdot \begin{bmatrix} \phi_1 & 0 & \phi_2 & 0 & \phi_3 & 0 & \phi_4 & 0 \\ 0 & \phi_1 & 0 & \phi_2 & 0 & \phi_3 & 0 & \phi_4 \end{bmatrix} \tag{61}$$

$$[B] = \begin{bmatrix} \frac{\partial \phi_1}{\partial x} & 0 & \frac{\partial \phi_2}{\partial x} & 0 & \frac{\partial \phi_3}{\partial x} & 0 & \frac{\partial \phi_4}{\partial x} & 0 \\ 0 & \frac{\partial \phi_1}{\partial y} & 0 & \frac{\partial \phi_2}{\partial y} & 0 & \frac{\partial \phi_3}{\partial y} & 0 & \frac{\partial \phi_4}{\partial y} \\ \frac{\partial \phi_1}{\partial y} & \frac{\partial \phi_1}{\partial x} & \frac{\partial \phi_2}{\partial y} & \frac{\partial \phi_2}{\partial x} & \frac{\partial \phi_3}{\partial y} & \frac{\partial \phi_3}{\partial x} & \frac{\partial \phi_4}{\partial y} & \frac{\partial \phi_4}{\partial x} \end{bmatrix} \tag{62}$$

$$\left[B\right]^{\mathrm{T}} = \begin{bmatrix} \frac{\partial \phi_1}{\partial x} & 0 & \frac{\partial \phi_1}{\partial y} \\ 0 & \frac{\partial \phi_1}{\partial y} & \frac{\partial \phi_1}{\partial x} \\ \frac{\partial \phi_2}{\partial x} & 0 & \frac{\partial \phi_2}{\partial y} \\ 0 & \frac{\partial \phi_2}{\partial y} & \frac{\partial \phi_2}{\partial x} \\ \frac{\partial \phi_3}{\partial x} & 0 & \frac{\partial \phi_3}{\partial y} \\ 0 & \frac{\partial \phi_3}{\partial y} & \frac{\partial \phi_3}{\partial x} \\ \frac{\partial \phi_4}{\partial x} & 0 & \frac{\partial \phi_4}{\partial y} \\ 0 & \frac{\partial \phi_4}{\partial y} & \frac{\partial \phi_4}{\partial x} \end{bmatrix} \tag{63}$$

Por fim, sendo dA o elemento infinitesimal de área do elemento e derivando as equações em (46), pode-se escrever:

$$dA = dx \cdot dy \tag{64}$$

$$\frac{d\xi}{dx} = \frac{1}{a} \qquad \frac{d\eta}{dy} = \frac{1}{b} \tag{65}$$

Dessa forma, através da derivação utilizando a Regra da Cadeia, tem-se:

$$\frac{\partial \phi_i}{\partial x} = \frac{\partial \phi_i}{\partial \xi} \cdot \frac{\partial \xi}{\partial x} + \frac{\partial \phi_i}{\partial \eta} \cdot \frac{\partial \eta}{\partial x} = \frac{1}{a} \cdot \frac{\partial \phi_i}{\partial \xi} \tag{66}$$

$$\frac{\partial \phi_i}{\partial y} = \frac{\partial \phi_i}{\partial \xi} \cdot \frac{\partial \xi}{\partial y} + \frac{\partial \phi_i}{\partial \eta} \cdot \frac{\partial \eta}{\partial y} = \frac{1}{b} \cdot \frac{\partial \phi_i}{\partial \eta} \tag{67}$$

Assim, substituindo as relações obtidas em (65) na equação (43), tem-se que:

$$[k]^{e} = t \cdot a \cdot b \cdot \int_{-1}^{1}\int_{-1}^{1} [B]^{T} \cdot [E] \cdot [B] \cdot d\xi \cdot d\eta \tag{68}$$

A equação (68) define a matriz de rigidez para o elemento retangular com 4 pontos nodais e 2 graus de liberdade por nó, conhecido como elemento *CSQ*. Resolvendo cada um dos termos dessa equação obtém-se como resultado a matriz simétrica e quadrada de ordem oito mostrada na equação (69):

$$[k]^{e} = \frac{t \cdot E}{(1-\nu^{2})} \cdot \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} & k_{17} & k_{18} \\ \vdots & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} & k_{27} & k_{28} \\ \vdots & & k_{33} & k_{34} & k_{35} & k_{36} & k_{37} & k_{38} \\ \vdots & & & k_{44} & k_{45} & k_{46} & k_{47} & k_{48} \\ \vdots & & & \ddots & k_{55} & k_{56} & k_{57} & k_{58} \\ \vdots & & \ddots & & & k_{66} & k_{67} & k_{68} \\ \vdots & \ddots & & & & & k_{77} & k_{78} \\ sim. & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & k_{88} \end{bmatrix} \tag{69}$$

Sendo os termos da matriz de rigidez apresentados individualmente em (70):

$$\begin{bmatrix}
k_{11}=\frac{2b^2+(1-\nu)a^2}{6ab}; & k_{12}=\frac{(1+\nu)}{8}; & k_{13}=\frac{-4b^2+(1-\nu)a^2}{12ab}; \\
k_{14}=\frac{(-1+3\nu)}{8}; & k_{15}=\frac{-2b^2+(-1+\nu)a^2}{12ab}; & k_{16}=\frac{(-1-\nu)}{8}; \\
k_{17}=\frac{b^2+(-1+\nu)a^2}{6ab}; & k_{18}=\frac{(1-3\nu)}{8}; & k_{22}=\frac{2a^2+(1-\nu)b^2}{6ab}; \\
k_{23}=\frac{(1-3\nu)}{8}; & k_{24}=\frac{a^2+(-1+\nu)b^2}{6ab}; & k_{25}=\frac{(-1-\nu)}{8}; \\
k_{26}=\frac{-2a^2+(-1+\nu)b^2}{12ab}; & k_{27}=\frac{(-1+3\nu)}{8}; & k_{28}=\frac{-4a^2+(1-\nu)b^2}{12ab}; \\
k_{33}=\frac{2b^2+(1-\nu)a^2}{6ab}; & k_{34}=\frac{(-1-\nu)}{8}; & k_{35}=\frac{b^2+(-1+\nu)a^2}{6ab}; \\
k_{36}=\frac{(-1+3\nu)}{8}; & k_{37}=\frac{-2b^2+(-1+\nu)a^2}{12ab}; & k_{38}=\frac{(1+\nu)}{8}; \\
k_{44}=\frac{2a^2+(1-\nu)b^2}{6ab}; & k_{45}=\frac{(1-3\nu)}{8}; & k_{46}=\frac{-4a^2+(1-\nu)b^2}{12ab}; \\
k_{47}=\frac{(1+\nu)}{8ab}; & k_{48}=\frac{-2a^2+(-1+\nu)b^2}{12ab}; & k_{55}=\frac{2b^2+(1-\nu)a^2}{6ab}; \\
k_{56}=\frac{(1+\nu)}{8}; & k_{57}=\frac{-4b^2+(1-\nu)a^2}{12ab}; & k_{58}=\frac{(-1+3\nu)}{8}; \\
k_{66}=\frac{2a^2+(1-\nu)b^2}{6ab}; & k_{67}=\frac{(1-3\nu)}{8}; & k_{68}=\frac{a^2+(-1+\nu)b^2}{6ab}; \\
k_{77}=\frac{2b^2+(1-\nu)a^2}{6ab}; & k_{78}=\frac{(-1-\nu)}{8}; & k_{88}=\frac{2a^2+(1-\nu)b^2}{6ab};
\end{bmatrix} \quad (70)$$

E, assim como já foi definido para um elemento qualquer, para o elemento *CSQ* podemos reduzir a equação inicial do PTV utilizando a formulação do MEF, obtendo a equação que relaciona as forças nodais com os deslocamentos do elemento:

$$\{F\}^e = [k]^e \cdot \{d\}^e \quad (71)$$

Vale lembrar que a equação (71) é específica para um único elemento *CSQ*. Portanto, em uma malha com inúmeros elementos se faz necessário ordenar cada uma das matrizes de cada um dos elementos para determinar a matriz de rigidez global da estrutura.

A facilidade de programar as equações obtidas até então e os resultados satisfatórios que elas podem apresentar, permite concluir que o Método dos Elementos Finitos, embasado na formulação mostrada neste trabalho para os elementos *CSQ*, realmente é um método importante e muito útil durante a análise estrutural de chapas.

2.7.4 A análise de chapas aplicada ao Dimensionamento de Vigas-Parede de Concreto Armado

2.7.4.1 Definição e Metodologia de Cálculo de Viga-Parede

Borda (2013) afirma que vigas-parede são elementos estruturais planos verticais, caracterizados por apresentarem carregamentos atuantes em seu próprio plano, e para os quais as deformações apresentadas não são lineares. Destaca ainda que na prática da engenharia as vigas-parede são utilizadas, por exemplo, na construção de silos, reservatórios e fachadas de edifícios. Santos (1999) também destaca a utilização de vigas-parede como blocos de coroamento de estacas, em tetos de transição (suportando carga de pilares) ou ainda como elementos estruturais de contenção em subsolos.

As vigas-parede são consideradas elementos estruturais especiais, pois para elas não se aplica a teoria da flexão de Navier-Bernoulli, que diz que as seções transversais planas permanecem planas após a aplicação do carregamento. Isto significa que para esses elementos as distribuições de deformações e tensões horizontais não são lineares. (BORDA, 2013)

Franco (2015) também destaca que existem elementos estruturais para os quais não se aplica a hipótese de Bernoulli, isto porque estes elementos possuem regiões descontínuas, ou seja, que não apresentam distribuição linear de tensões e deformações. Essa descontinuidade pode ocorrer em regiões próximas aos pontos de aplicação de forças, próximas aos apoios, regiões com mudança brusca na geometria da peça, entre outros casos. Para as vigas-parede, por exemplo, essa descontinuidade ocorre ao longo de toda a sua extensão.

Santos (1999) também afirma que a teoria elementar de flexão para vigas esbeltas, utilizada na resistência dos materiais, que diz que as seções planas permanecem planas após a deformação, não se aplica ao caso das vigas-parede. Pois mesmo considerando um

material homogêneo e elástico, a distribuição das tensões normais não é linear e a das cisalhantes não é parabólica. Assim sendo, na determinação dos esforços internos deve-se levar em conta as condições de equilíbrio, as condições de contorno e outras relações constitutivas mais complexas.

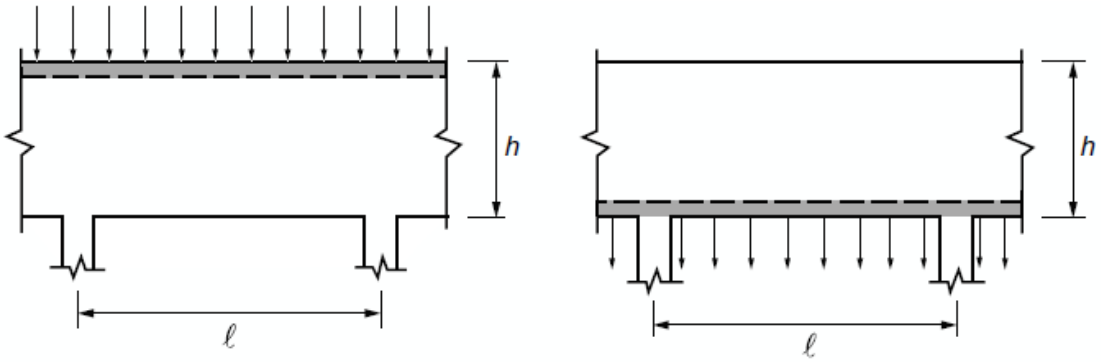
Segundo a ABNT NBR 6118:2014, vigas-parede são tipos de chapas de concreto caracterizadas por apresentarem uma das seguintes relações entre comprimento de vão livre (l) e altura da seção transversal (h):

$\frac{l}{h} \leq 2$, para o caso de vigas biapoiadas;

$\frac{l}{h} \leq 3$, para o caso de vigas contínuas;

Podendo ainda apresentar carregamento aplicado em sua superfície superior ou inferior, de acordo com a Figura 12.

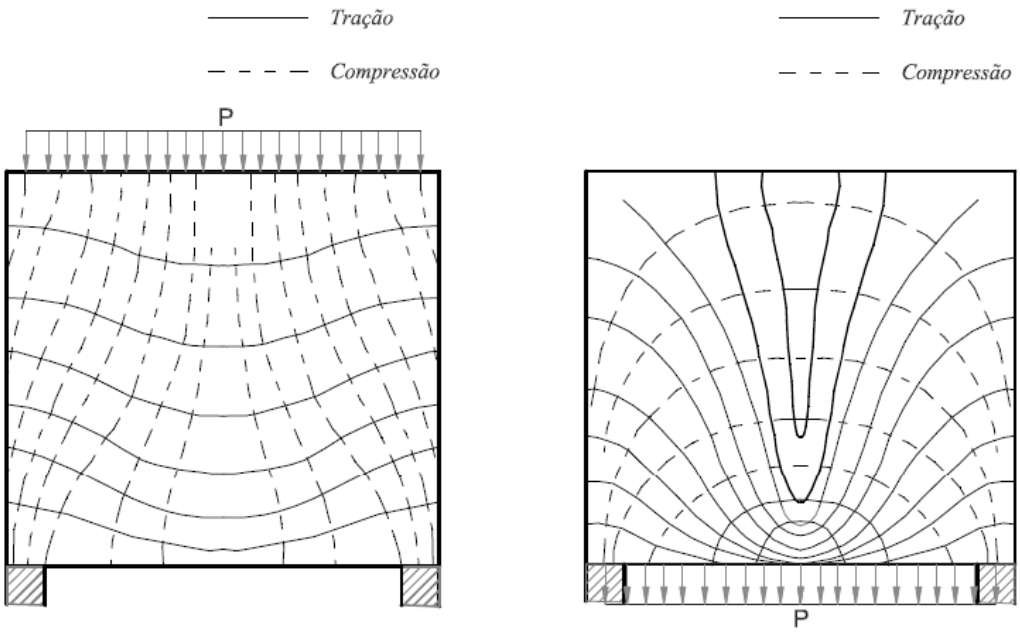
Figura 12 – Tipos comuns de carregamentos em vigas-parede.



Fonte: ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (2014).

Borda (2013) diz que a forma com que o carregamento é aplicado define o comportamento da estrutura, pois isso representa diferentes trajetórias e distribuições de tensões, definindo assim a configuração final das fissuras. Para ilustrar, o mesmo autor apresenta a Figura 13, que mostra a trajetória das tensões principais em dois casos distintos de carregamentos em uma viga-parede com relação de esbeltez igual a um (l/h = 1).

Figura 13 – Trajetórias das tensões principais de acordo com diferentes formas de carregamento.



Fonte: Leonhardt & Mönnig (1978), *apud* Borda (2013).

A ABNT NBR 6118:2014 destaca ainda que as vigas-parede possuem uma ineficiência, em relação a flexão e ao cisalhamento, quando comparadas com vigas usuais. Além disso, por serem altas costumam apresentar problemas de estabilidade de corpo rígido, sendo necessário na maioria das vezes, o uso de enrijecedores de apoio ou travamentos em alguns pontos.

A norma também afirma que para o dimensionamento de vigas-parede pode-se utilizar modelos de cálculo do tipo plano elástico-linear ou não-linear, baseado em métodos numéricos, tais como o método dos elementos finitos, abordado neste trabalho. Além disso, afirma que para o dimensionamento no estado-limite último, pode-se utilizar modelos concebidos a partir do método das bielas e tirantes. (ABNT NBR 6118:2014)

Franco (2015) diz que para o dimensionamento de tais vigas existem dois métodos que contam com grande prestígio entre os pesquisadores. São eles, o Método das Bielas e Tirantes (baseado na análise plástica) e o Método dos Elementos Finitos. Segundo ele, ambos os métodos conseguem realizar uma análise satisfatória do comportamento resistente destas estruturas.

Santos (1999) em seu trabalho faz uma comparação entre as diferentes recomendações de dimensionamento de vigas-paredes existentes atualmente em diversas normas ao redor do mundo, afirmando que nenhuma dessas recomendações satisfaz completamente o projeto de tais elementos. Destaca ainda que a norma brasileira não traz

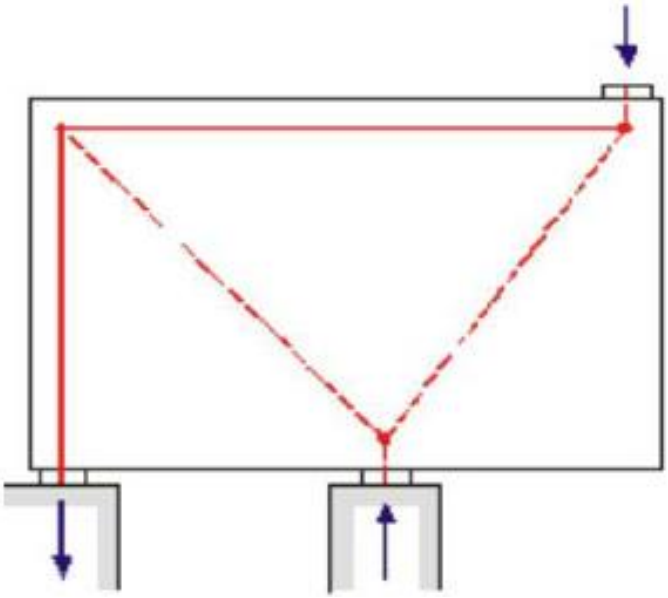
nenhuma indicação mais aprofundada para o dimensionamento de tais vigas, mas apenas diz que este deve ser feito como se a viga fosse uma chapa no regime elástico.

Santos (1999) afirma que as primeiras pesquisas sobre o comportamento de vigas-parede foram baseadas no comportamento elástico, o que impõe a condição de que o material seja isotrópico e obedeça a Lei de Hooke, trazendo limitações para estudos relativos à projetos práticos. Além disso, também são citados inúmeros trabalhos de pesquisa relativos ao comportamento das vigas, com destaque para o trabalho de Robins e Kong (1973, apud SANTOS, 1999), que utilizaram o método dos elementos finitos para prever a carga última e os modelos de fissuração em tais vigas.

Segundo Santos (1999) as dificuldades em se ter um método eficiente para o dimensionamento de vigas-parede se dão pela grande quantidade de variáveis que atuam influenciando no comportamento dessas estruturas. Entre essas variáveis se destacam a geometria da viga (espessura, relação l/h), os tipos de apoio, a quantidade de enrijecedores, o material utilizado, além do tipo de carregamento e do seu ponto de aplicação. Além disso, afirma que se utilizam três diferentes processos de análise da capacidade de carga para estas estruturas, sendo eles: métodos de projetos empíricos ou semi-empíricos; análise bi ou tridimensional (linear ou não-linear); utilização do modelo de treliça composta de bielas de concreto e tirantes de aço. Este último por sua vez, é tido pelos pesquisadores como o método mais promissor para o cálculo da resistência dessas vigas.

Borda (2013) diz que o método de bielas e tirantes teve início com os trabalhos de *Wilhelm Ritter* em 1899, que determinou a armadura transversal necessária para uma viga associando o mecanismo resistente de uma viga fissurada (estágio II) com o comportamento de uma treliça idealizada. Em 1902, *Mörsch* refinou este modelo, assumindo que as forças diagonais do modelo de *Ritter* eram melhor representadas por campos de tensão de compressão. Posteriormente, à medida que novos trabalhos foram sendo realizados, se criou uma base teórica de uso racional sobre o método, fazendo com que este fosse adotado em diversas normas pelo mundo, inclusive a norma brasileira, já citada neste trabalho. A Figura 14 mostra uma viga-parede idealizada pelo método de bielas e tirantes.

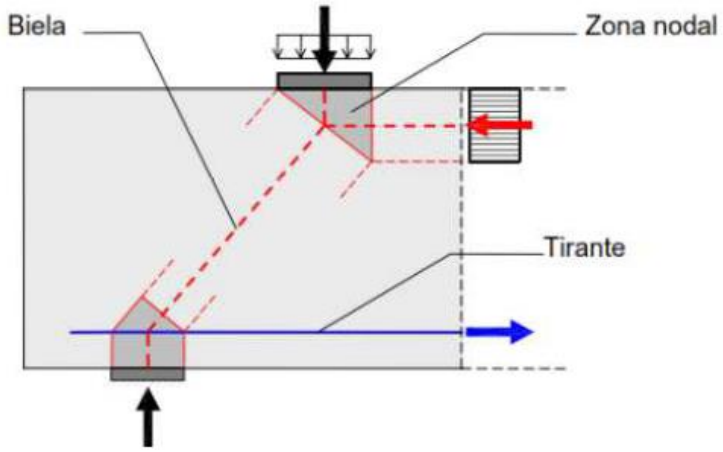
Figura 14 – Viga-parede idealizada pelo modelo de bielas e tirantes.



Fonte: Borda (2013).

Borda (2013) destaca ainda que o modelo de bielas e tirantes consiste em idealizar uma estrutura real como se fosse uma treliça formada por elementos denominados bielas, que representam o campo de tensões de compressão, e por elementos denominados tirantes, que representam o campo de tensões de tração. Além das zonas nodais, formadas pelos pontos de ligação entre as bielas e os tirantes. A Figura 15 ilustra esse modelo.

Figura 15 – Modelo de bielas e tirantes.



Fonte: Miguel *et al* (2009), *apud* Borda (2013).

Franco (2015) realizou um trabalho no qual se faz uma comparação entre diferentes metodologias de cálculo de vigas-parede. Assim, para duas vigas-paredes, ela comparou resultados experimentais existentes na literatura e os resultados obtidos pelo Método dos Elementos Finitos no trabalho desenvolvido por d'Avila (2003), com os resultados obtidos pelo Método das Bielas e Tirantes (análise plástica), além de um terceiro método, baseado em uma análise não-linear e denominado Método Corda-Painel. As duas vigas utilizadas no trabalho possuíam mesma geometria, porém taxas de armadura e carregamentos distintos.

Em seu trabalho, Franco (2015) se baseou em uma análise elástica realizada por elementos finitos para escolher o modelo utilizado no dimensionamento via método das bielas e tirantes. Desta análise de tensões elásticas se obteve as direções das tensões principais de compressão e de tração, definindo assim as direções das bielas e dos tirantes, respectivamente. (FRANCO, 2015)

2.7.4.2 Alternativas de Dimensionamento

As estruturas civis são dimensionadas seguindo métodos estabelecidos nas normas e que sejam seguros mediante as solicitações que a estrutura virá a receber. Como já dito anteriormente, no caso das vigas-parede, a norma brasileira não deixa claro qual método deverá ser utilizado no dimensionamento, somente diz que este pode ser feito utilizando métodos numéricos, como o Método dos Elementos Finitos ou ainda métodos utilizando modelos de bielas e tirantes.

Uma alternativa de dimensionamento seria utilizar o MEF para realizar a análise elástica linear da estrutura, determinando seus pontos críticos de maiores tensões e através desses pontos realizar o dimensionamento a partir de um modelo de bielas e tirantes que represente de maneira eficaz a estrutura.

Como já dito anteriormente, as vigas-parede são caracterizadas por apresentarem distribuição não linear de deformações e tensões. Por esse motivo, soluções elásticas são válidas para analisar o comportamento destas estruturas somente até o surgimento das primeiras fissuras (LEONHARDT e WALTHER, 1996, *apud* SANTOS, 1999), após isso, ocorre uma redistribuição das tensões e a capacidade da viga deve ser prevista através de uma solução inelástica, tal como a solução via modelo de bielas e tirantes. (ASHOUR e

MORLEY, 1996, *apud* SANTOS, 1999)

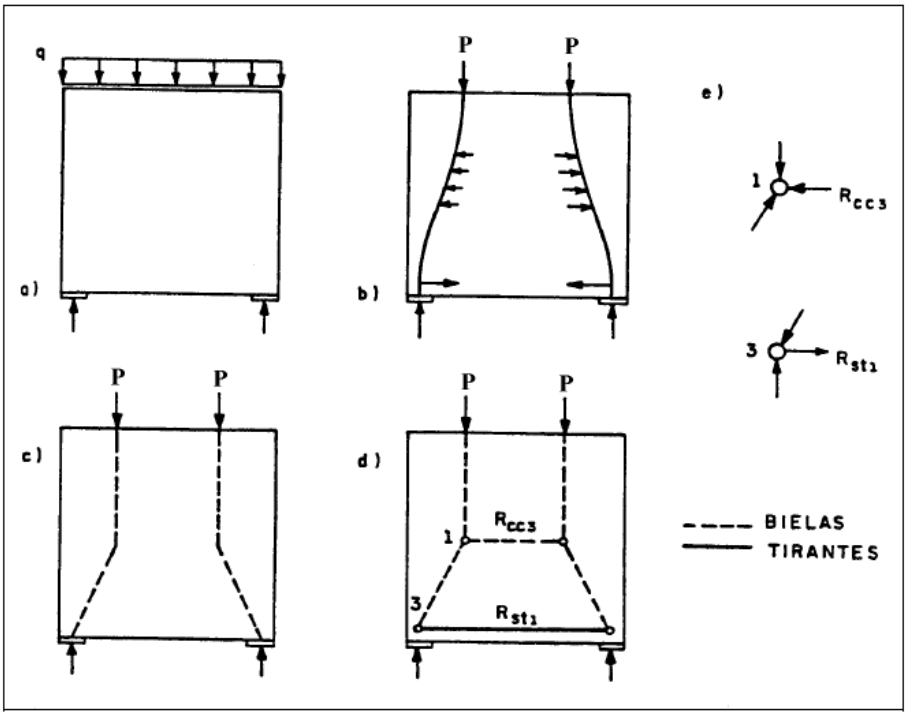
Santos (1999) destaca que se conhecendo um modelo adequado, as forças nas bielas e tirantes serão calculadas pelo equilíbrio entre forças externas e internas. Dessa forma, será possível realizar o dimensionamento dos tirantes e verificar as bielas de concreto. Para as vigas-parede, o seu comportamento é influenciado fortemente pelo ponto de aplicação das cargas e pelas condições de vinculação. Assim sendo, a escolha do modelo ideal deve levar em conta esses fatores, bem como o ângulo entre as bielas e tirantes e as características da armadura utilizada.

Segundo Franco (2015) os esforços solicitantes para uma viga-parede são as ações aplicadas e as reações de apoios obtidas a partir da análise estrutural do problema. A definição desses esforços é de fundamental importância para a definição dos caminhos de tensões dentro do elemento estrutural. Além disso, a definição do modelo utilizado deve levar em conta as características geométricas do problema.

Citando o exemplo de uma viga-parede biapoiada com carregamento distribuído, a escolha do modelo se inicia com o processo de fluxo de carga dentro da estrutura. O equilíbrio externo deve ser satisfeito, bem como o equilíbrio interno (equilíbrio de nós). A ação do carregamento distribuído deve ser substituída por forças concentradas equivalentes. As cargas devem ser interligadas pelos caminhos mais curtos possíveis, formando-se o polígono do modelo e realizando-se o equilíbrio dos nós. Após isso, se inicia o dimensionamento das bielas e tirantes, bem como a verificação das transferências de forças nas regiões nodais. (SANTOS, 1999)

A Figura 16 exemplifica a escolha de um modelo de bielas e tirantes para uma viga-parede biapoiada submetida à carregamento distribuído. Na Figura 16 (a) se tem a estrutura e os seus apoios, em 16 (b) se tem os caminhos definidos para as cargas e reações, em 16 (c) se tem as linhas do polígono definido para o modelo, em 16 (d) se tem a configuração geométrica final do modelo, escolhida após a definição do ângulo entre as bielas e tirantes, e em 16 (e) se tem os equilíbrios de nós necessários para a resolução do modelo e dimensionamento da estrutura.

Figura 16 – Modelo de Bielas e Tirantes adotado para viga-parede biapoiada com carregamento distribuído.



Fonte: Santos (1999, p.42).

Segundo Campos (1996), a utilização de um programa computacional baseado no Método dos Elementos Finitos (análise elástica linear) pode facilitar o traçado do caminho das cargas e auxiliar na montagem do modelo de bielas e tirantes à ser utilizado.

3 ASPECTOS COMPUTACIONAIS

O objetivo principal deste trabalho foi a elaboração de um código computacional em linguagem de programação Python capaz de realizar a análise elástica de linear de chapas via MEF, de forma a obter como resultados deslocamentos nodais, tensões e deformações atuantes, bem como as reações de apoio.

De forma geral o código está dividido em três "módulos" principais, sendo eles: entrada de dados do problema, processamento dos cálculos e saída de dados. Na entrada de dados o usuário informa os principais dados do problema à ser estudado, tais como as características físicas e geométricas da chapa, as condições de vinculação e de carregamento, bem como a quantidade de elementos finitos que se deseja utilizar para a composição da malha.

Na etapa de processamento dos cálculos o código realiza todo o procedimento de resolução do problema utilizando o Método dos Elementos Finitos. Essa etapa é executada de forma que não apareça em tela para o usuário, evitando assim a poluição visual com dados que não são de interesse nesse momento. Dentro dessa etapa estão procedimentos como a divisão da estrutura na quantidade de elementos definida pelo usuário, a montagem da matriz de rigidez elementar e da matriz de rigidez da estrutura, a montagem do vetor de forças externas, a resolução do sistema linear de equações para a determinação dos deslocamentos nodais, a determinação das reações de apoio e a determinação das tensões e deformações ao longo dos nós da chapa.

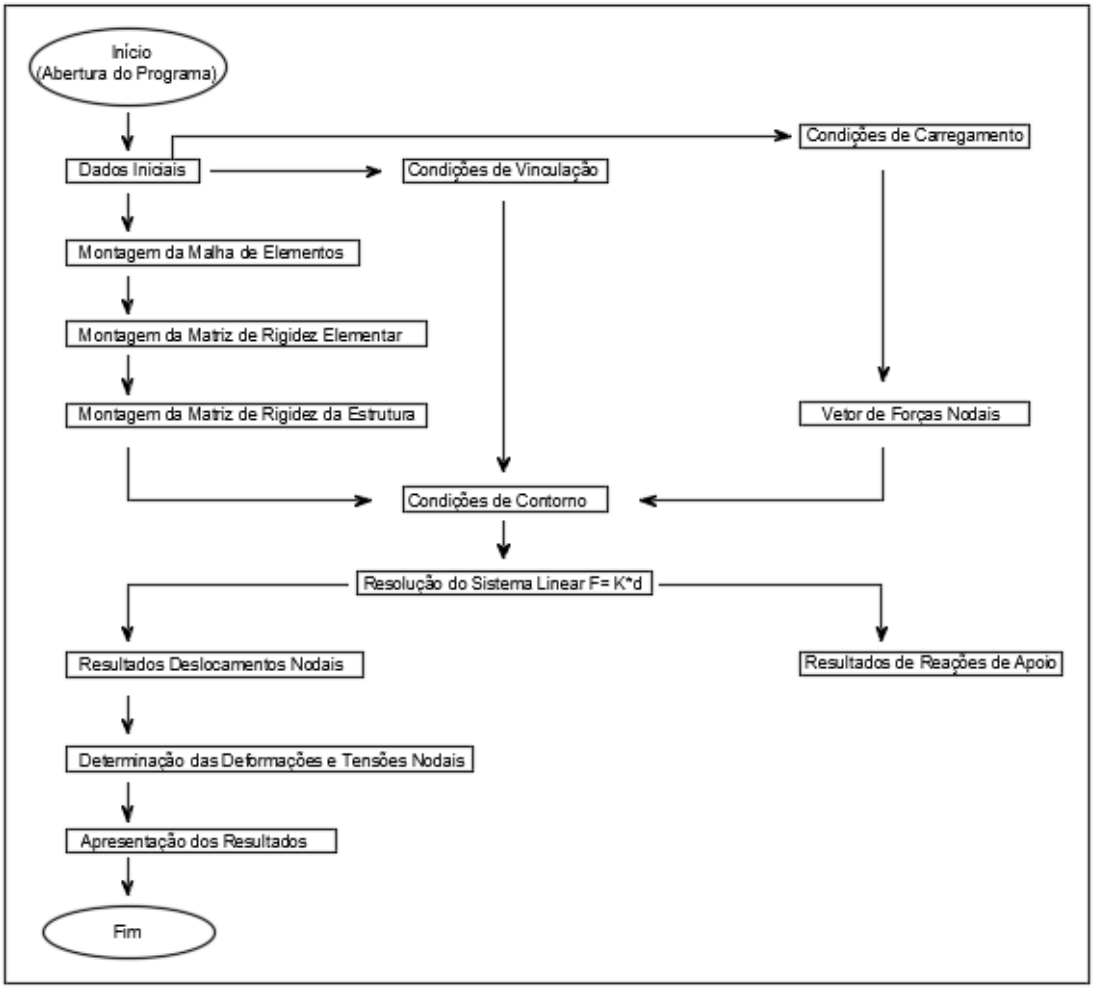
O último módulo é a de saída de dados, na qual o código apresenta em arquivo de texto os resultados encontrados, para que assim o usuário possa utilizar aquilo que mais lhe interessa. Essa apresentação de resultados ocorre por pontos nodais devido as características do elemento *CSQ*. Dessa forma ao final da análise estrutural o usuário possui informações de forças, deslocamentos, tensões e deformações em cada um dos nós dos elementos finitos que compõem a malha.

3.1 Esquema Geral de Cálculo

Como dito anteriormente, o código computacional foi desenvolvido de forma que ficasse dividido em módulos, sendo esses módulos divididos em sub-rotinas que estão

esquematizadas através do fluxograma da Figura 17. Cada uma dessas sub-rotinas será explicada e detalhada no próximo item deste capítulo.

Figura 17 – Fluxograma de etapas do programa desenvolvido.



Fonte: Autoria própria.

3.2 Sub-Rotinas

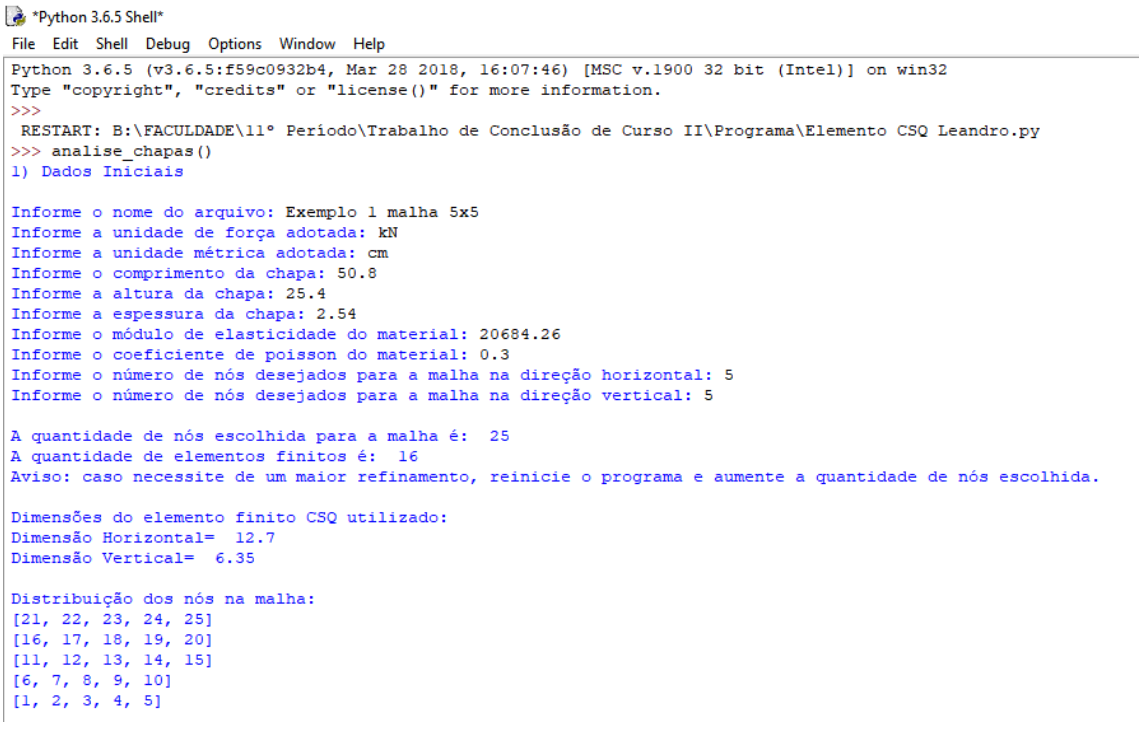
3.2.1 Dados Iniciais

Nesta sub-rotina é solicitado ao usuário que informe os dados necessários para os procedimentos de cálculo utilizados no método, esses dados tratam de propriedades físicas do material e geométricas da chapa.

Inicialmente são solicitadas as informações da chapa como comprimento, altura, espessura, módulo de elasticidade e coeficiente de Poisson do material. As unidades de medida utilizadas ficam a critério do usuário, porém uma vez adotada uma unidade ela deve ser mantida até o final da execução.

Por fim são solicitadas informações referentes ao refinamento de malha desejado para o problema, isto é, a quantidade de nós que o usuário deseja utilizar na análise, tanto na direção horizontal quanto na direção vertical. A partir do momento que a quantidade de nós é escolhida o programa calcula automaticamente a quantidade de elementos finitos que será utilizada, bem como as dimensões deste elemento, além de distribuir esses nós ao longo da estrutura formando a malha e informando na tela todas essas informações para que o usuário possa realizar a conferência e avaliar o refinamento da malha. A Figura 18 apresenta uma cópia de tela que mostra exatamente essa etapa do funcionamento do programa.

Figura 18 – Cópia de tela sub-rotina "Dados Iniciais".



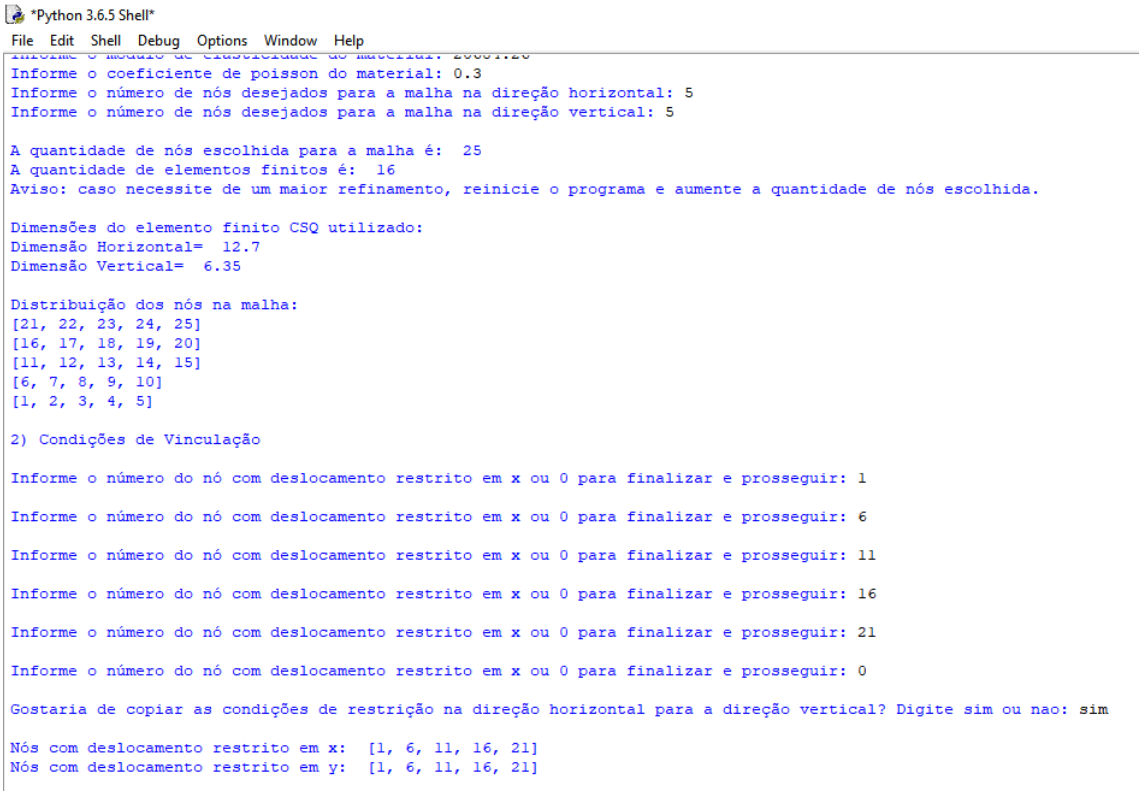
Fonte: Autoria própria.

3.2.2 Definição das Condições de Vinculação

Nesta sub-rotina é solicitado ao usuário que informe quais são as condições de vinculação do problema. Para isso é solicitado que o usuário informe o número dos nós com deslocamento restrito, tanto na direção horizontal (eixo x) quanto na direção vertical (eixo y).

Por facilidade, o número do nó pode ser visualizado na distribuição dos nós na malha de elementos que é mostrada em tela na etapa anterior. Além disso, o programa oferece a opção de copiar as mesmas informações de restrição adotadas na direção horizontal para a direção vertical. A Figura 19 apresenta uma cópia de tela que mostra essa etapa do funcionamento do programa.

Figura 19 – Cópia de tela sub-rotina "Definição das Condições de Vinculação".



Fonte: Autoria própria.

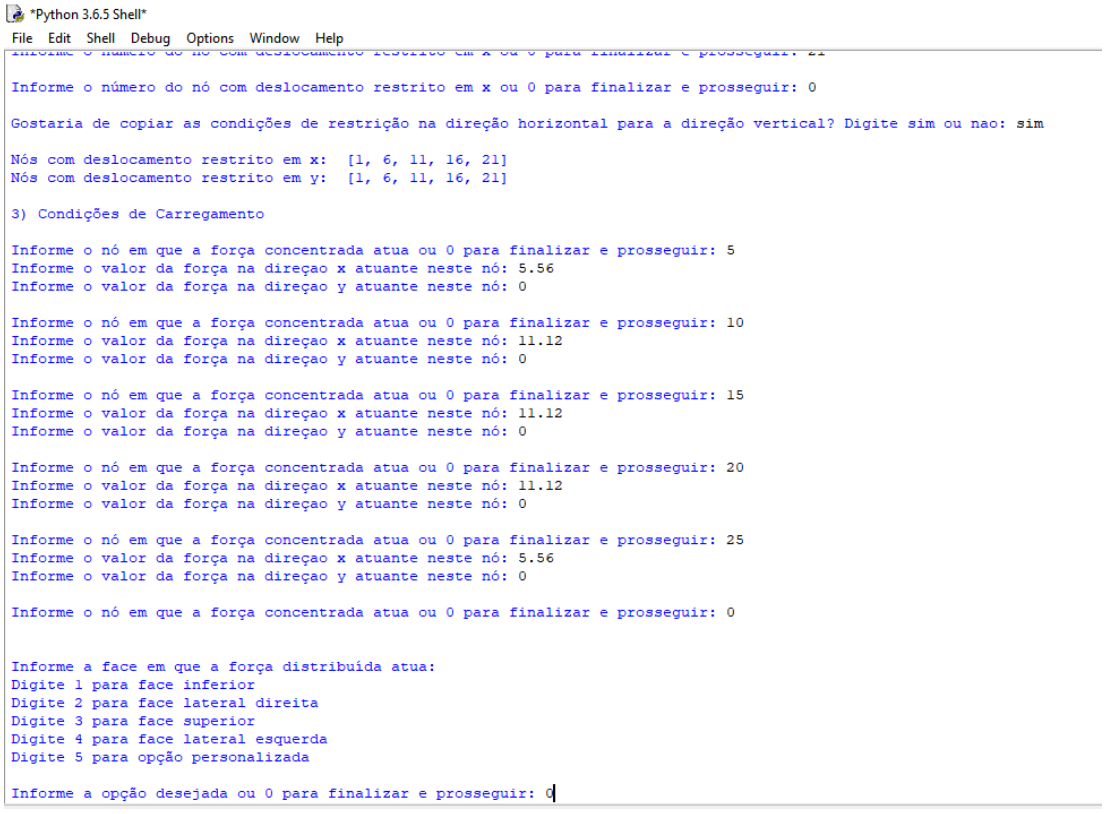
3.2.3 Definição das Condições de Carregamento

Nesta sub-rotina é solicitado ao usuário que informe quais são as condições de carregamentos aos quais a chapa está submetida. Para isso é solicitado que o usuário informe o número dos nós que possuam algum tipo de força aplicada, bem como os valores dessa força tanto na direção horizontal (eixo x) quanto na direção vertical (eixo y).

O programa oferece a opção de adicionar forças concentradas, informando para isso o número do nó que recebe estas forças, ou então a opção de adicionar força distribuída, informando para isso a face da chapa ou o intervalo de nós que recebe este carregamento.

Durante essa etapa as unidades de medida utilizadas para as forças devem ser compatíveis com as unidades definidas nos dados iniciais do problema. Além disso, é importante salientar que por convenção os valores positivos indicam forças para a direita na direção horizontal ou para cima na direção vertical, enquanto que valores negativos indicam sentidos contrários. A Figura 20 apresenta uma cópia de tela que mostra essa etapa do funcionamento do programa.

Figura 20 – Cópia de tela sub-rotina "Definição das Condições de Carregamento".



Fonte: Autoria própria.

3.2.4 Montagem da Malha de Elementos Finitos

Esta sub-rotina é caracterizada por ser a primeira do código pertencente exclusivamente à etapa de processamento do cálculo, ou seja, nela não é necessária a participação do usuário e o próprio programa é responsável pela sua execução.

A elaboração da malha se inicia na etapa de "Dados Iniciais" quando o usuário informa a quantidade de nós desejados para a análise do problema. A partir disso o próprio programa calcula a quantidade de elementos que será utilizada.

Então, nesta sub-rotina ocorre a alocação desses elementos na estrutura e a definição de quais são os nós pertencentes a cada um dos elementos finitos de acordo com a numeração sequencial estabelecida. Lembrando que cada um dos elementos é composto por quatro nós.

3.2.5 Montagem da Matriz de Rigidez Elementar

Nesta sub-rotina o programa interpreta os dados informados pelo usuário e os cálculos realizados nas sub-rotinas anteriores e efetua a montagem da matriz de rigidez do elemento *CSQ*, dada pela equação (70). Lembrando que essa matriz possui ordem 8x8, pois o elemento *CSQ* possui 4 pontos nodais e 2 graus de liberdade em cada um deles.

3.2.6 Vetor de Deslocamentos Nodais Para Cada Elemento

Nesta sub-rotina o programa armazena dados referentes aos deslocamentos dos nós de cada um dos elementos finitos na malha. Assim, para cada elemento ele armazena em uma lista os nós que possuem deslocamento restrito, os nós que possuem deslocamento livre e até mesmo os nós que não pertencem àquele elemento finito. Esta lista é importante na etapa de montagem de matriz de rigidez da estrutura, pois indica as posições corretas para armazenar os termos da matriz daquele elemento na matriz global.

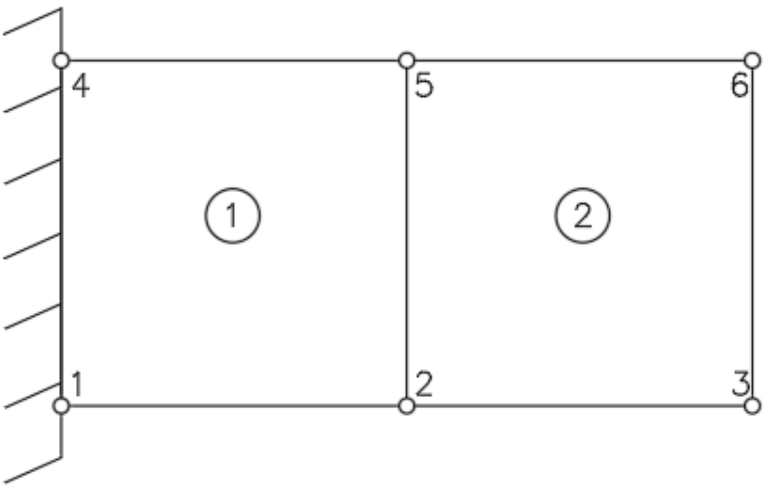
3.2.7 Montagem da Matriz de Rigidez Global

Nesta sub-rotina o programa é responsável por agrupar a matriz de rigidez elementar de cada um dos elementos finitos presentes na malha dentro da matriz de rigidez global da estrutura. Cada matriz elementar possui sua posição dentro da matriz global pré-estabelecida de acordo com a numeração dos nós pertencentes ao elemento que ela representa.

O procedimento utilizado pelo programa é o seguinte: ele cria para cada um dos elementos uma matriz nula de mesma ordem que a matriz de rigidez global, ordem essa que por sua vez é determinada pelo produto entre a quantidade de nós estabelecidos na malha e a quantidade de graus de liberdade por nó do elemento *CSQ*. Então para cada elemento, a matriz de rigidez elementar é alocada, dentro da matriz nula criada, nas linhas e colunas referentes aos nós pertencentes a este elemento. No fim, todas as matrizes criadas para cada elemento estão na mesma ordem e são somadas dando origem a matriz de rigidez global da estrutura.

A Figura 21 ilustra uma malha composta por dois elementos finitos apenas para exemplificar o processo de montagem da matriz de rigidez global.

Figura 21 – Exemplo de malha com dois elementos finitos.



Fonte: Autoria própria.

As equações (72) e (73) representam a matriz de rigidez elementar para o elemento 1 e para o elemento 2, respectivamente. Enquanto que a equação (74) representa a soma matricial, caracterizando a matriz de rigidez global.

$$[k_1]=\begin{bmatrix} k_{11}^{1} & \cdots & k_{14}^{1} & 0 & 0 & k_{15}^{1} & \cdots & k_{18}^{1} & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ k_{41}^{1} & \cdots & k_{44}^{1} & 0 & 0 & k_{45}^{1} & \cdots & k_{48}^{1} & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ k_{51}^{1} & \cdots & k_{54}^{1} & 0 & 0 & k_{55}^{1} & \cdots & k_{58}^{1} & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ k_{81}^{1} & \cdots & k_{84}^{1} & 0 & 0 & k_{85}^{1} & \cdots & k_{88}^{1} & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \tag{72}$$

$$[k_2]=\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & k_{11}^{2} & \cdots & k_{14}^{2} & 0 & 0 & k_{15}^{2} & \cdots & k_{18}^{2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & k_{41}^{2} & \cdots & k_{44}^{2} & 0 & 0 & k_{45}^{2} & \cdots & k_{48}^{2} \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & k_{51}^{2} & \cdots & k_{54}^{2} & 0 & 0 & k_{55}^{2} & \cdots & k_{58}^{2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \cdots \\ 0 & 0 & k_{81}^{2} & \cdots & k_{84}^{2} & 0 & 0 & k_{85}^{2} & \cdots & k_{88}^{2} \end{bmatrix} \tag{73}$$

$$[k]=\begin{bmatrix} k_{11}^{1} & k_{12}^{1} & k_{13}^{1} & k_{14}^{1} & 0 & 0 & k_{15}^{1} & k_{16}^{1} & k_{17}^{1} & k_{18}^{1} & 0 & 0 \\ k_{21}^{1} & k_{22}^{1} & k_{23}^{1} & k_{24}^{1} & 0 & 0 & k_{25}^{1} & k_{26}^{1} & k_{27}^{1} & k_{28}^{1} & 0 & 0 \\ k_{31}^{1} & k_{32}^{1} & k_{33}^{1}+k_{11}^{2} & k_{34}^{1}+k_{12}^{2} & k_{13}^{2} & k_{14}^{2} & k_{35}^{1} & k_{36}^{1} & k_{37}^{1}+k_{15}^{2} & k_{38}^{1}+k_{16}^{2} & k_{17}^{2} & k_{18}^{2} \\ k_{41}^{1} & k_{42}^{1} & k_{43}^{1}+k_{21}^{2} & k_{44}^{1}+k_{22}^{2} & k_{23}^{2} & k_{24}^{2} & k_{45}^{1} & k_{46}^{1} & k_{47}^{1}+k_{25}^{2} & k_{48}^{1}+k_{26}^{2} & k_{27}^{2} & k_{28}^{2} \\ 0 & 0 & k_{31}^{2} & k_{32}^{2} & k_{33}^{2} & k_{34}^{2} & 0 & 0 & k_{35}^{2} & k_{36}^{2} & k_{37}^{2} & k_{38}^{2} \\ 0 & 0 & k_{41}^{2} & k_{42}^{2} & k_{43}^{2} & k_{44}^{2} & 0 & 0 & k_{45}^{2} & k_{46}^{2} & k_{47}^{2} & k_{48}^{2} \\ k_{51}^{1} & k_{52}^{1} & k_{53}^{1} & k_{54}^{1} & 0 & 0 & k_{55}^{1} & k_{56}^{1} & k_{57}^{1} & k_{58}^{1} & 0 & 0 \\ k_{61}^{1} & k_{62}^{1} & k_{63}^{1} & k_{64}^{1} & 0 & 0 & k_{65}^{1} & k_{66}^{1} & k_{67}^{1} & k_{68}^{1} & 0 & 0 \\ k_{71}^{1} & k_{72}^{1} & k_{73}^{1}+k_{51}^{2} & k_{74}^{1}+k_{52}^{2} & k_{53}^{2} & k_{54}^{2} & k_{75}^{1} & k_{76}^{1} & k_{77}^{1}+k_{55}^{2} & k_{78}^{1}+k_{56}^{2} & k_{57}^{2} & k_{58}^{2} \\ k_{81}^{1} & k_{82}^{1} & k_{83}^{1}+k_{61}^{2} & k_{84}^{1}+k_{62}^{2} & k_{63}^{2} & k_{64}^{2} & k_{85}^{1} & k_{86}^{1} & k_{87}^{1}+k_{65}^{2} & k_{88}^{1}+k_{66}^{2} & k_{67}^{2} & k_{68}^{2} \\ 0 & 0 & k_{71}^{2} & k_{72}^{2} & k_{73}^{2} & k_{74}^{2} & 0 & 0 & k_{75}^{2} & k_{76}^{2} & k_{77}^{2} & k_{78}^{2} \\ 0 & 0 & k_{81}^{2} & k_{82}^{2} & k_{83}^{2} & k_{84}^{2} & 0 & 0 & k_{85}^{2} & k_{86}^{2} & k_{87}^{2} & k_{88}^{2} \end{bmatrix} \tag{74}$$

Fica claro que, à medida que se aumenta a quantidade de nós existentes na malha se aumenta a ordem da matriz de rigidez global e cada matriz elementar é alocada de acordo com os nós pertencentes ao elemento que ela representa.

3.2.8 Vetor de Forças Nodais

Nesta sub-rotina o código utiliza o que foi informado pelo usuário nas sub-rotinas "Condições de Vinculação" e "Definição de Carregamento" para estabelecer o vetor de forças externas da estrutura analisada.

Utilizando a numeração global dos nós da estrutura é realizada a montagem do vetor indicando o valor da força atuante em cada nó, tanto na direção horizontal quanto na direção vertical. Caso o nó esteja com deslocamento restrito isto indica que nesse ponto existe uma reação de apoio, que nesse vetor é indicada apenas por 'R'. Lembrando que esse vetor em momento algum aparece para o usuário durante a execução do programa, ele é apenas um dos mecanismos do procedimento de cálculo.

A equação (75) ilustra um exemplo de vetor de forças para a estrutura de chapa mostrada na Figura 21 e submetida a um carregamento qualquer.

$$\{F\}=\begin{bmatrix} R_{1X} \\ R_{1Y} \\ F_{2X} \\ F_{2Y} \\ F_{3X} \\ F_{3Y} \\ R_{4X} \\ R_{4Y} \\ F_{5X} \\ F_{5Y} \\ F_{6X} \\ F_{6Y} \end{bmatrix} \tag{75}$$

3.2.9 Cálculo dos Deslocamentos Nodais e Reações de Apoio

A matriz de rigidez da estrutura será sempre uma matriz singular, isto é, com determinante nulo, o que lhe torna uma matriz não inversível, impossibilitando assim a resolução do sistema linear de equações algébricas estabelecido na equação (71). Assim sendo, nessa sub-rotina o programa é responsável por estabelecer as condições de contorno do problema, alterando a matriz de rigidez de forma a tornar possível a resolução do sistema.

Estas condições de contorno se referem aos pontos nodais que possuem deslocamento restrito, isto é, possui algum tipo de impedimento em um ou mais dos seus graus de liberdade. Para isso o programa utiliza-se da chamada “Regra do 0-1” que consiste em alterar as linhas e colunas referentes aos nós com deslocamentos restritos na matriz de rigidez e no vetor de forças.

Inicialmente altera-se o vetor de forças nos campos referentes aos nós que possuem deslocamento restrito, para isso troca-se o valor da reação de apoio indicado por ‘R’ por um valor nulo. Após isso, altera-se a matriz de rigidez da estrutura, tornando todas as posições das linhas e colunas referente aos nós restritos com valores nulos, exceto a posição referente a diagonal principal que recebe valor unitário. Esse procedimento está exemplificado de forma genérica na equação (76):

$$\begin{bmatrix} 0 \\ 0 \\ F_{2X} \\ F_{2Y} \\ F_{3X} \\ F_{3Y} \\ 0 \\ 0 \\ \vdots \\ \vdots \\ F_{nX} \\ F_{nY} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & k_{33} & k_{34} & k_{35} & k_{36} & 0 & 0 & k_{39} & \cdots & \cdots & k_{3,2n} \\ 0 & 0 & k_{43} & k_{44} & k_{45} & k_{46} & 0 & 0 & k_{49} & \cdots & \cdots & k_{4,2n} \\ 0 & 0 & k_{53} & k_{54} & k_{55} & k_{56} & 0 & 0 & k_{59} & \cdots & \cdots & k_{5,2n} \\ 0 & 0 & k_{63} & k_{64} & k_{65} & k_{66} & 0 & 0 & k_{69} & \cdots & \cdots & k_{6,2n} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & k_{93} & k_{94} & k_{95} & k_{96} & 0 & 0 & k_{99} & \cdots & \cdots & k_{9,2n} \\ \vdots & & & & \vdots & & & \vdots & & \ddots & & \vdots \\ \vdots & & & & \vdots & & & \vdots & & & \ddots & \vdots \\ \cdots & & & & \cdots & & & \cdots & & & & k_{2n,2n} \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \\ \vdots \\ \vdots \\ u_n \\ v_n \end{bmatrix} \tag{76}$$

Após ter alterado as matrizes, o programa então inverte a matriz de rigidez e multiplica esta pelo vetor de forças, realizando assim o procedimento matemático necessário para a resolução do sistema e determinação dos deslocamentos nodais. Após isso, o programa retorna com a matriz de rigidez antes de ter sido alterada e multiplica esta pelo vetor de deslocamentos determinado anteriormente, resultando assim no vetor de forças contendo os valores das reações de apoio.

Após esses procedimentos matemáticos o programa inicia a etapa de saída de dados, apresentando em arquivo de texto os resultados encontrados, tanto para os deslocamentos quanto para as reações, separados em direção horizontal e direção vertical para facilitar a visualização e o entendimento.

3.2.10 Separação dos Deslocamentos Nodais por Elemento

Nesta sub-rotina o programa separa os deslocamentos nodais encontrados anteriormente por elemento, de acordo com os nós pertencentes a cada elemento. Este procedimento é importante para criar o vetor de deslocamentos utilizado na determinação das deformações nodais.

3.2.11 Cálculo das Deformações e Tensões Nodais

Após os procedimentos realizados na sub-rotina anterior, cada elemento apresenta um vetor de deslocamentos que será utilizado na equação (34) para se determinar as deformações nodais deste elemento.

Além disso, também de acordo com o que foi deduzido, o programa utiliza o vetor de deformações de cada elemento na equação (21) para se determinar as tensões atuantes em cada nó deste mesmo elemento.

Por fim, esses resultados de deformações e tensões são armazenados em listas dentro do código, separados por elemento. Estas listas são então utilizadas na etapa posterior.

3.2.12 Cálculo das Deformações e Tensões Nodais Médias

Esta sub-rotina faz parte da etapa de saída de dados. Nela, o programa utiliza as listas criadas na sub-rotina anterior e para cada um dos nós da malha faz o cálculo da média das deformações e tensões encontradas. Isto se faz necessário pois para os nós que são comuns para dois ou mais elementos poderá ocorrer de os valores serem ligeiramente divergentes ao se calcular por um ou por outro elemento.

Após realizados os cálculos, o programa mostra em arquivo de texto os resultados encontrados, separados por tipo de deformação ou tensão e para cada um dos nós.

3.2.13 Saída de Dados

Nesta sub-rotina o programa exporta em arquivo de texto os resultados que foram obtidos durante a análise do problema. Esse arquivo fica armazenado na mesma pasta onde se encontra o código fonte. Todos os resultados são apresentados por pontos nodais e com as unidades mostradas ao lado.

Os resultados de deslocamentos ficam separados por deslocamentos na horizontal (deslocamento x) e deslocamentos na vertical (deslocamento y). As forças nodais são apresentadas separadamente em forças na direção x e forças na direção y. Para as deformações e tensões os resultados também são apresentados separadamente por cada um dos tipos.

No Apêndice B é mostrado uma cópia do arquivo de saída gerado pelo código desenvolvido. A cópia em questão trata-se do arquivo gerado para a análise do Exemplo 1, que é mostrado no Capítulo 4 deste trabalho, utilizando-se uma malha 5x5 nós (16 elementos *CSQ*).

4 RESULTADOS E DISCUSSÕES

Com o objetivo de validar o programa desenvolvido ao longo deste trabalho, foram analisados alguns exemplos de chapas. Os exemplos foram retirados da literatura ou então elaborados pelo próprio autor e executados em programas desenvolvidos por terceiros, e que possuam livre utilização.

Um dos softwares utilizados neste trabalho foi o desenvolvido por Bruno Cesarino Soares e Rodrigo E. Niquini da Costa em seus mestrados na Universidade Federal de Minas Gerais, sob supervisão do Prof. Roque Luiz da Silva Pitangueira, que é destinado a realizar análises estruturais utilizando elementos finitos bidimensionais, e que recebeu o nome de MEFOBJ.

Outra ferramenta utilizada para a validação do programa desenvolvido foi o software *ANSYS Workbench* 19.2 (versão educacional). O *ANSYS* é um software conhecido e renomado em análises via elementos finitos, sendo muito utilizado por diversas áreas da ciência, tais como engenharias, medicina, física e matemática.

Além disso, os softwares *SCIA Engineer* 18.1 (versão educacional) e *Ftool* também foram utilizados para a comparação de resultados. Ambos os softwares são renomados na análise estrutural e possuem amplo caráter de utilização educacional.

Em alguns exemplos os resultados computacionais, fornecidos pelo programa desenvolvido, foram comparados com os resultados obtidos por outros softwares ou pela solução analítica através do conceito de erro relativo. Este conceito, estabelecido pela equação (77), permite obter a porcentagem da divergência ocorrida entre dois resultados distintos. Assim sendo, quanto maior o erro relativo menor foi a aproximação entre os resultados.

$$Er\ (\%) = \left| \left(\frac{r_o - r_t}{r_t} \right) \cdot 100 \right| \tag{77}$$

Onde, r_o = resultado obtido pelo programa desenvolvido;

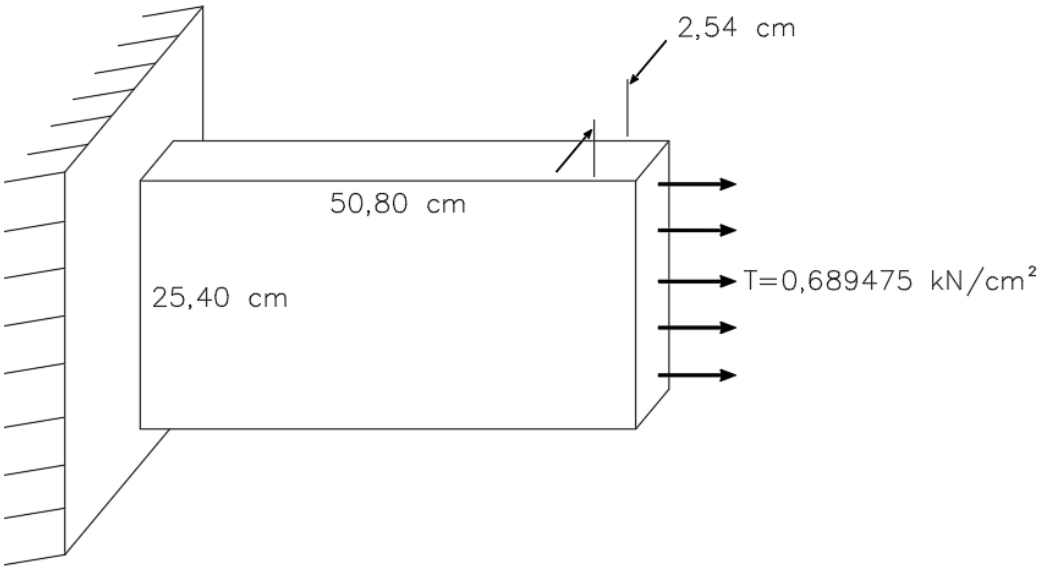
r_t = resultado obtido por software/solução analítica;

4.1 Exemplo 1

O primeiro exemplo foi proposto por Logan (2007) e trata de um problema envolvendo uma chapa tracionada, submetida a um carregamento uniformemente distribuído em sua seção transversal. A chapa está engastada em uma de suas extremidades, e possui seção transversal e material constante ao longo de todo o seu comprimento.

As unidades apresentadas no problema proposto foram transformadas para o Sistema Internacional de Unidades. Os dados fornecidos para o problema foram: espessura da chapa $t = 2,54\ cm$, módulo de elasticidade $E = 2,068426 \times 10^4$ kN/cm², coeficiente de Poisson $\upsilon = 0,3$. A Figura 22 ilustra o problema, indicando as características geométricas, vinculação e carregamento da chapa.

Figura 22 – Exemplo 1: Chapa tracionada com carregamento distribuído.



Fonte: Adaptado de Logan (2007).

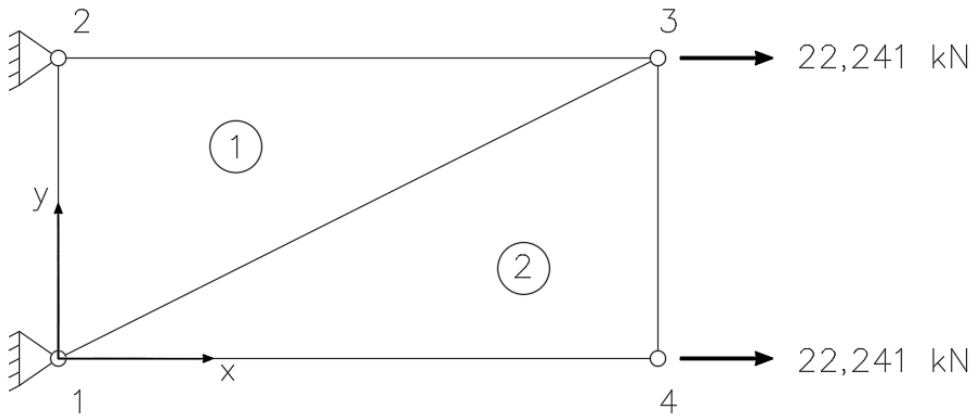
Logan (2007) propõe a solução analítica do problema e numérica utilizando elementos finitos triangulares *CST* (*Constant Strain Triangular*). Para isso, o autor utilizou uma malha formada por dois elementos, assim como ilustra a Figura 23. Além disso, como o carregamento aplicado na chapa é um carregamento de superfície foi

necessário transformá-lo em forças equivalentes aplicadas aos nós da extremidade. Esse procedimento foi feito de acordo com a equação (78):

$$F = \frac{T \bullet A}{2} = \frac{0,689475 \bullet 2,54 \bullet 25,40}{2} = 22,241kN \tag{78}$$

A Figura 23 ilustra a malha utilizada por Logan (2007) em sua solução, bem como o carregamento equivalente.

Figura 23 – Malha formada por dois elementos CST.

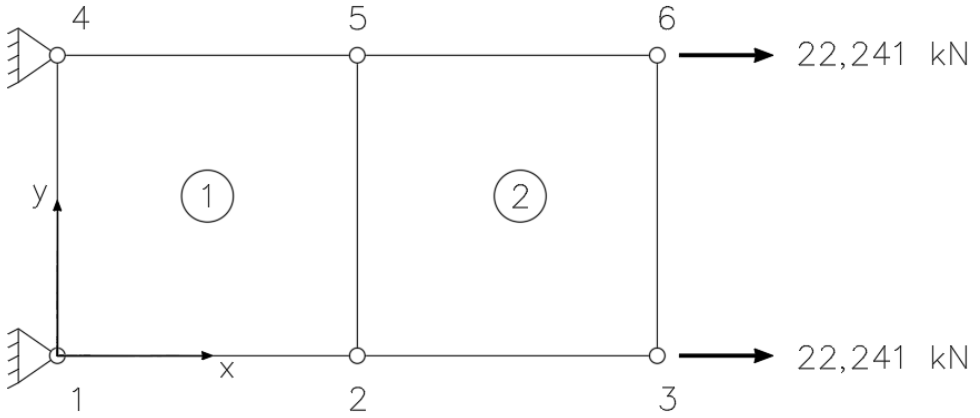


Fonte: Adaptado de Logan (2007).

O programa desenvolvido neste trabalho foi implementado para trabalhar com elementos finitos *CSQ* (*Constant Strain Quadrilateral*). Dessa forma, para o exemplo da Figura 22 pode-se realizar a comparação de resultados entre a solução numérica via elementos *CST* proposta por Logan (2007) e a solução via elementos *CSQ* utilizada pelo programa.

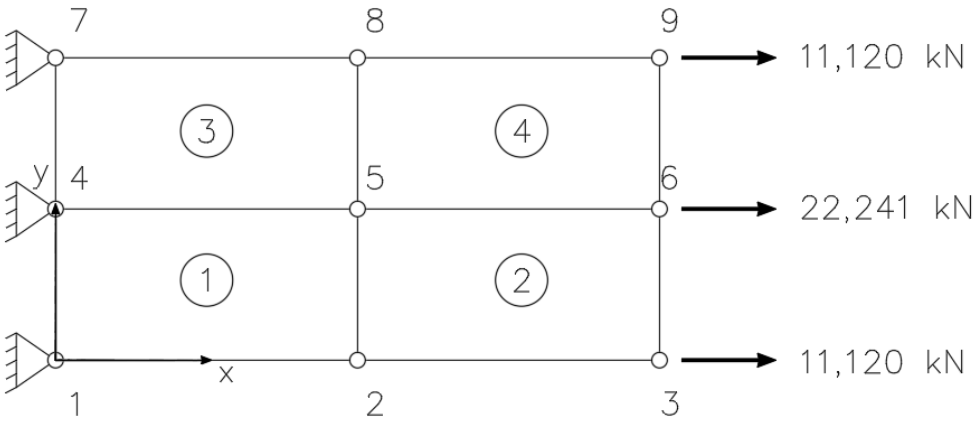
Para efeitos de comparação de resultados, este mesmo exemplo foi analisado no programa desenvolvido utilizando diferentes malhas. Para todas as malhas foi necessário transformar o carregamento distribuído na superfície da chapa em um carregamento nodal equivalente. O procedimento utilizado para isso foi dividir o carregamento entre os nós, de acordo com as áreas de influência. As Figura 24, Figura 25 e Figura 26 ilustram respectivamente, as malhas com dois, quatro e dezesseis elementos, utilizadas para a execução do problema e comparação dos resultados.

Figura 24 – Malha com dois elementos CSQ.



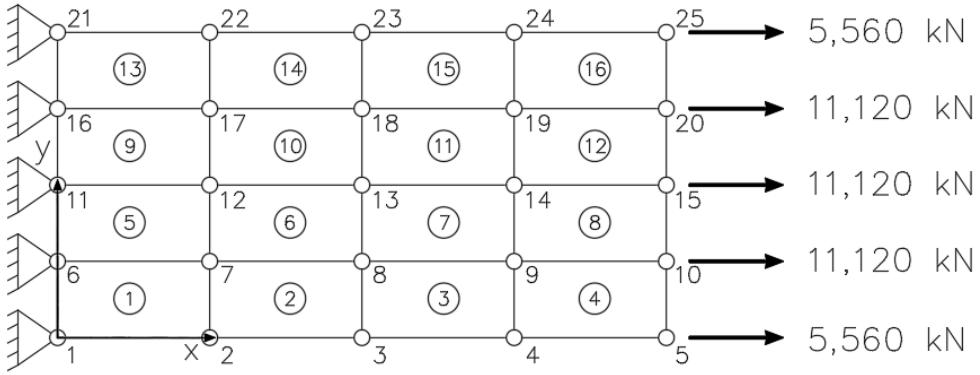
Fonte: Autoria própria.

Figura 25 – Malha com quatro elementos CSQ.



Fonte: Autoria própria.

Figura 26 – Malha com dezesseis elementos CSQ.



Fonte: Autoria própria.

A Tabela 2 indica a comparação dos resultados obtidos para os deslocamentos nodais entre o programa desenvolvido e os fornecidos por Logan (2007), tendo como base a numeração de nós utilizada pelo próprio autor e ilustrada na Figura 23.

Tabela 2 - Comparação de resultados para deslocamentos – Exemplo 1.

	Programa Desenvolvido (Malha com 2 elementos CSQ)		Logan (2007) (Malha com 2 elementos CST)	
Nó	**Deslocamento em x (cm)**	**Deslocamento em y (cm)**	**Deslocamento em x (cm)**	**Deslocamento em y (cm)**
1	0,00000000	0,00000000	0,00000000	0,00000000
2	0,00000000	0,00000000	0,00000000	0,00000000
3	0,00166658	-0,00011929	0,00154838	0,00001066
4	0,00166658	0,00011929	0,00168579	0,00026441

Fonte: Autoria Própria.

A Tabela 2 indica que os resultados ficaram muito próximos para os deslocamentos na direção horizontal, enquanto que houveram divergências para os deslocamentos na direção vertical. Logan (2007) destaca que a malha utilizada por ele acaba sendo muito grosseira, o que acaba resultando em erros na aproximação dos resultados. Por exemplo, o autor afirma que devido ao efeito de Poisson, o deslocamento em y para o nó 3 deveria ser negativo, indicando um deslocamento vertical para baixo, enquanto que para o nó 4 deveria ser positivo, indicando um deslocamento para cima. Dessa forma, a diferença encontrada pode ser atribuída ao fraco refinamento da malha utilizada pelo autor, além dos erros numéricos que são inerentes a esse processo. Além disso, pode-se afirmar que o programa desenvolvido conseguiu alcançar resultados melhores utilizando um mesmo refinamento de malha, o que indica que a utilização de elementos *CSQ* é mais precisa do que a utilização de elementos *CST*.

Logan (2007) destaca que o elemento *CSQ* consegue obter resultados mais satisfatórios que o elemento *CST* utilizando a mesma quantidade de elementos finitos na malha. Isto ocorre devido ao fato de que a interpolação dos deslocamentos no caso do elemento *CSQ* é feita utilizando funções de graus maiores que no caso do elemento *CST*.

Ainda para efeito de comparação dos resultados se utilizou o software MEFOBJ para analisar a chapa. Assim sendo, as três diferentes malhas de elementos CSQ, mostradas anteriormente, foram utilizadas para executar o problema no programa

desenvolvido neste trabalho e no MEFOBJ.

Nesse momento se torna importante salientar que, diferente do software desenvolvido neste trabalho que usa a integração analítica, o programa MEFOBJ trabalha por meio de integração de Gauss, o que faz necessário indicar no programa a quantidade de pontos de Gauss desejada. Para este trabalho, todas as análises foram realizadas utilizando 20 pontos de Gauss, de forma a aumentar a precisão nos resultados obtidos.

Além disso, para os deslocamentos o programa MEFOBJ apresenta os resultados por pontos nodais, assim como o programa desenvolvido neste trabalho. Porém, para as tensões esses resultados são apresentados por ponto nodal em cada um dos elementos. Então, para os nós que são comuns a dois ou mais elementos, ocorrerá divergência de valores ao se calcular por um ou por outro elemento. Dessa forma, para se chegar aos resultados apresentados nesse capítulo foi necessário fazer o cálculo manual da média aritmética simples para esses casos. Esse procedimento é o mesmo realizado pelo programa desenvolvido neste trabalho, porém nesse caso ele foi programado para ser feito de forma automática pelo computador.

As Tabela 3 e Tabela 4 indicam os resultados encontrados para deslocamentos nodais em cada um dos programas, seguindo como base a numeração de nós utilizada por Logan (2007) e ilustrada na Figura 23.

Tabela 3 - Comparação de resultados para deslocamentos horizontais – Exemplo 1.

Deslocamento em x (cm)						
Quantidade de Elementos	**2**		**4**		**16**	
Nó	**Programa Desenvolvido**	**MEFOBJ Versão 1.1**	**Programa Desenvolvido**	**MEFOBJ Versão 1.1**	**Programa Desenvolvido**	**MEFOBJ Versão 1.1**
1	0,00000000	0,00000000	0,00000000	0,00000000	0,00000000	0,00000000
2	0,00000000	0,00000000	0,00000000	0,00000000	0,00000000	0,00000000
3	0,00166658	0,00166658	0,00165939	0,00165939	0,00167643	0,00167642
4	0,00166658	0,00166658	0,00165939	0,00165939	0,00167643	0,00167642

Fonte: Autoria própria.

Tabela 4 - Comparação de resultados para deslocamentos verticais – Exemplo 1.

Deslocamento em y (cm)						
Quantidade de Elementos	**2**		**4**		**16**	
Nó	**Programa Desenvolvido**	**MEFOBJ Versão 1.1**	**Programa Desenvolvido**	**MEFOBJ Versão 1.1**	**Programa Desenvolvido**	**MEFOBJ Versão 1.1**
1	0,00000000	0,00000000	0,00000000	0,00000000	0,00000000	0,00000000
2	0,00000000	0,00000000	0,00000000	0,00000000	0,00000000	0,00000000
3	-0,00011929	-0,00011929	-0,00011968	-0,00011968	-0,00012644	-0,00012644
4	0,00011929	0,00011929	0,00011968	0,00011968	0,00012644	0,00012644

Fonte: Autoria própria.

As Tabela 3 e Tabela 4 indicam que a comparação entre os dois programas foi satisfatória, estabelecendo resultados exatamente iguais em valores até a sétima casa decimal. Houveram divergências de valores apenas após a oitava casa decimal, o que pode ser atribuído a precisão numérica no cálculo computacional.

Logan (2007) ainda destaca que para uma chapa engastada, submetida a carregamento de tração, a solução analítica para o deslocamento da extremidade livre é dada pela equação (79):

$$\delta = \frac{T \bullet L}{E \bullet A} \tag{79}$$

Onde, δ = Deslocamento da extremidade livre;

T = Força normal de tração;

L = Comprimento da chapa;

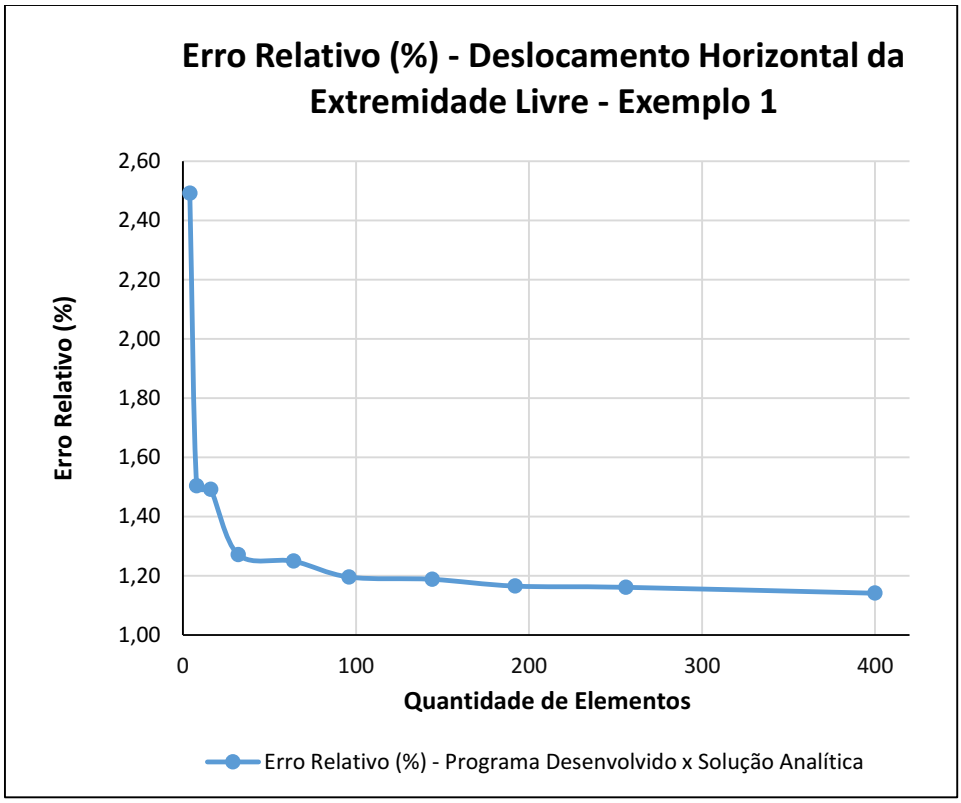
E = Módulo de elasticidade;

A = Área de seção transversal;

Assim sendo, o programa desenvolvido foi utilizado para analisar o problema da Figura 22 em malhas com diferentes quantidades de elementos, bem como a equação (79) foi utilizada para chegar-se em um valor numérico de deslocamento através de solução analítica.

Feito isso, a equação (78) foi utilizada para calcular o erro relativo (*Er*) entre o resultado computacional fornecido pelo programa desenvolvido e a solução analítica. Dessa forma, foi possível visualizar a influência do refinamento da malha na precisão dos resultados, como está ilustrado no Gráfico 1.

Gráfico 1 - Erro relativo para deslocamento horizontal do exemplo 1.



Fonte: Autoria própria.

Através do Gráfico 1 é possível visualizar que a medida que se aumenta a quantidade de elementos utilizados, o valor encontrado para o deslocamento da extremidade livre da chapa vai se aproximando do valor encontrado através da solução analítica, reduzindo assim a porcentagem de erro relativo. Pode-se notar que a partir de uma determinada quantidade de elementos, a melhora na aproximação foi muito pouca, não justificando mais o refinamento da malha e comprovando a melhora nos resultados à medida que se tem um maior refinamento da malha.

Os resultados obtidos em termos das tensões nodais também foram comparados com os resultados fornecidos pelo programa MEFOBJ. Para esta comparação foi utilizada a malha com dezesseis elementos finitos, ilustrada na Figura 26. A Tabela 5 mostra os resultados obtidos em ambos os programas.

Tabela 5 - Comparação de resultados para tensões normais na direção x – Exemplo 1.

Tensão Normal Direção x (kN/cm²)					
Quantidade de Elementos	**16**				
Nó	**21**	**22**	**23**	**24**	**25**
Programa Desenvolvido	0,7609	0,6666	0,6734	0,6888	0,6907
MEFOBJ	0,7606	0,6669	0,6735	0,6889	0,6908
Nó	**16**	**17**	**18**	**19**	**20**
Programa Desenvolvido	0,7154	0,6722	0,6941	0,6889	0,6897
MEFOBJ	0,7153	0,6723	0,6942	0,6890	0,6898
Nó	**11**	**12**	**13**	**14**	**15**
Programa Desenvolvido	0,7078	0,6745	0,6958	0,6920	0,6883
MEFOBJ	0,7076	0,6747	0,6958	0,6921	0,6884
Nó	**6**	**7**	**8**	**9**	**10**
Programa Desenvolvido	0,7154	0,6722	0,6941	0,6889	0,6897
MEFOBJ	0,7153	0,6723	0,6942	0,6890	0,6898
Nó	**1**	**2**	**3**	**4**	**5**
Programa Desenvolvido	0,7609	0,6666	0,6734	0,6888	0,6907
MEFOBJ	0,7606	0,6699	0,6735	0,6889	0,6908

Fonte: Autoria Própria.

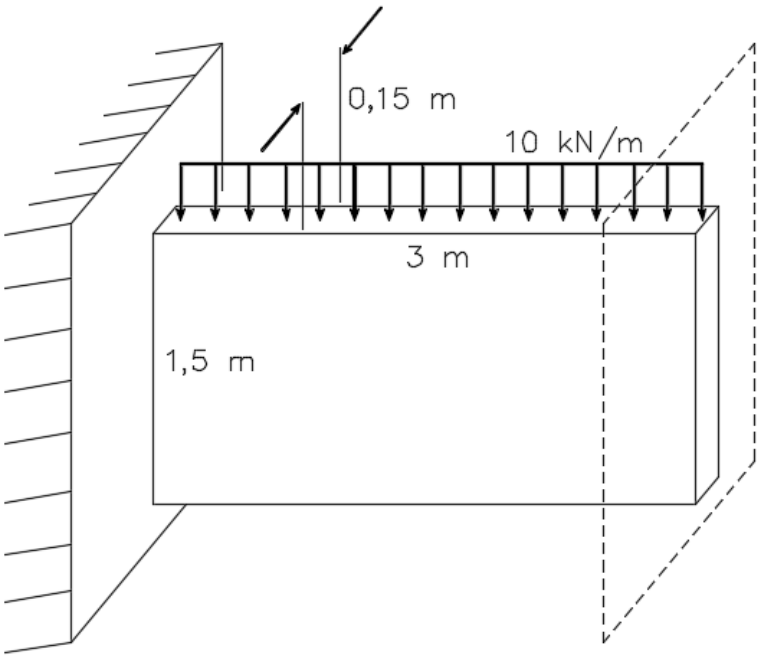
Pela Tabela 5 se pode notar que os resultados obtidos pelo programa desenvolvido neste trabalho foram satisfatórios em relação as tensões normais na direção x. Para todos os pontos nodais as duas primeiras casas decimais foram idênticas em ambos os programas. As diferenças ocorridas a partir da terceira casa decimal, podem ser atribuídas aos arredondamentos vindos do cálculo computacional e não possuem influência significativa nos resultados.

Assim sendo, se pode afirmar que para a chapa descrita no exemplo 1 os resultados fornecidos pelo programa foram satisfatórios tanto na comparação com os resultados apresentados por Logan (2007), quanto para a comparação realizada com o software MEFOBJ.

4.2 Exemplo 2

O segundo exemplo trata-se de uma chapa bi-engastada e submetida à ação de um carregamento distribuído em sua superfície superior, assim como ilustra a Figura 27. A chapa possui uma espessura $t = 0{,}15$ m e é feita de um material que possui módulo de elasticidade $E = 30{\times}10^3$ MPa e coeficiente de Poisson $\upsilon = 0{,}2$.

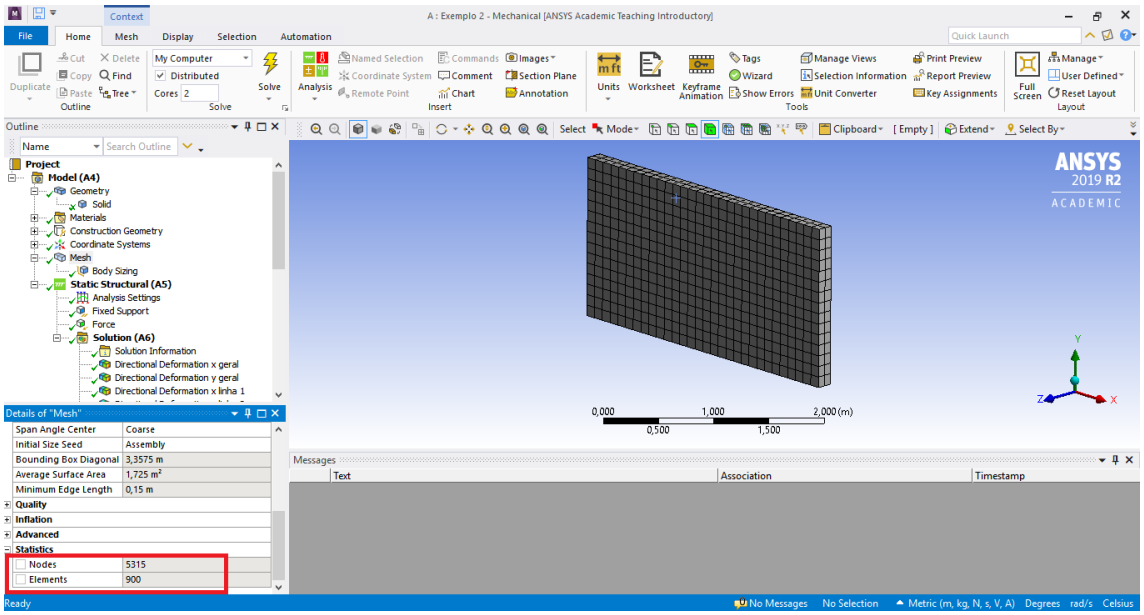
Figura 27 – Exemplo 2: Chapa bi-engastada submetida à carregamento distribuído.



Fonte: Autoria Própria.

Para realizar a validação do programa desenvolvido os resultados foram comparados com os resultados fornecidos pelo software *ANSYS*. A análise no *ANSYS* foi feita através de elementos finitos sólidos, limitando-se a dimensão dos mesmos em no máximo 100 mm. Dessa forma, a malha gerada pelo programa, segundo dados do próprio software, foi formada por 5315 nós distribuídos em 900 elementos finitos tridimensionais, assim como ilustra a Figura 28, que é uma cópia de tela feita durante a execução do problema no software.

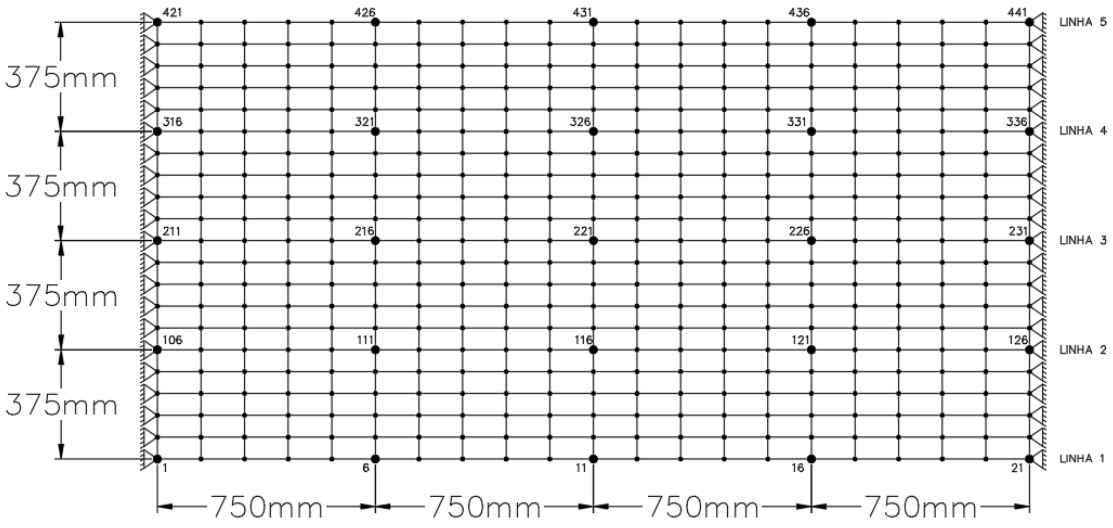
Figura 28 – Malha gerada pelo ANSYS para o exemplo 2.



Fonte: Autoria Própria.

Para realizar a comparação de resultados a mesma chapa foi executada no programa desenvolvido neste trabalho. Para isso, optou-se por utilizar uma malha 21x21 nós, totalizando 400 elementos finitos *CSQ*. A Figura 29 ilustra a malha utilizada.

Figura 29 – Malha 21x21 utilizada no programa para o exemplo 2.



Fonte: Autoria Própria.

A Tabela 6 mostra a comparação dos deslocamentos verticais obtidos pelas análises. Devido à grande quantidade de nós utilizada na malha, as tabelas indicam os resultados apenas para alguns dos pontos nodais.

Tabela 6 - Comparação de deslocamentos verticais Programa Desenvolvido x ANSYS – Exemplo 2.

Deslocamentos Verticais (mm)					
Quantidade de Elementos	**Programa Desenvolvido (400 elementos) ANSYS Workbench 19.2 (900 elementos)**				
Nó	**421**	**426**	**431**	**436**	**441**
Programa Desenvolvido	0,00000000	-0,00538780	-0,00722840	-0,00538780	0,00000000
ANSYS	0,00000000	-0,00542280	-0,00727420	-0,00542280	0,00000000
Erro Relativo (%)	0,00	-0,65	-0,63	-0,65	0,00
Nó	**316**	**321**	**326**	**331**	**336**
Programa Desenvolvido	0,00000000	-0,00467149	-0,00659236	-0,00467149	0,00000000
ANSYS	0,00000000	-0,00470770	-0,00663860	-0,00470770	0,00000000
Erro Relativo (%)	0,00	-0,77	-0,70	-0,77	0,00
Nó	**211**	**216**	**221**	**226**	**231**
Programa Desenvolvido	0,00000000	-0,00414731	-0,00606255	-0,00414731	0,00000000
ANSYS	0,00000000	-0,00418520	-0,00610970	-0,00418520	0,00000000
Erro Relativo (%)	0,00	0,91	0,77	0,91	0,00
Nó	**106**	**111**	**116**	**121**	**126**
Programa Desenvolvido	0,00000000	-0,00388973	-0,00576125	-0,00388973	0,00000000
ANSYS	0,00000000	-0,00392760	-0,00580930	-0,00392760	0,00000000
Erro Relativo (%)	0,00	0,96	0,83	0,96	0,00
Nó	**1**	**6**	**11**	**16**	**21**
Programa Desenvolvido	0,00000000	-0,00381465	-0,00559022	-0,00381465	0,00000000
ANSYS	0,00000000	-0,00385250	-0,00563840	-0,00385250	0,00000000
Erro Relativo (%)	0,00	0,98	0,85	0,98	0,00

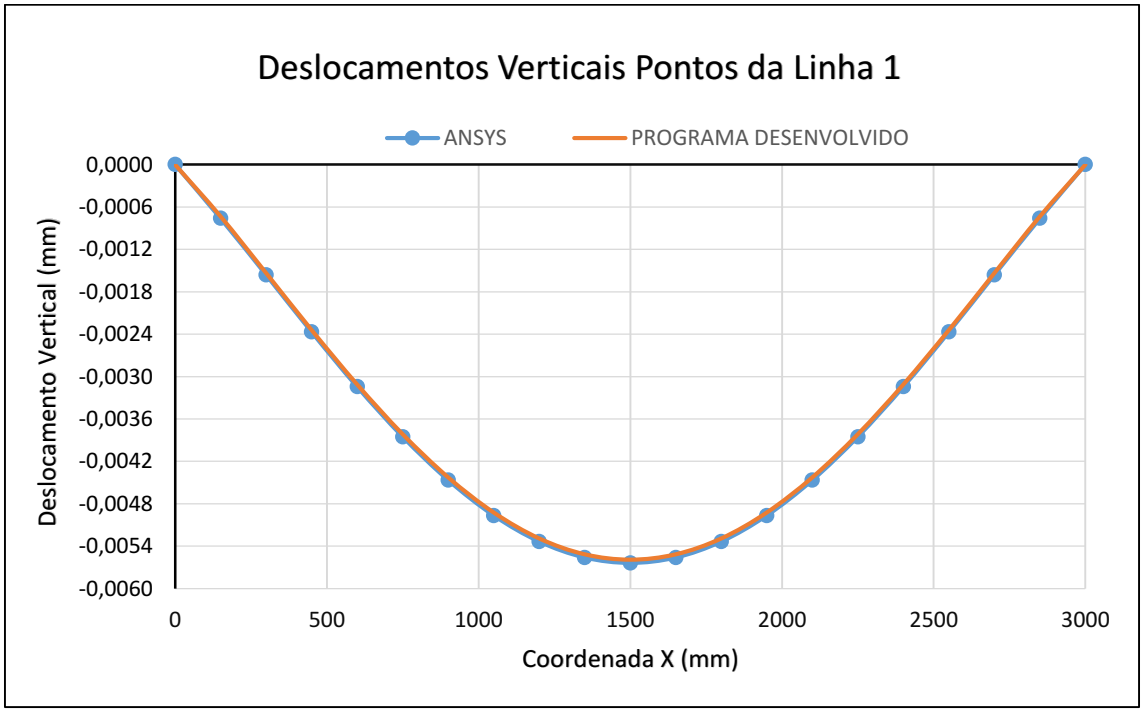
Fonte: Autoria Própria.

Pela Tabela 6 se percebe que para todos os pontos a comparação entre os programas é satisfatória, tendo em vista que em todos eles os resultados foram idênticos até a terceira casa decimal pelo menos. As diferenças ocorridas a partir da quarta decimal podem ser atribuídas à diferença no refinamento utilizado em ambos os programas, uma vez que o ANSYS trabalhou com um refinamento de malha muito maior.

Para facilitar a visualização dos resultados obtidos, pode se utilizar as linhas horizontais de nós destacadas na Figura 29. Com isso, é possível verificar que ambos os programas indicaram o mesmo comportamento em relação aos deslocamentos da chapa, que é também o comportamento esperado para o elemento fletido em questão.

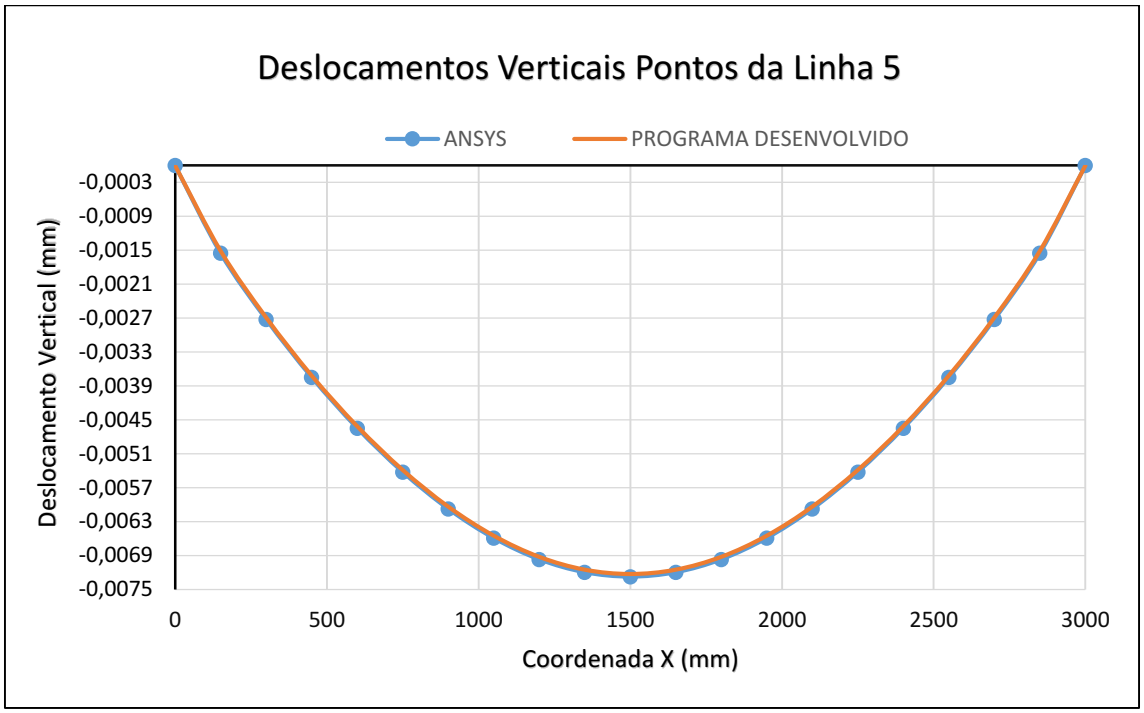
Para os deslocamentos verticais, todos os pontos tiveram deslocamentos negativos, indicando uma movimentação para baixo, como era de se esperar. Além disso, os pontos com maiores deslocamentos em módulo estão situados no centro da chapa, e os pontos próximos ao engaste podem ser considerados com deslocamentos nulos. Os Gráfico 2 e Gráfico 3 ilustram esse comportamento, fazendo uma comparação dos deslocamentos verticais fornecidos pelo ANSYS e pelo programa desenvolvido.

Gráfico 2 - Comparação de deslocamentos verticais dos pontos da linha 1 – Exemplo 2.



Fonte: Autoria Própria.

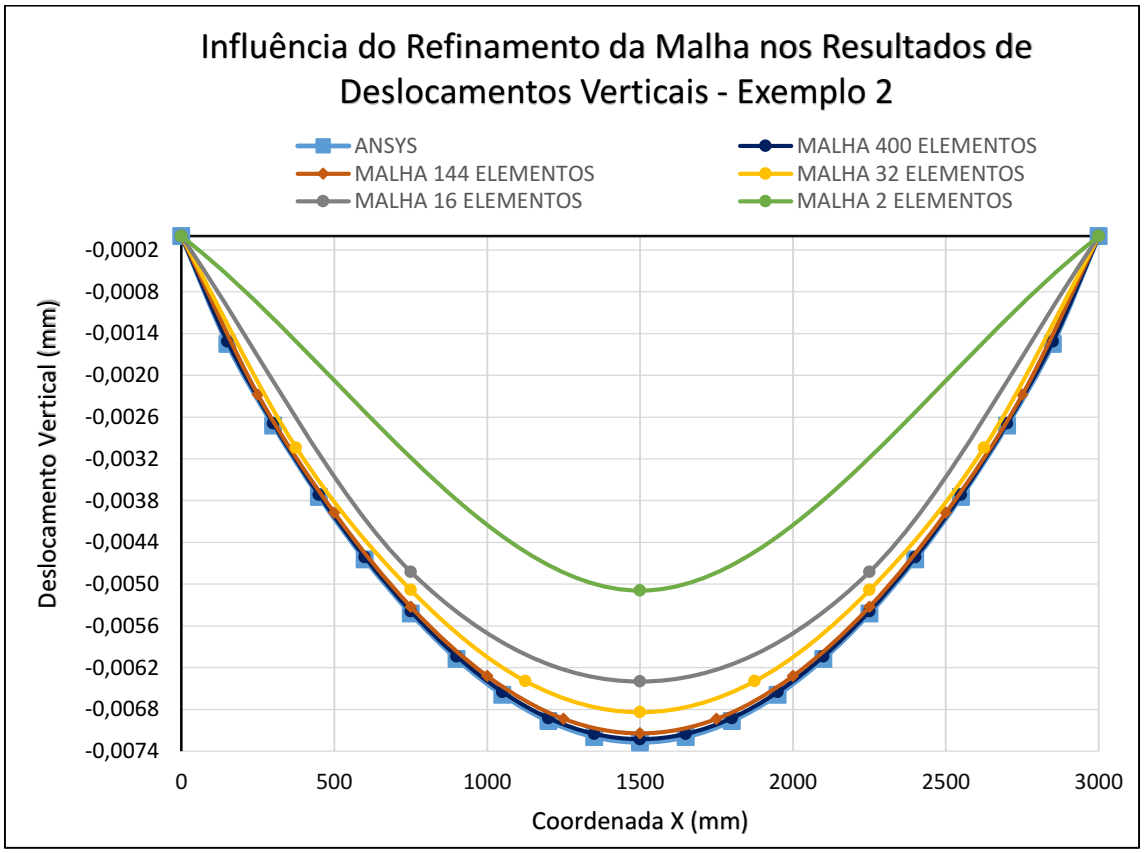
Gráfico 3 - Comparação de deslocamentos verticais dos pontos da linha 5 – Exemplo 2.



Fonte: Autoria Própria.

Além dos resultados já mostrados acima, se realizou uma comparação entre diferentes malhas no programa desenvolvido para se comprovar a melhora na aproximação dos resultados conforme se aumenta o refinamento da malha. Essa comparação foi realizada para os deslocamentos verticais dos pontos situados na borda superior da chapa (linha 5), que são aqueles que apresentam o maior valor de deslocamento em módulo. O Gráfico 4 ilustra os resultados obtidos para as diferentes malhas testadas em comparação com os resultados obtidos da análise realizada via ANSYS.

Gráfico 4 - Influência do refinamento da malha nos resultados de deslocamentos verticais - Exemplo 2.



Fonte: Autoria Própria.

Os dois programas também foram comparados com relação aos valores de tensões normais e de cisalhamento. No programa desenvolvido, a malha utilizada também foi a malha 21x21 ilustrada na Figura 29.

A Tabela 7 mostra os resultados obtidos para as tensões normais na direção do eixo *x*. Da mesma forma que foi para os deslocamentos, são mostrados apenas alguns pontos da malha, mas que já é suficiente para analisar o comportamento.

Tabela 7 - Comparação de tensões normais no eixo x Programa Desenvolvido x ANSYS – Exemplo 2.

Tensões Normais Direção x (MPa)					
Quantidade de Elementos	**Programa Desenvolvido (400 elementos) ANSYS Workbench 19.2 (900 elementos)**				
Nó	**421**	**426**	**431**	**436**	**441**
Programa Desenvolvido	0,24909261	-0,04724774	-0,09258763	-0,04724774	0,24909261
ANSYS	0,38441000	-0,04751200	-0,09349400	-0,04751200	0,38441000
Erro Relativo (%)	35,20	0,56	0,97	0,56	35,20
Nó	**316**	**321**	**326**	**331**	**336**
Programa Desenvolvido	0,01801770	-0,01026224	-0,03564381	-0,01026224	0,01801770
ANSYS	0,01424100	-0,01091000	-0,03645500	-0,01091000	0,01424100
Erro Relativo (%)	26,52	5,94	2,23	5,94	26,52
Nó	**211**	**216**	**221**	**226**	**231**
Programa Desenvolvido	-0,01075444	-0,00357722	-0,00456265	-0,00357722	-0,01075444
ANSYS	-0,01218400	-0,00355010	-0,00468790	-0,00355010	-0,01218400
Erro Relativo (%)	11,73	0,76	2,67	0,76	11,73
Nó	**106**	**111**	**116**	**121**	**126**
Programa Desenvolvido	-0,03528973	0,00063366	0,02439802	0,00063366	-0,03528973
ANSYS	-0,03591000	0,00106580	0,02509700	0,00106580	-0,03591000
Erro Relativo (%)	1,73	40,55	2,79	40,55	1,73
Nó	**1**	**6**	**11**	**16**	**21**
Programa Desenvolvido	-0,18145709	0,02470489	0,07834687	0,02470489	-0,18145709
ANSYS	-0,23936000	0,02528900	0,07932600	0,02528900	-0,23936000
Erro Relativo (%)	24,19	2,31	1,23	2,31	24,19

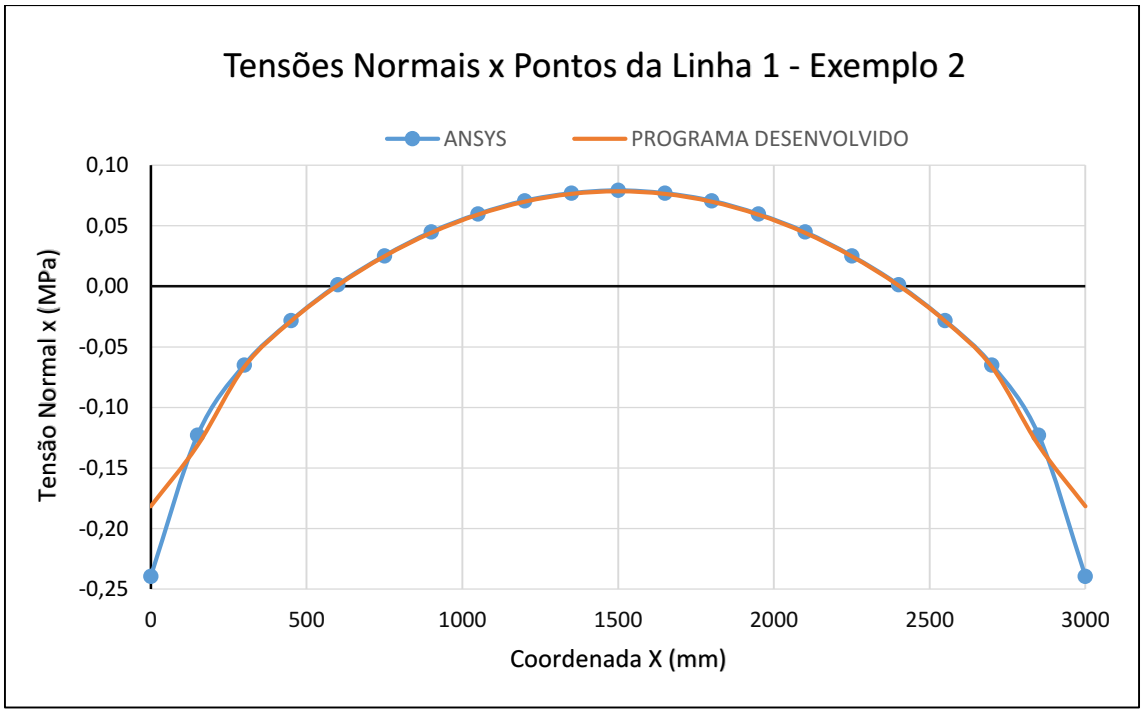
Fonte: Autoria Própria.

Através da Tabela 7 se pode notar que ambos os programas indicaram o mesmo comportamento das tensões normais na direção x. Para a borda inferior da chapa, os pontos situados mais ao centro estão submetidos às tensões de tração, havendo valores de compressão apenas nos pontos mais próximos ao engaste. Essa inversão de sinal surge devido à ocorrência de momentos negativos no engaste. Para a borda superior ocorre o inverso, sendo que os pontos situados ao centro estão submetidos às tensões de

compressão, ocorrendo uma inversão apenas nos pontos próximos aos apoios.

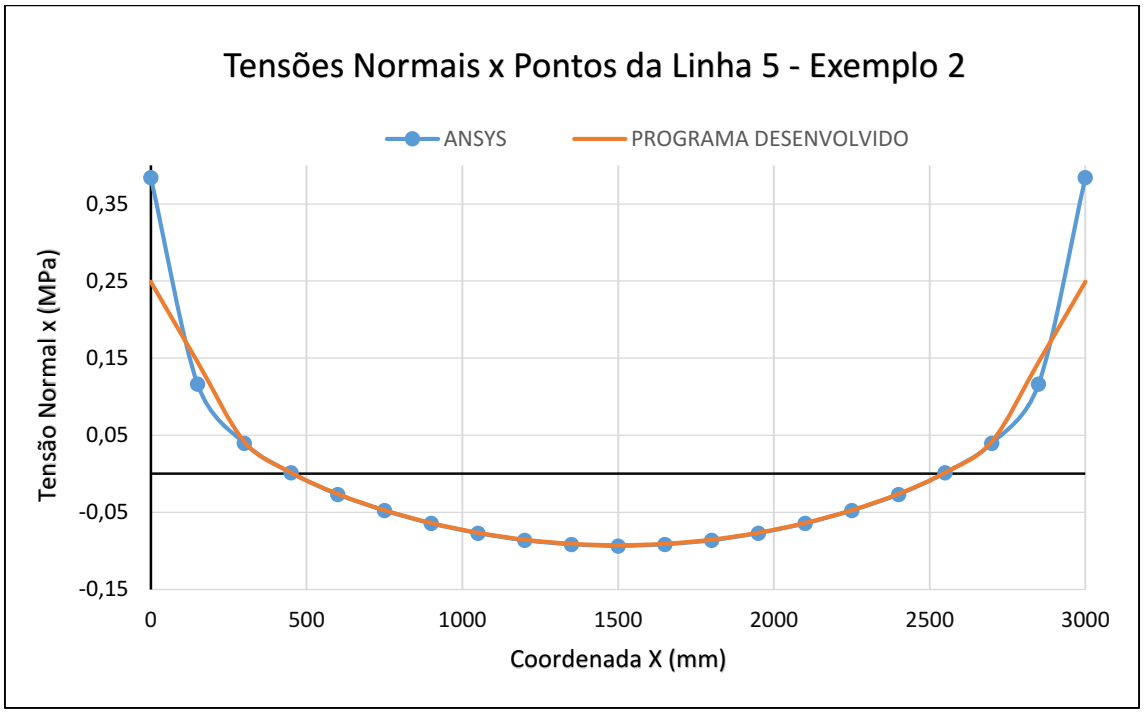
Para facilitar a visualização dos resultados obtidos, os Gráfico 5 e Gráfico 6 ilustram o comportamento das tensões normais na direção x para os pontos situados nas linhas 1 e 5, respectivamente.

Gráfico 5 - Comparação de tensões normais x dos pontos da linha 1 – Exemplo 2.



Fonte: Autoria Própria.

Gráfico 6 - Comparação de tensões normais x dos pontos da linha 5 – Exemplo 2.



Fonte: Autoria Própria.

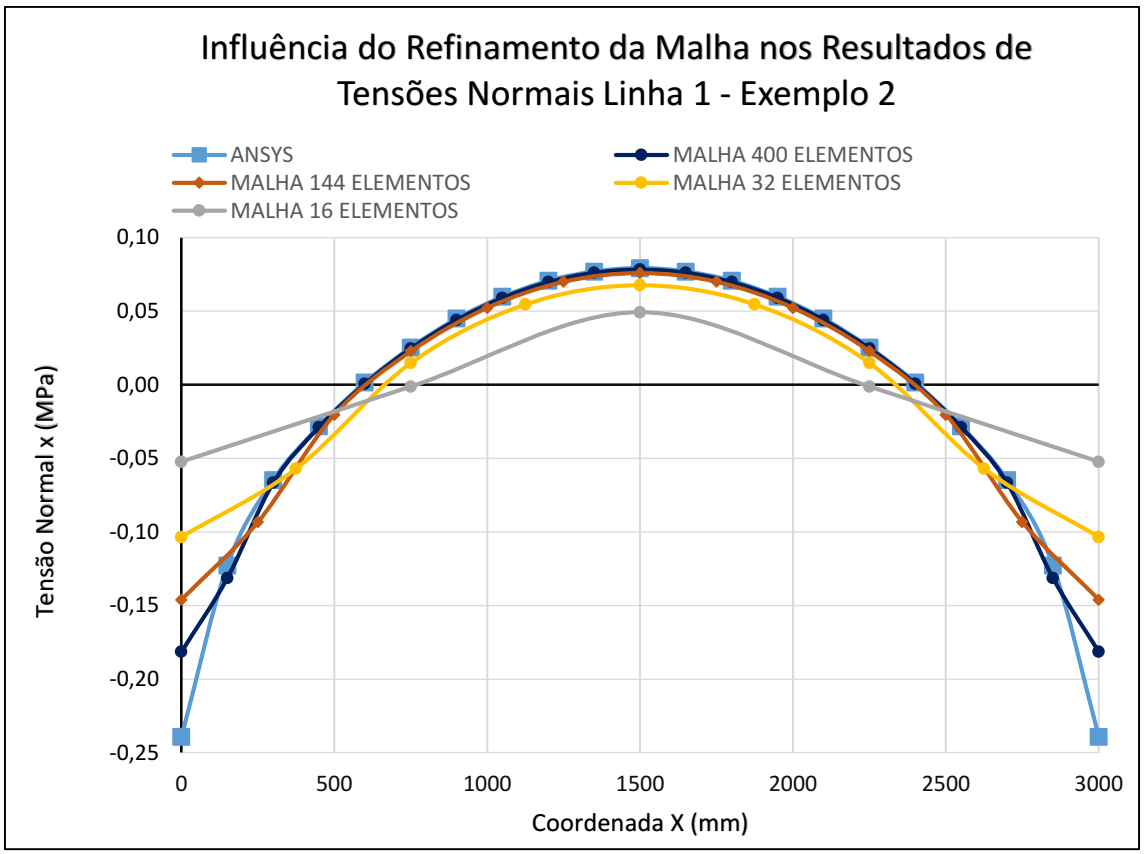
Em relação aos valores se percebe que em alguns pontos, principalmente aqueles situados mais ao centro da chapa, ambos os programas forneceram resultados próximos. Porém em alguns pontos, principalmente aqueles situados nas bordas engastadas, os resultados tiveram divergências. Essas divergências podem ter ocorridos devido aos diferentes refinamentos da malha, uma vez que a malha utilizada pelo ANSYS foi bem mais refinada que aquela utilizada no programa desenvolvido. A solução para isso seria utilizar um maior refinamento de malha nas regiões próximas aos apoios, pois já é de conhecimento geral que essas regiões necessitam de maiores refinamentos devido as maiores concentrações de tensões. Porém, o programa desenvolvido nesse trabalho não permite utilizar um refinamento maior apenas na região dos apoios, isto porque foi desenvolvido para trabalhar de forma com que todos os elementos possuam o mesmo tamanho.

Além disso, as divergências ocorridas devido ao baixo refinamento da malha costumam aparecer mais nos resultados de tensões do que de deslocamentos, o que explica os percentuais de erro relativo serem maiores para as comparações entre tensões. Isso ocorre porque os deslocamentos são a primeira aproximação do método numérico e nesse caso o percentual de erro acumulado é baixo. Como as tensões são calculadas a partir das deformações, que por sua vez são oriundas das derivadas dos deslocamentos, o

percentual de erro acumulado é maior, pois há perda no grau das funções aproximadoras a medida em que são feitas as derivadas.

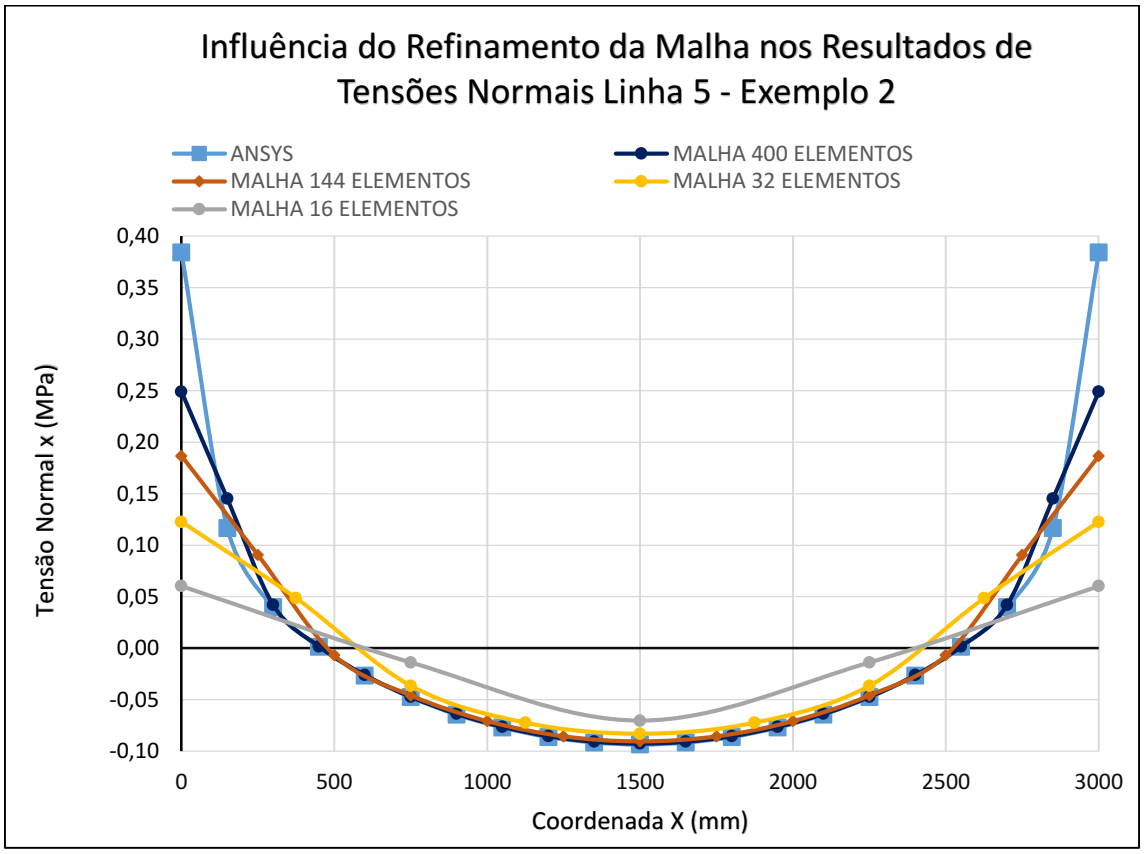
Para avaliar se as divergências ocorreram de fato por causa do refinamento, o programa desenvolvido foi utilizado para executar a chapa deste exemplo utilizando diferentes malhas. Essa comparação foi feita apenas para as linhas 1 e 5, que representam respectivamente os pontos situados na borda inferior e superior da chapa. Os Gráfico 7 e Gráfico 8 mostram os resultados obtidos.

Gráfico 7 - Influência do refinamento da malha nos resultados de tensões normais x para a linha 1 - Exemplo 2.



Fonte: Autoria Própria.

Gráfico 8 - Influência do refinamento da malha nos resultados de tensões normais x para a linha 5 - Exemplo 2.

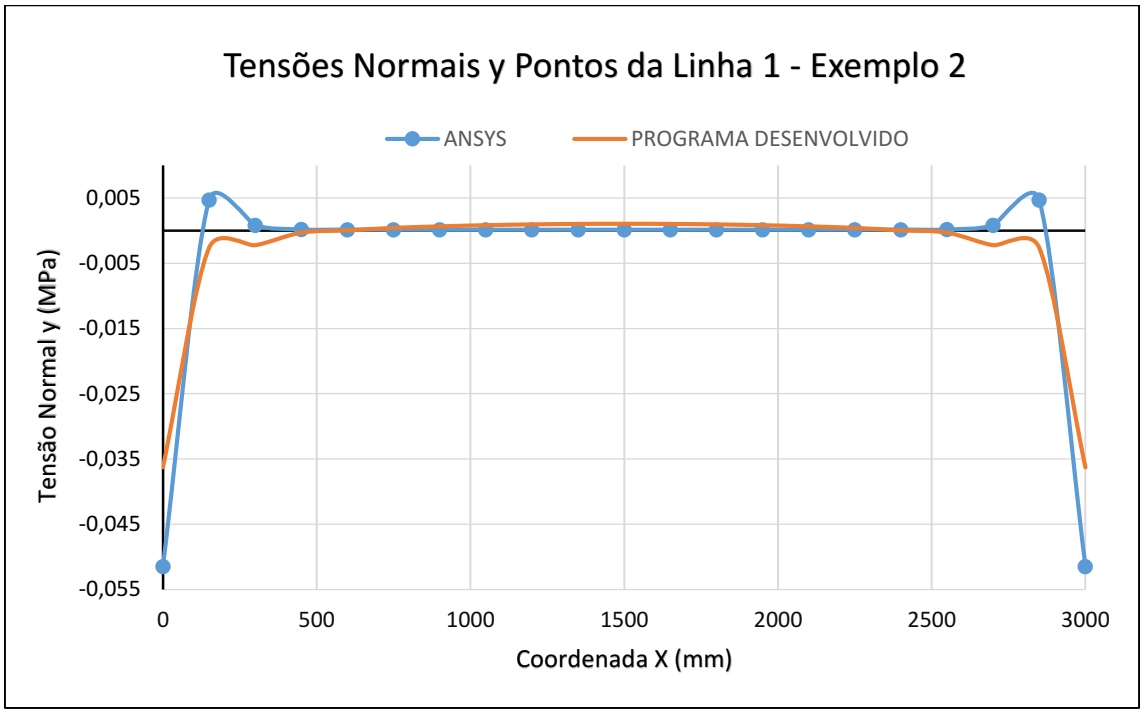


Fonte: Autoria Própria.

Pelos Gráfico 7 e Gráfico 8 é possível visualizar que a medida que se aumenta o refinamento da malha os resultados realmente vão se aproximando dos resultados fornecidos pelo ANSYS. Em contrapartida, esse aumento no refinamento da malha exige um maior cálculo computacional, o que as vezes pode acabar tornando inviável a execução caso o computador não suporte todo esse processamento. Apesar disso, tendo em vista o caráter educacional e didático do programa desenvolvido neste trabalho, os resultados fornecidos por ele podem ser considerados aceitáveis dentro do contexto proposto.

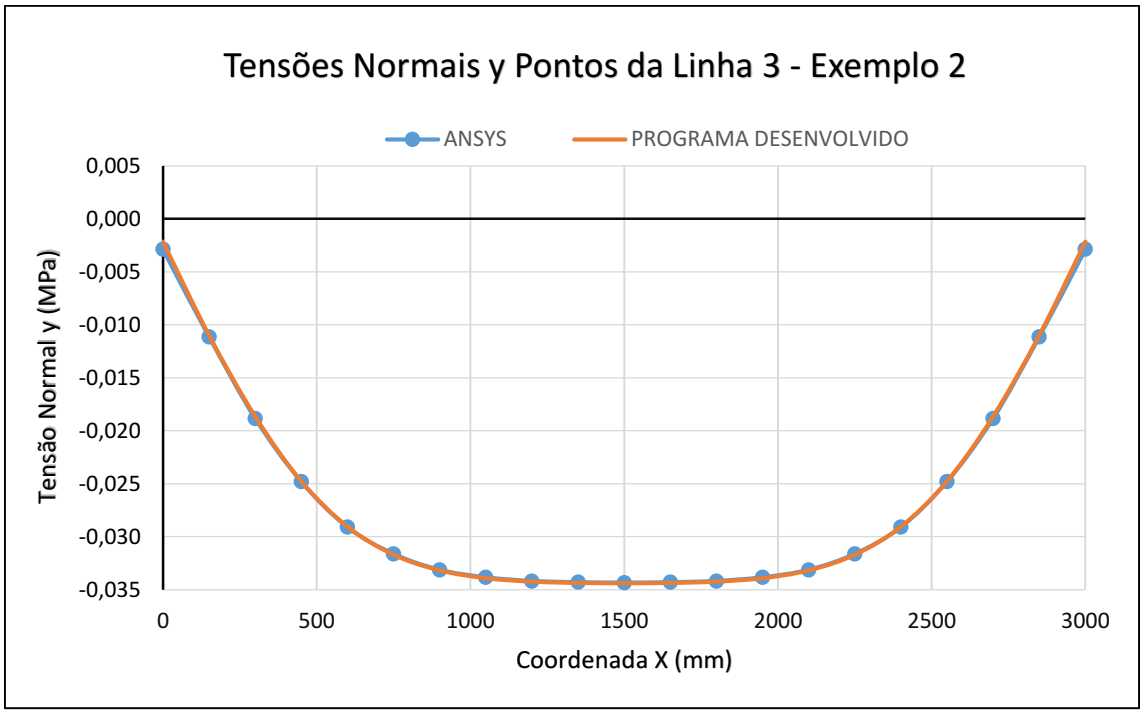
As tensões normais na direção *y* também foram analisadas para a chapa deste exemplo em ambos os programas. Os Gráfico 9, Gráfico 10 e Gráfico 11 ilustram a comparação entre os resultados encontrados para os pontos das linhas 1, 3 e 5, respectivamente.

Gráfico 9 - Comparação de tensões normais y dos pontos da linha 1 – Exemplo 2.



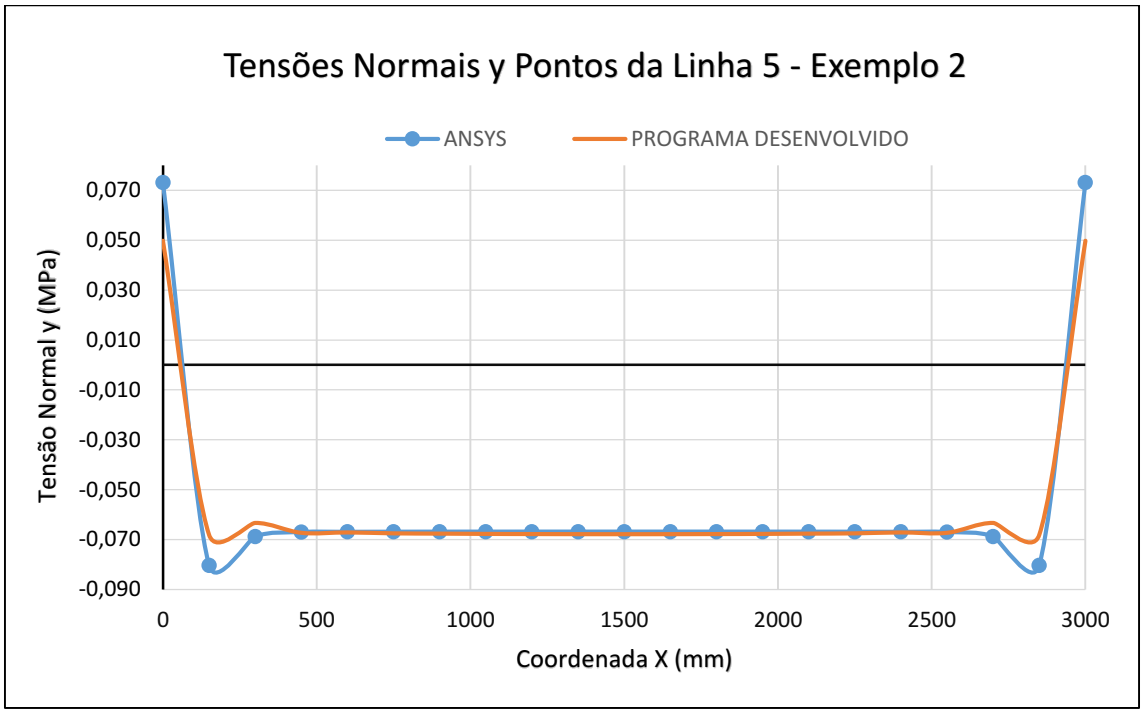
Fonte: Autoria Própria.

Gráfico 10 - Comparação de tensões normais y dos pontos da linha 3 – Exemplo 2.



Fonte: Autoria Própria.

Gráfico 11 - Comparação de tensões normais y dos pontos da linha 5 – Exemplo 2.

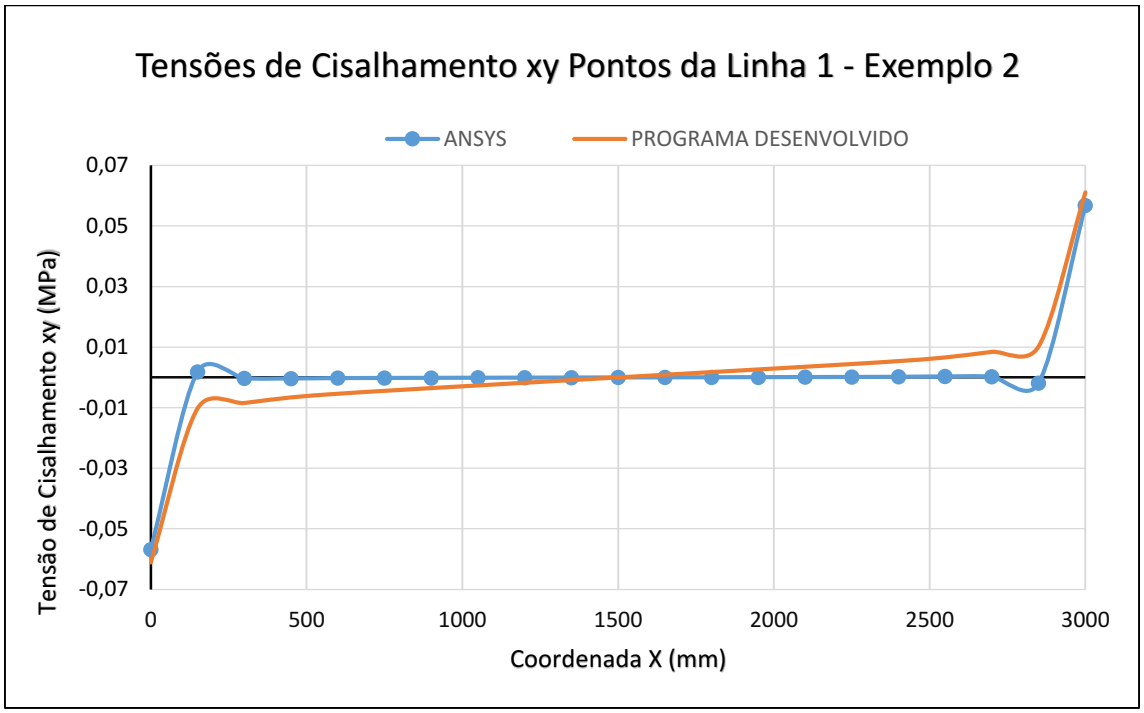


Fonte: Autoria Própria.

Os Gráfico 9, Gráfico 10 e Gráfico 11 mostram que na maioria dos pontos o programa desenvolvido consegue apresentar resultados bastante próximos dos resultados apresentados pelo ANSYS. Apesar de que em alguns pontos, principalmente naqueles situados próximos ao engaste, os resultados foram um pouco divergentes. De maneira geral, essas divergências podem ter ocorrido devido ao refinamento não adequado da malha de elementos finitos nessas regiões.

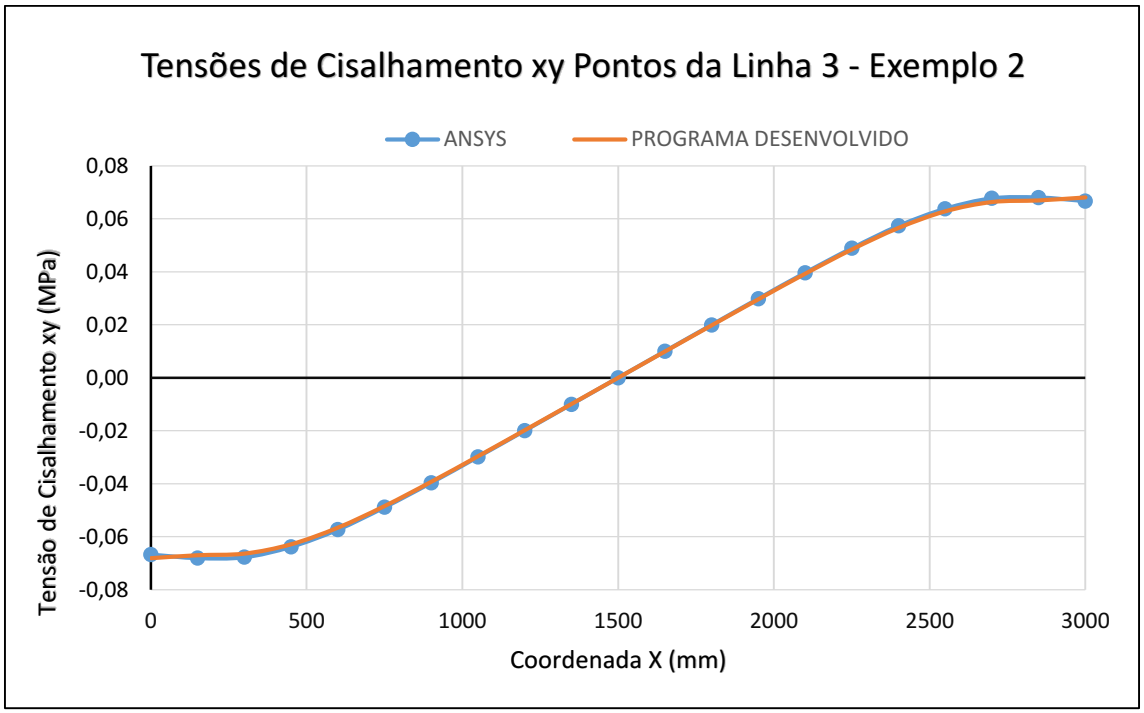
Os resultados apresentados por ambos os programas também foram comparados em termos das tensões de cisalhamento no plano *xy*. Os Gráfico 12, Gráfico 13 e Gráfico 14 ilustram a comparação entre os resultados encontrados para os pontos das linhas 1, 3 e 5, respectivamente.

Gráfico 12 - Comparação de tensões de cisalhamento xy dos pontos da linha 1 – Exemplo 2.



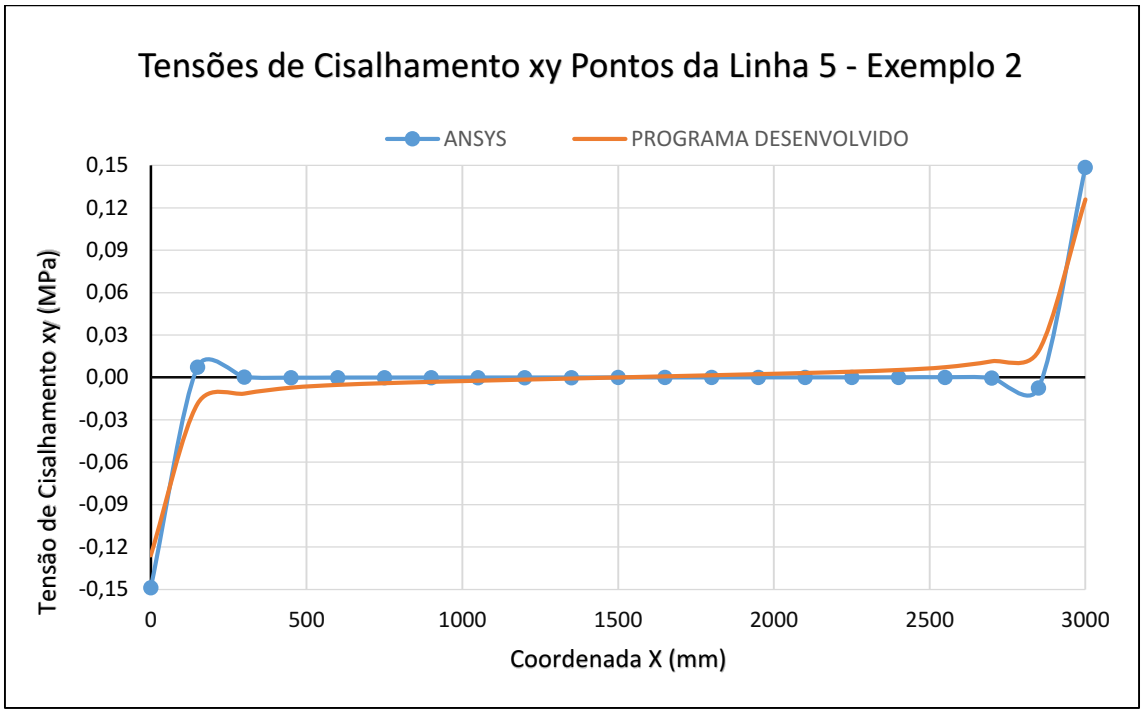
Fonte: Autoria Própria.

Gráfico 13 - Comparação de tensões de cisalhamento xy dos pontos da linha 3 – Exemplo 2.



Fonte: Autoria Própria.

Gráfico 14 - Comparação de tensões de cisalhamento xy dos pontos da linha 5 – Exemplo 2.



Fonte: Autoria Própria.

Os Gráfico 12, Gráfico 13 e Gráfico 14 mostram que ambos os programas indicaram o mesmo comportamento para as tensões cisalhantes. Além disso, os valores de tensões cisalhantes são simétricos em relação ao eixo de simetria vertical da chapa, ocorrendo apenas uma inversão de sinais devido ao sentido da força cortante.

Em relação aos valores, na maioria dos pontos os resultados fornecidos por ambos os programas foram próximos. Em alguns pontos, principalmente naqueles situados mais próximos aos engastes, houveram algumas divergências. De maneira geral nestes pontos, o programa *ANSYS* indicou tensões cisalhantes de intensidades menores e mais próximas de zero do que o programa desenvolvido neste trabalho.

Por fim, ao se analisar os resultados indicados anteriormente e tendo em vista o caráter didático que o programa desenvolvido se propôs inicialmente, se pode dizer que os resultados são satisfatórios para a análise estrutural deste exemplo de chapa biengastada.

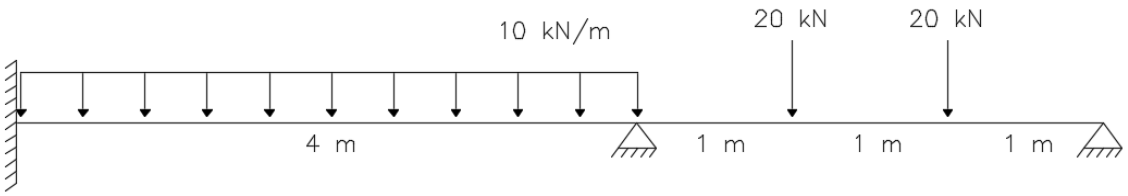
Os resultados obtidos no programa para os deslocamentos nodais da estrutura foram muito bons e ficaram muito próximos dos encontrados no *ANSYS*. Para as tensões, ambos os programas indicaram o mesmo comportamento, havendo divergência nos valores de alguns pontos, mas que podem ser tomadas como divergências normais, principalmente devido a diferença no refinamento da malha de elementos utilizada pelos programas.

4.3 Exemplo 3

O terceiro exemplo trata-se de uma aplicação do elemento de chapa em problemas envolvendo vigas contínuas. A intenção é mostrar que o elemento *CSQ* pode ser utilizado para além da análise de chapas, realizar também a análise estrutural de vigas hiperestáticas submetidas as mais diversas condições de vinculação e carregamentos.

O exemplo trata de uma viga engastada em uma de suas extremidades e composta por dois vãos apoiados, como ilustra a Figura 30. A viga possui seção transversal retangular 150x400 mm. O material que compõe a viga possui módulo de elasticidade $E = 25000$ MPa e coeficiente de Poisson $\upsilon = 0{,}2$.

Figura 30 – Exemplo 3: Viga contínua engastada-apoiada.



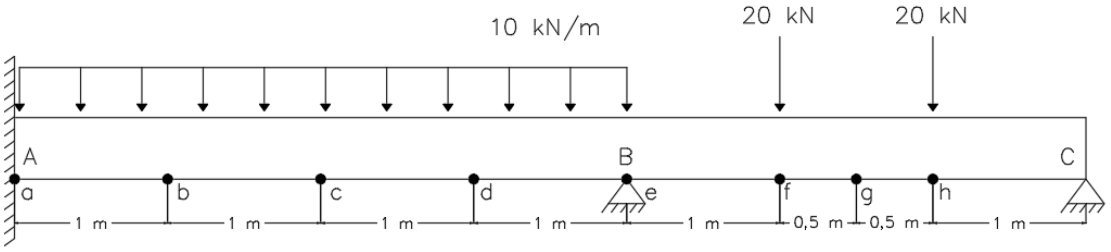
Fonte: Autoria Própria.

Para realizar a validação do programa desenvolvido os resultados foram comparados com os fornecidos pelos softwares *SCIA Engineer* e *Ftool*, que são softwares amplamente utilizados e renomados na análise de estruturas. Para realizar a comparação o exemplo foi analisado no programa desenvolvido utilizando uma malha 71x6 nós, totalizando 350 elementos finitos *CSQ*.

O software *SCIA Engineer* trabalha com elementos de pórtico tridimensionais, isto é, um elemento com 2 nós e 6 graus de liberdade em cada nó. Portanto, na análise do exemplo os elementos finitos foram limitados à dimensão de 100 mm, totalizando 70 elementos finitos. Por outro lado, o *Ftool* trabalha pelo do Método de Análise Matricial, portanto não há a utilização de elementos finitos no cálculo da estrutura.

Para a apresentação dos resultados foram escolhidos alguns pontos específicos da viga. A Figura 31 ilustra a posição e a nomenclatura dos pontos escolhidos, bem como a nomenclatura dos apoios.

Figura 31 – Exemplo 3: Pontos de comparação dos resultados.



Fonte: Autoria Própria.

Os resultados obtidos para as reações de apoio verticais pelo programa desenvolvido foram comparados com os resultados fornecidos pelos softwares *Ftool* e *SCIA Engineer*. A Tabela 8 mostra as comparações entre os resultados, bem como o erro relativo à cada uma delas.

Tabela 8 - Comparação de reações de apoio verticais Programa Desenvolvido x Ftool x SCIA Engineer – Exemplo 3.

Reações de Apoio Verticais (kN)			
Quantidade de Elementos	**Programa Desenvolvido (350 elementos CSQ) Ftool (indefinido) SCIA Engineer (70 elementos de pórtico tridimensionais)**		
Apoio	**A**	**B**	**C**
Programa Desenvolvido	18,32	46,67	14,50
Ftool	18,75	46,80	14,44
SCIA	18,81	46,69	14,49
Erro Relativo Ftool (%)	2,29	0,28	0,42
Erro Relativo SCIA (%)	2,60	0,04	0,07

Fonte: Autoria Própria.

Pela Tabela 8 é possível notar que o programa desenvolvido consegue fornecer resultados satisfatórios em termos das reações de apoio. Nos apoios articulados os erros relativos ficaram com ordem de grandeza menor que 1%, enquanto que no engaste os erros relativos ficaram em torno de 2%. De qualquer forma, as diferenças ocorridas não influem de maneira significativa no resultado obtido.

Os resultados obtidos para os deslocamentos nodais pelo programa desenvolvido foram comparados apenas com os resultados fornecidos pelo software *SCIA Engineer*. A Tabela 9 mostra as comparações entre os resultados obtidos para alguns pontos ao longo da viga, bem como os erros relativos à cada uma delas.

Tabela 9 - Comparação de deslocamentos verticais Programa Desenvolvido x SCIA Engineer – Exemplo 3.

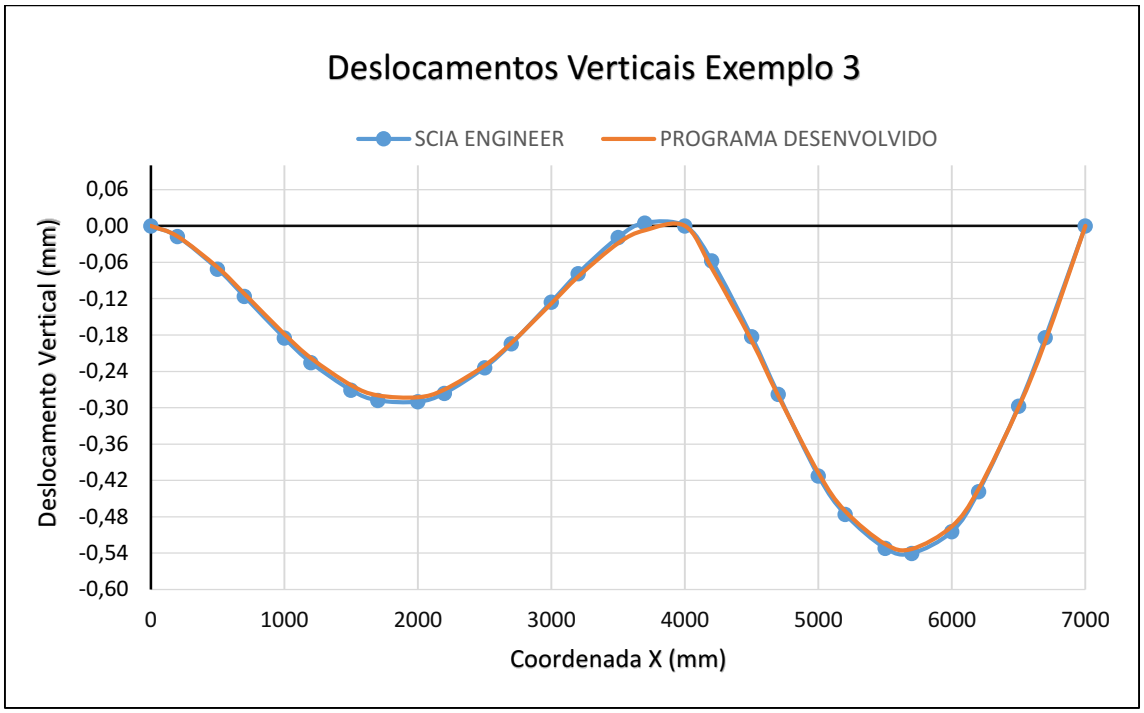
Deslocamentos Verticais (mm)				
Quantidade de Elementos	**Programa Desenvolvido (350 elementos CSQ) SCIA Engineer (70 elementos de pórtico tridimensionais)**			
Ponto	**a**	**b**	**c**	**d**
Programa Desenvolvido	0,00000000	-0,17848408	-0,28283670	-0,12818138
SCIA	0,00000000	-0,18484400	-0,29016001	-0,12539500
Erro Relativo (%)	0,00	3,44	2,52	2,22
Ponto	**e**	**f**	**g**	**h**
Programa Desenvolvido	0,00000000	-0,40756891	-0,52503222	-0,49579756
SCIA	0,00000000	-0,41263000	-0,53189101	-0,50444499
Erro Relativo (%)	0,00	1,23	1,29	1,71

Fonte: Autoria Própria.

A Tabela 9 mostra que o programa desenvolvido consegue fornecer resultados próximos aos fornecidos pelo *SCIA*. Em todos os pontos os erros relativos ficaram com ordens de grandeza pequenas, apresentando valores máximos na ordem de 3%.

Além disso, através do Gráfico 15 é possível analisar o comportamento dos resultados de deslocamentos verticais ao longo de toda a viga. Através dele nota-se que a curva dos resultados obtidos pelo programa desenvolvido fica muito próxima da curva obtida pelo *SCIA Engineer*, comprovando a eficiência do programa desenvolvido em termos dos deslocamentos.

Gráfico 15 - Comparação de deslocamentos verticais – Exemplo 3.



Fonte: Autoria Própria.

Os resultados obtidos para as tensões normais na direção x no programa desenvolvido também foram comparados com os resultados fornecidos pelo *SCIA*. A Tabela 10 mostra as comparações, bem como os erros relativos à cada uma delas.

Tabela 10 - Comparação de tensões normais no eixo x Programa Desenvolvido x SCIA – Exemplo 3.

Tensões Normais x (MPa)				
Quantidade de Elementos	**Programa Desenvolvido (350 elementos CSQ) SCIA Engineer (70 elementos de pórtico tridimensionais)**			
Ponto	**a**	**b**	**c**	**d**
Programa Desenvolvido	-2,92247193	0,50169413	1,43992853	-0,07473183
SCIA	-2,94270996	0,51004526	1,46280078	-0,08444367
Erro Relativo (%)	-0,69	-1,64	-1,56	-11,50
Ponto	**e**	**f**	**g**	**h**
Programa Desenvolvido	-4,95153842	2,05398918	2,88827788	3,40272286
SCIA	-4,13168750	2,24554150	2,93415503	3,62277002
Erro Relativo (%)	19,84	-8,53	-1,56	-6,07

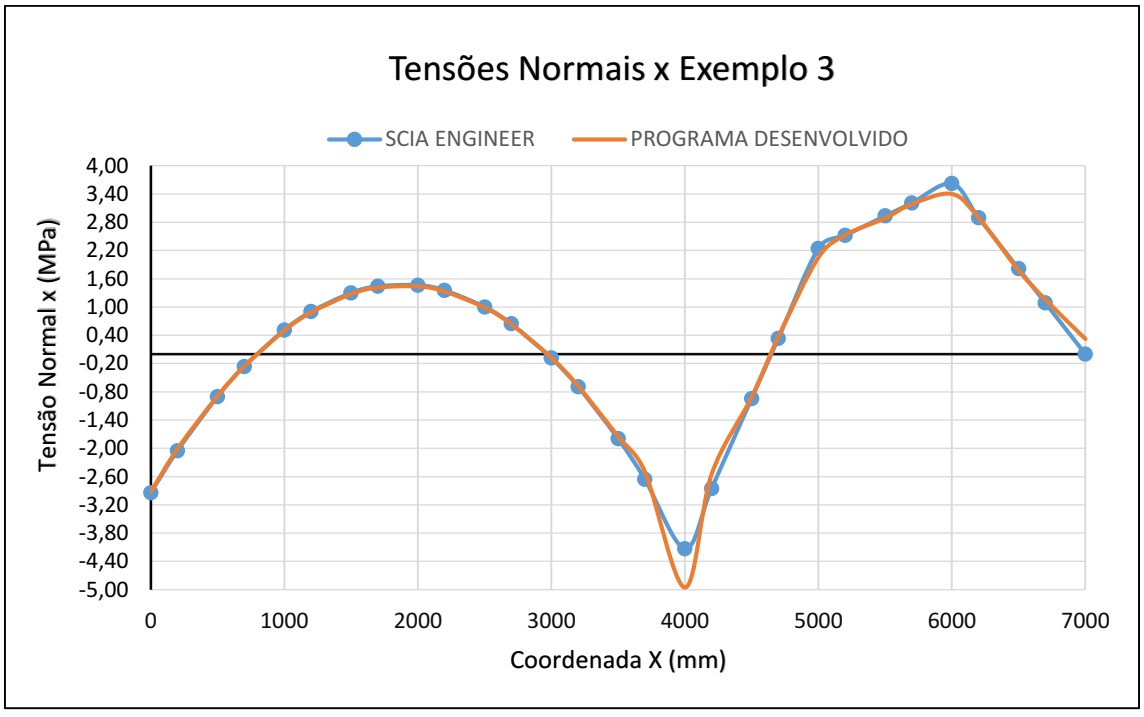
Fonte: Autoria Própria.

Através da Tabela 10 percebe-se que na maioria dos pontos os erros relativos ficaram com ordem de grandeza baixa, exceto nos pontos próximos ao apoio central e nos pontos de aplicação das forças concentradas. Essas divergências provavelmente ocorreram devido aos diferentes tipos de malha e de elementos finitos utilizados em cada um dos programas.

Além disso, nas regiões onde ocorrem picos de tensão é sempre necessário um maior refinamento da malha. Mesmo assim, os resultados podem ser considerados aceitáveis tendo em vista que com uma melhora no refinamento eles tenderiam a se aproximar.

Por fim, o Gráfico 16 permite visualizar o comportamento das tensões normais na direção x ao longo de toda a viga.

Gráfico 16 - Comparação de tensões normais x – Exemplo 3.



Fonte: Autoria Própria.

Através do Gráfico 16 é possível visualizar que a curva de resultados obtidas pelo programa desenvolvido fica muito próxima da curva de resultados fornecidos pelo *SCIA*.

Percebe-se que os resultados mostram que o programa desenvolvido consegue realizar de maneira satisfatória aquilo que lhe foi proposto, tanto em termos de reações de apoio e deslocamentos, quanto em termos de tensões normais.

Na grande maioria dos pontos analisados os erros relativos ficaram com ordens de grandeza baixas. Nos pontos onde houveram divergências maiores, estas podem ser explicadas principalmente pelo uso de diferentes malhas de elementos, uma vez que o programa desenvolvido trabalha com elementos bidimensionais e o *SCIA* trabalha com elementos tridimensionais. De toda forma, os resultados podem ser considerados aceitáveis e mostram que o programa desenvolvido consegue utilizar a formulação do Método dos Elementos Finitos baseada no uso de elementos retangulares *CSQ* para realizar a análise estrutural de vigas contínuas.

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo desenvolver um código computacional de caráter didático que realizasse a análise elástica linear de chapas via método dos elementos finitos, utilizando para isso elementos bidimensionais do tipo *CSQ*. Dessa forma, foi necessário realizar uma revisão bibliográfica que fosse capaz de abordar os conceitos básicos do método e servir de suporte para a formulação das equações matemáticas.

Assim sendo, a partir dos conceitos de estado plano de tensão e da Lei de Hooke generalizada se obteve as relações entre deslocamentos, deformações e tensões. Foram introduzidas as propriedades mecânicas dos materiais, tais como coeficiente de Poisson, módulo de Young e módulo de elasticidade transversal.

Então, se mostrou como o MEF consegue realizar a análise estrutural de um sistema físico transformando um modelo contínuo, isto é, regido por equações diferenciais parciais, em um sistema discreto formado por elementos finitos. Após a discretização, o MEF consegue aproximar os deslocamentos do elemento em função dos seus deslocamentos nodais, a partir de funções de aproximação. Uma vez determinados os deslocamentos nodais é possível determinar as deformações normais e angulares, bem como as tensões normais e cisalhantes que atuam na estrutura.

A partir então do Princípio dos Trabalhos Virtuais (PTV) e de todos os conceitos estabelecidos, foi possível realizar a dedução da matriz de rigidez do elemento *CSQ*. Essa matriz rege a equação fundamental do MEF e faz a relação entre forças e deslocamentos nodais. Após a dedução da matriz de rigidez, a mesma foi implementada no código computacional, assim como os demais conceitos inerentes à análise.

O código computacional foi desenvolvido em linguagem Python, sendo dividido em sub-rotinas de modo a facilitar a visualização e o entendimento do mesmo.

Para realizar a validação do código foram executados alguns exemplos e os resultados foram comparados com a literatura ou com outros softwares. De forma geral, em todos os exemplos o código conseguiu obter resultados satisfatórios tanto em termos de deslocamentos e reações de apoio, quanto em termos de deformações ou tensões.

Através das tabelas apresentadas foi possível visualizar numericamente a comparação entre os resultados obtidos e então concluir que na maioria dos casos, principalmente em termos dos deslocamentos nodais, os resultados ficaram bem

próximos. Nos pontos em que houve divergências, estas podem ser atribuídas as diferentes malhas utilizadas, aos diferentes tipos de elementos finitos e até mesmo aos erros numéricos que são inerentes ao processo de cálculo computacional.

Além disso, através dos gráficos apresentados foi possível visualizar que em todos os casos o programa conseguiu estabelecer o mesmo comportamento estrutural que os demais softwares, seja em termos de deslocamentos nodais, seja em termos de tensões atuantes.

Por fim, as diferentes malhas utilizadas para realizar a análise dos exemplos comprovaram que a medida que se aumenta o refinamento se obtém uma melhora nos resultados, até que a partir de uma determinada quantidade de elementos esse refinamento, e por consequência aumento no esforço computacional, não se justificam mais pois o ganho de aproximação é mínimo.

Todos esses resultados demonstraram que o código consegue estabelecer de maneira satisfatória aquilo que foi proposto inicialmente. Pode-se então concluir que o código pode ser usado para realizar a análise elástica linear de estruturas de chapas, fornecendo resultados confiáveis e adequados.

Por último, se faz algumas sugestões para trabalhos futuros. Primeiramente, pode-se realizar a formulação matemática do método para elementos finitos retangulares com 8 pontos nodais, ou seja, com nós intermediários nas faces do elemento. Pode-se também otimizar o código, fazendo com que o mesmo consiga trabalhar com diferentes tamanhos de elementos finitos, permitindo a realização de refinamento de malha apenas em regiões críticas, tais como regiões de concentração de tensões.

Além disso, sugere-se trabalhar com estruturas de chapas mais diversificadas, tais como chapas em outros formatos geométricos, ou então chapas com variações de material e de seção transversal. Para trabalhos mais complexos, sugere-se também trabalhar com análises que contemplem a não-linearidade física e ainda a análise dinâmica de chapas.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6118: Projeto de estruturas de concreto - procedimento.** 3. Ed. Rio de Janeiro, 2014.

AZEVEDO, Álvaro F. M. **Método dos Elementos Finitos.** 1. ed. Porto: Faculdade de Engenharia da Universidade do Porto, 2003.

BORDA, Ricardo A. A. **Análise experimental de vigas parede de concreto reforçado com bambu.** 2013. 123 f. Dissertação (Mestrado em Engenharia Civil) – Departamento de Engenharia Civil, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2013. Disponível em: <http://www.civ.puc-rio.br/wp-content/view/down_pdf.php?pdf=../pdf/2013_Ricardo_Arthuro_Amado_Borda.pdf>. Acesso em: 05 dez. 2018.

CAMPOS FILHO, Américo. **Detalhamento das estruturas de concreto pelo método das bielas e dos tirantes.** 1996. 27 f. Caderno de Engenharia – Programa de Pós-Graduação em Engenharia Civil, Universidade Federal do Rio Grande do Sul – UFRGS, Porto Alegre, 1996. Disponível em: < https://chasqueweb.ufrgs.br/~americo/topicos/bielas.pdf>. Acesso em: 26 fev. 2019.

D'AVILA, Virgínia M. R. **Estudo sobre modelos de fissuração de peças de concreto armado via método dos elementos finitos.** 2003. 287 f. Tese (Doutorado em Engenharia Civil) - Programa de Pós-Graduação em Engenharia Civil, Universidade Federal do Rio Grande do Sul - UFRGS, Porto Alegre, 2003. Disponível em: <https://lume.ufrgs.br/handle/10183/1685>. Acesso em: 14 jan. 2019.

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

FRANCO, Marina I. E. **Vigas-parede: comparação entre diferentes metodologias de cálculo.** 2015. 131 f. Trabalho de Conclusão de Curso – Departamento de Engenharia Civil, Universidade Federal do Rio Grande do Sul – UFRGS, Porto Alegre, 2015. Disponível em: <https://lume.ufrgs.br/handle/10183/138304>. Acesso em: 05 dez. 2018.

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 - UFU, Uberlândia, 2010.

HIBBELER, Russel C. **Resistência dos Materiais.** 7. ed. São Paulo: Pearson Prentice Hall, 2010.

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

MARTHA, Luiz F**. Análise de Estruturas: Conceitos e métodos básicos.** 1. ed. Rio de Janeiro: Elsevier, 2010.

PEREIRA, Orlando J.B.A. **Introdução ao Método dos Elementos Finitos na Análise de Problemas Planos de Elasticidade.** [S.l.: s.n.], 2005. Disponível em: <http://www.civil.ist.utl.pt/~orlando/ae2/IMEFAPPE.pdf>. Acesso em: 05 dez. 2018.

RIBEIRO, Fernando L. B. **Introdução ao Método dos Elementos Finitos**. 2004. 93f. Notas de aula - Universidade Federal do Rio de Janeiro - UFRJ, Rio de Janeiro, 2004.

SALES, José J. et al. **Sistemas Estruturais: Teorias e exemplos.** 1. ed. São Carlos: Escola de Engenharia de São Carlos (EESC), 2005.

SANTOS, Gláucia G. M. **Análise sistemática de vigas-parede biapoiadas de concreto armado.** 1999. 175 f. Dissertação (Mestrado em Engenharia de Estruturas) – Departamento de Engenharia Civil, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 1999. Disponível em: < http://wwwp.feb.unesp.br/pbastos/concreto3/Diss-V.Parede.pdf>. Acesso em: 05 dez. 2018.

SAVASSI, Walter. **Introdução ao Método dos Elementos Finitos em Análise Linear de Estruturas.** 1. ed. São Carlos: Escola de Engenharia de São Carlos (EESC), 1996.

SORIANO, Humberto L. **Elementos Finitos: Formulação e aplicação na estática e dinâmica das estruturas.** 1. ed. Rio de Janeiro: Ciência Moderna, 2009.

SUSSEKIND, José C. **Curso de Análise Estrutural: Estuturas Isostáticas.** 6. ed. Porto Alegre-Rio de Janeiro: Globo, 1981.

VAZ, Luiz E. **Método dos Elementos Finitos em Análise de Estruturas.** 1. ed. Rio de Janeiro: Elsevier, 2011.

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

WAIDEMAM, Leandro**. Formulação do método dos elementos de contorno para placas enrijecidas considerando-se não-linearidades física e geométrica.** 2008. 240 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 DESENVOLVIDO

```
import numpy as np
import math
def analise_chapas():

#Sub-Rotina 1: Dados Iniciais

    #o usuário informa os principais dados do problema
    print('1) Dados Iniciais')
    print('')
    nome=input('Informe o nome do arquivo: ')
    un_forca=input('Informe a unidade de força adotada: ')
    un_metrica=input('Informe a unidade métrica adotada: ')
    comp=float(input('Informe o comprimento da chapa: '))
    alt=float(input('Informe a altura da chapa: '))
    esp=float(input('Informe a espessura da chapa: '))
    mod=float(input('Informe o módulo de elasticidade do material: '))
    poi=float(input('Informe o coeficiente de poisson do material: '))
    num_no_h=int(input('Informe o número de nós desejados para a
malha na direção horizontal: '))
    num_no_v=int(input('Informe o número de nós desejados para a
malha na direção vertical: '))
    print('')
    num_no=num_no_h*num_no_v
    print('A quantidade de nós escolhida para a malha é: ', num_no)
    num_ele_h=(num_no_h-1)
    num_ele_v=(num_no_v-1)
    num_ele=(num_ele_h*num_ele_v)
    print('A quantidade de elementos finitos é: ', num_ele)
    print('Aviso: caso necessite de um maior refinamento, reinicie
o programa e aumente a quantidade de nós escolhida.')
    print('')
    if (num_no_h==2):
        a=comp/2
    else:
        a=(comp/(num_ele_h))/2
    if (num_no_v==2):
        b=alt/2
    else:
        b=(alt/(num_ele_v))/2
    print('Dimensões do elemento finito CSQ utilizado:')
    print('Dimensão Horizontal= ', 2*a)
    print('Dimensão Vertical= ', 2*b)
    print('')

    #o programa cria uma lista contendo todos os nós em numeração global
    i=1
    list_nos=[]
    while (i<=num_no):
        no=i
        list_nos.append(no)
        i+=1

    #o programa cria sub-listas, onde cada uma representa uma
linha horizontal da malha
    v1=1
    h1=1
```

```
    h2=num_no_h
    list_lin_nos=[]
    while (v1<=num_no_v):
        linha=[]
        for i in range (h1, h2+1):
            no=list_nos[i-1]
            linha.append(no)
        h1=h1+num_no_h
        h2=h2+num_no_h
        list_lin_nos.append(linha)
        v1+=1

    #o programa imprime na tela a distribuição dos nós de acordo
com a malha escolhida
    print('Distribuição dos nós na malha:')
    linha=len(list_lin_nos)
    for i in range (0, len(list_lin_nos)):
        print(list_lin_nos[linha-1])
        linha-=1
    print('')

#Sub-Rotina 2: Definição das Condições de Vinculação

    print('2) Condições de Vinculação')
    print('')

    #o programa pergunta ao usuário quais nós possuem deslocamento
restrito na direção horizontal (direção x) e armazena isso em uma
lista
    list_ap_x=[]
    c='inicio'
    while (c!='fim'):
        no=int(input('Informe o número do nó com deslocamento
restrito em x ou 0 para finalizar e prosseguir: '))
        if (no!=0):
            list_ap_x.append(no)
        else:
            c='fim'
        print('')

    #o programa pergunta ao usuário se ele quer copiar as mesmas
informações de vinculação na direção horizontal para a direção
vertical
    list_ap_y=[]
    resposta=str(input('Gostaria de copiar as condições de
restrição na direção horizontal para a direção vertical? Digite
sim ou nao: '))
    print('')

    #caso as informações não sejam copiadas, o programa pergunta
ao usuário quais nós possuem deslocamento restrito na direção
vertical (direção y) e armazena isso em uma lista
    if (resposta=='sim'):
        list_ap_y=list_ap_x
    else:
        c='inicio'
```

```
        while (c!='fim'):
            no=int(input('Informe o número do nó com deslocamento
restrito em y ou 0 para finalizar e prosseguir: '))
            if (no!=0):
                list_ap_y.append(no)
            else:
                c='fim'
            print('')
    print ('Nós com deslocamento restrito em x: ', list_ap_x)
    print ('Nós com deslocamento restrito em y: ', list_ap_y)
    print('')

    #o programa cria listas que armazenam as informações de
restrição, sendo que 0 indica deslocamento livre e 1 indica
deslocamento restrito
    list_restr_x=[]
    list_restr_y=[]
    for i in range (0, len(list_nos)):
        termo=0
        list_restr_x.append(termo)
        list_restr_y.append(termo)
    for i in range (0, len(list_ap_x)):
        termo=list_ap_x[i]
        list_restr_x[termo-1]=1
    for i in range (0, len(list_ap_y)):
        termo=list_ap_y[i]
        list_restr_y[termo-1]=1

#Sub-Rotina 3: Definição das Condições de Carregamento

    print('3) Condições de Carregamento')
    print('')

    #o programa cria listas para armazenar os valores das forças
aplicadas em cada um dos nós na direção x e y
    list_forc_x=[]
    list_forc_y=[]
    for i in range (0, len(list_nos)):
        termo=0.0
        list_forc_x.append(termo)
        list_forc_y.append(termo)

    #o usuário informa as forças concentradas aplicadas na estrutura
    c='inicio'
    while (c!='fim'):
        no=int(input('Informe o nó em que a força concentrada atua
ou 0 para finalizar e prosseguir: '))
        if (no!=0):
            fx=float(input('Informe o valor da força na direçao x
atuante neste nó: '))
            list_forc_x[no-1]=fx
            fy=float(input('Informe o valor da força na direçao y
atuante neste nó: '))
            list_forc_y[no-1]=fy
        else:
            c='fim'
```

```
        print('')

    #o usuário informa as forças distribuidas aplicadas na estrutura
    c='inicio'
    while (c!='fim'):
        print('')
        print('Informe a face em que a força distribuída atua: ')
        print('Digite 1 para face inferior')
        print('Digite 2 para face lateral direita')
        print('Digite 3 para face superior')
        print('Digite 4 para face lateral esquerda')
        print('Digite 5 para opção personalizada')
        print('')
        face=int(input('Informe a opção desejada ou 0 para
finalizar e prosseguir: '))
        if (face==1):
            print('')
            print('Face Inferior')
            Fx=float(input('Informe o valor da força distribuida
na direção x atuante nesta face: '))
            Fy=float(input('Informe o valor da força distribuida
na direção y atuante nesta face: '))
            no=1
            while (no<=num_no_h):
                if (no==1 or no==num_no_h):
                    fx=Fx*a
                    fy=Fy*a
                    list_forc_x[no-1]=list_forc_x[no-1]+fx
                    list_forc_y[no-1]=list_forc_y[no-1]+fy
                else:
                    fx=Fx*2*a
                    fy=Fy*2*a
                    list_forc_x[no-1]=list_forc_x[no-1]+fx
                    list_forc_y[no-1]=list_forc_y[no-1]+fy
                no+=1
        if (face==2):
            print('')
            print('Face Lateral Direita')
            Fx=float(input('Informe o valor da força distribuida
na direção x atuante nesta face: '))
            Fy=float(input('Informe o valor da força distribuida
na direção y atuante nesta face: '))
            no=num_no_h
            while (no<=num_no):
                if (no==num_no_h or no==num_no):
                    fx=Fx*b
                    fy=Fy*b
                    list_forc_x[no-1]=list_forc_x[no-1]+fx
                    list_forc_y[no-1]=list_forc_y[no-1]+fy
                else:
                    fx=Fx*2*b
                    fy=Fy*2*b
                    list_forc_x[no-1]=list_forc_x[no-1]+fx
                    list_forc_y[no-1]=list_forc_y[no-1]+fy
                no=no+num_no_h
        if (face==3):
```

```
            print('')
            print('Face Superior')
            Fx=float(input('Informe o valor da força distribuida
na direção x atuante nesta face: '))
            Fy=float(input('Informe o valor da força distribuida
na direção y atuante nesta face: '))
            no=(num_no-num_no_h)+1
            while (no<=num_no):
                if (no==(num_no-num_no_h)+1 or no==num_no):
                    fx=Fx*a
                    fy=Fy*a
                    list_forc_x[no-1]=list_forc_x[no-1]+fx
                    list_forc_y[no-1]=list_forc_y[no-1]+fy
                else:
                    fx=Fx*2*a
                    fy=Fy*2*a
                    list_forc_x[no-1]=list_forc_x[no-1]+fx
                    list_forc_y[no-1]=list_forc_y[no-1]+fy
                no+=1
        if (face==4):
            print('')
            print('Face Lateral Esquerda')
            Fx=float(input('Informe o valor da força distribuida
na direção x atuante nesta face: '))
            Fy=float(input('Informe o valor da força distribuida
na direção y atuante nesta face: '))
            no=1
            while (no<=((num_no-num_no_h)+1)):
                if (no==1 or no==(num_no-num_no_h)+1):
                    fx=Fx*b
                    fy=Fy*b
                    list_forc_x[no-1]=list_forc_x[no-1]+fx
                    list_forc_y[no-1]=list_forc_y[no-1]+fy
                else:
                    fx=Fx*2*b
                    fy=Fy*2*b
                    list_forc_x[no-1]=list_forc_x[no-1]+fx
                    list_forc_y[no-1]=list_forc_y[no-1]+fy
                no=no+num_no_h
        if (face==5):
            print('')
            print('Carga Distribuída Opção Personalizada')
            print('Digite 1 para aplicar força distribuída em
superfície horizontal da chapa')
            print('Digite 2 para aplicar força distribuída em
superfície vertical da chapa')
            resposta=int(input('Informe a opção desejada: '))
            print('')
            if (resposta==1):
                no_inicial=int(input('Informe o nó inicial do
carregamento: '))
                no_final=int(input('Informe o nó final do
carregamento: '))
                Fx=float(input('Informe o valor da força
distribuida na direção x: '))
```

```
                Fy=float(input('Informe o valor da força
distribuida na direção y: '))
                no=no_inicial
                while (no<=no_final):
                    if (no==no_inicial or no==no_final):
                        fx=Fx*a
                        fy=Fy*a
                        list_forc_x[no-1]=list_forc_x[no-1]+fx
                        list_forc_y[no-1]=list_forc_y[no-1]+fy
                    else:
                        fx=Fx*2*a
                        fy=Fy*2*a
                        list_forc_x[no-1]=list_forc_x[no-1]+fx
                        list_forc_y[no-1]=list_forc_y[no-1]+fy
                    no+=1
            if (resposta==2):
                no_inicial=int(input('Informe o nó inicial do
carregamento: '))
                no_final=int(input('Informe o nó final do
carregamento: '))
                Fx=float(input('Informe o valor da força
distribuida na direção x: '))
                Fy=float(input('Informe o valor da força
distribuida na direção y: '))
                no=no_inicial
                while (no<=no_final):
                    if (no==no_inicial or no==no_final):
                        fx=Fx*b
                        fy=Fy*b
                        list_forc_x[no-1]=list_forc_x[no-1]+fx
                        list_forc_y[no-1]=list_forc_y[no-1]+fy
                    else:
                        fx=Fx*2*b
                        fy=Fy*2*b
                        list_forc_x[no-1]=list_forc_x[no-1]+fx
                        list_forc_y[no-1]=list_forc_y[no-1]+fy
                    no=no+num_no_h
        elif (face==0):
            c='fim'
    print('')

#Sub-Rotina 4: Montagem da Malha de Elementos Finitos

    #o programa cria uma lista com os nós iniciais de cada elemento
    v1=1
    v2=num_no_v-1
    h1=1
    h2=num_no_h-1
    list_nos_in=[]
    for i in range (v1, v2+1):
        for j in range (h1, h2+1):
            no=list_lin_nos[i-1][j-1]
            list_nos_in.append(no)

    #o programa cria uma lista com os nós secundários de cada elemento
    v1=1
```

```
    v2=num_no_v-1
    h1=2
    h2=num_no_h
    list_nos_sec=[]
    for i in range (v1, v2+1):
        for j in range (h1, h2+1):
            no=list_lin_nos[i-1][j-1]
            list_nos_sec.append(no)

    #o programa cria uma lista com os nós terciários de cada elemento
    v1=2
    v2=num_no_v
    h1=2
    h2=num_no_h
    list_nos_terc=[]
    for i in range (v1, v2+1):
        for j in range (h1, h2+1):
            no=list_lin_nos[i-1][j-1]
            list_nos_terc.append(no)

    #o programa cria uma lista com os nós finais de cada elemento
    v1=2
    v2=num_no_v
    h1=1
    h2=num_no_h-1
    list_nos_fi=[]
    for i in range (v1, v2+1):
        for j in range (h1, h2+1):
            no=list_lin_nos[i-1][j-1]
            list_nos_fi.append(no)

    #o programa cria uma lista com sub-listas, em que cada uma
representa um elemento e os nós pertencentes à ele
    list_ele=[]
    for i in range (0, num_ele):
        no_in=list_nos_in[i]
        no_sec=list_nos_sec[i]
        no_terc=list_nos_terc[i]
        no_fi=list_nos_fi[i]
        ele=[no_in, no_sec, no_terc, no_fi]
        list_ele.append(ele)

#Sub-Rotina 5: Montagem da Matriz de Rigidez Elementar

    #o programa recebe e armazena em uma lista os termos da matriz
de rigidez deduzida no trabalho para o elemento CSQ
    c=(esp*mod)/(1-(poi**2))
    list_termos=[]
    k11=c*((2*(b**2)+(1-poi)*(a**2))/(6*a*b))
    list_termos.append(k11)
    k12=c*((1+poi)/(8))
    list_termos.append(k12)
    k13=c*(((-4)*(b**2)+(1-poi)*(a**2))/(12*a*b))
    list_termos.append(k13)
    k14=c*((-1+3*poi)/(8))
    list_termos.append(k14)
```

```
k15=c*(((-2)*(b**2)+(-1+poi)*(a**2))/(12*a*b))
list_termos.append(k15)
k16=c*((-1-poi)/(8))
list_termos.append(k16)
k17=c*(((b**2)+(-1+poi)*(a**2))/(6*a*b))
list_termos.append(k17)
k18=c*((1-3*poi)/(8))
list_termos.append(k18)
k21=k12
list_termos.append(k21)
k22=c*((2*(a**2)+(1-poi)*(b**2))/(6*a*b))
list_termos.append(k22)
k23=c*((1-3*poi)/(8))
list_termos.append(k23)
k24=c*(((a**2)+(-1+poi)*(b**2))/(6*a*b))
list_termos.append(k24)
k25=c*((-1-poi)/(8))
list_termos.append(k25)
k26=c*(((-2)*(a**2)+(-1+poi)*(b**2))/(12*a*b))
list_termos.append(k26)
k27=c*((-1+3*poi)/(8))
list_termos.append(k27)
k28=c*(((-4)*(a**2)+(1-poi)*(b**2))/(12*a*b))
list_termos.append(k28)
k31=k13
list_termos.append(k31)
k32=k23
list_termos.append(k32)
k33=c*((2*(b**2)+(1-poi)*(a**2))/(6*a*b))
list_termos.append(k33)
k34=c*((-1-poi)/(8))
list_termos.append(k34)
k35=c*(((b**2)+(-1+poi)*(a**2))/(6*a*b))
list_termos.append(k35)
k36=c*((-1+3*poi)/(8))
list_termos.append(k36)
k37=c*(((-2)*(b**2)+(-1+poi)*(a**2))/(12*a*b))
list_termos.append(k37)
k38=c*((1+poi)/(8))
list_termos.append(k38)
k41=k14
list_termos.append(k41)
k42=k24
list_termos.append(k42)
k43=k34
list_termos.append(k43)
k44=c*((2*(a**2)+(1-poi)*(b**2))/(6*a*b))
list_termos.append(k44)
k45=c*((1-3*poi)/(8))
list_termos.append(k45)
k46=c*(((-4)*(a**2)+(1-poi)*(b**2))/(12*a*b))
list_termos.append(k46)
k47=c*((1+poi)/(8))
list_termos.append(k47)
k48=c*(((-2)*(a**2)+(-1+poi)*(b**2))/(12*a*b))
list_termos.append(k48)
```

```
k51=k15
list_termos.append(k51)
k52=k25
list_termos.append(k52)
k53=k35
list_termos.append(k53)
k54=k45
list_termos.append(k54)
k55=c*((2*(b**2)+(1-poi)*(a**2))/(6*a*b))
list_termos.append(k55)
k56=c*((1+poi)/(8))
list_termos.append(k56)
k57=c*(((-4)*(b**2)+(1-poi)*(a**2))/(12*a*b))
list_termos.append(k57)
k58=c*((-1+3*poi)/(8))
list_termos.append(k58)
k61=k16
list_termos.append(k61)
k62=k26
list_termos.append(k62)
k63=k36
list_termos.append(k63)
k64=k46
list_termos.append(k64)
k65=k56
list_termos.append(k65)
k66=c*((2*(a**2)+(1-poi)*(b**2))/(6*a*b))
list_termos.append(k66)
k67=c*((1-3*poi)/(8))
list_termos.append(k67)
k68=c*(((a**2)+(-1+poi)*(b**2))/(6*a*b))
list_termos.append(k68)
k71=k17
list_termos.append(k71)
k72=k27
list_termos.append(k72)
k73=k37
list_termos.append(k73)
k74=k47
list_termos.append(k74)
k75=k57
list_termos.append(k75)
k76=k67
list_termos.append(k76)
k77=c*((2*(b**2)+(1-poi)*(a**2))/(6*a*b))
list_termos.append(k77)
k78=c*((-1-poi)/(8))
list_termos.append(k78)
k81=k18
list_termos.append(k81)
k82=k28
list_termos.append(k82)
k83=k38
list_termos.append(k83)
k84=k48
list_termos.append(k84)
```

```
    k85=k58
    list_termos.append(k85)
    k86=k68
    list_termos.append(k86)
    k87=k78
    list_termos.append(k87)
    k88=c*((2*(a**2)+(1-poi)*(b**2))/(6*a*b))
    list_termos.append(k88)
    o=8
    p=1
    q=8
    list_k_e=[]
    for i in range (1, o+1):
        linha=[]
        for j in range (p, q+1):
            termo=list_termos[j-1]
            linha.append(termo)
        p=p+o
        q=q+o
        list_k_e.append(linha)

    #o programa gera a matriz de rigidez elementar para o elemento
CSQ utilizando a plataforma numpy
    mat_k_e=np.array([list_k_e])

#Sub-Rotina 6: Vetor de Deslocamentos Nodais Para Cada Elemento

    #o programa cria uma lista para armazenar os deslocamentos de
todos os nós para cada um dos elementos, indicando os nós não
pertencentes ao elemento
    ele=1
    list_desl=[]
    while (ele<=num_ele):
        list_desl_x_ele=[]
        list_desl_y_ele=[]
        for i in range (0, len(list_nos)):
            desl='nó não pertencente ao elemento'
            list_desl_x_ele.append(desl)
            list_desl_y_ele.append(desl)
        for i in range (0, 4):
            no=list_ele[ele-1][i]
            if (list_restr_x[no-1]==1): #se tiver deslocamento
restrito substitui 'nó não pertencente' por 0
                list_desl_x_ele[no-1]=0.0
            else:                       #se não tiver deslocamento
restrito substitui 'nó não pertencente' por 'u'
                list_desl_x_ele[no-1]='u'
        for i in range (0, 4):
            no=list_ele[ele-1][i]
            if (list_restr_y[no-1]==1):
                list_desl_y_ele[no-1]=0.0
            else:
                list_desl_y_ele[no-1]='u'
        list_desl_ele=[]
        for i in range (0, len(list_nos)):
            list_desl_ele.append(list_desl_x_ele[i])
```

```
            list_desl_ele.append(list_desl_y_ele[i])
        list_desl.append(list_desl_ele)
        ele+=1

#Sub-Rotina 7: Montagem da Matriz de Rigidez Global

    #o programa troca linhas e colunas da matriz de rigidez
elementar para reagrupar esta de acordo com a numeração global de
nós
    list_k_e_r_c=[]
    for i in range (0, o):
        linha=[0,0,0,0,0,0,0,0]
        linha[0]=list_k_e[i][0]
        linha[1]=list_k_e[i][1]
        linha[2]=list_k_e[i][2]
        linha[3]=list_k_e[i][3]
        linha[4]=list_k_e[i][6]
        linha[5]=list_k_e[i][7]
        linha[6]=list_k_e[i][4]
        linha[7]=list_k_e[i][5]
        list_k_e_r_c.append(linha)
    list_k_e_r=[0,0,0,0,0,0,0,0]
    list_k_e_r[0]=list_k_e_r_c[0]
    list_k_e_r[1]=list_k_e_r_c[1]
    list_k_e_r[2]=list_k_e_r_c[2]
    list_k_e_r[3]=list_k_e_r_c[3]
    list_k_e_r[4]=list_k_e_r_c[6]
    list_k_e_r[5]=list_k_e_r_c[7]
    list_k_e_r[6]=list_k_e_r_c[4]
    list_k_e_r[7]=list_k_e_r_c[5]

    #o programa cria uma lista para armazenar as matrizes de
rigidez em ordem global de cada elemento
    ord_mat=num_no*2
    ele=1
    list_mat_rig=[]
    while (ele<=num_ele):

        #o programa cria uma matriz de mesma ordem da matriz de
rigidez da estrutura
        list_mat_rig_i=[]
        for i in range (0, ord_mat):
            linha_mat=[]
            for j in range (0, ord_mat):
                termo_mat='termo da matriz'
                linha_mat.append(termo_mat)
            list_mat_rig_i.append(linha_mat)

        #o programa pega a lista com os deslocamentos nodais para
o elemento e zera linhas e colunas da matriz referentes aos nós
não pertencentes ao elemento
        list_desl_i=list_desl[ele-1]
        for i in range (0, len(list_desl_i)):
            termo=list_desl_i[i]
            if (termo=='nó não pertencente ao elemento'):
                for l in range (0, ord_mat):
```

```
                    list_mat_rig_i[l][i]=0.0
                for c in range (0, ord_mat):
                    list_mat_rig_i[i][c]=0.0

        #o programa aloca a matriz de rigidez elementar nas posições corretas dentro da matriz de rigidez da estrutura referente ao elemento em questão
        cont_lin=0
        for i in range (0, len(list_desl_i)): #onde não estiver 'nó não pertencente ao elemento' pega os termos da matriz de rigidez elementar
            termo=list_desl_i[i]
            if (termo!='nó não pertencente ao elemento'):
                cont_col=0
                for c in range (0, ord_mat):
                    if (list_mat_rig_i[i][c]=='termo da matriz'):
                        list_mat_rig_i[i][c]=list_k_e_r[cont_lin][cont_col]
                        cont_col+=1
                cont_lin+=1
        list_mat_rig.append(list_mat_rig_i)
        ele+=1

    #o programa cria uma matriz de rigidez global para a estrutura com valores nulos
    list_K_nula=[]
    for i in range (0, ord_mat):
        linha=[]
        for j in range (0, ord_mat):
            termo=0.0
            linha.append(termo)
        list_K_nula.append(linha)
    mat_K=np.array([list_K_nula])

    #o programa soma as matrizes de rigidez de cada elemento para chegar na matriz de rigidez global da estrutura
    for i in range (0, num_ele):
        lista=list_mat_rig[i]
        mat=np.array([lista])
        mat_K=mat_K+mat

    #o programa cria a matriz de rigidez global da estrutura em forma de lista e a partir desta lista gera a matriz de ridídez global da estrutura novamente
    list_K=[]
    for i in range (0, ord_mat):
        linha=mat_K[0][i]
        list_K.append(linha)
    mat_K=np.array([list_K])

#Sub-Rotina 8: Vetor de Forças Nodais

    #o programa inclui as reações de apoio aplicadas aos nós vinculados nas listas de forças externas criadas anteriormente na Sub-Rotina 3
    for i in range (0, len(list_nos)):
        if (list_restr_x[i]==1):
```

```
                list_forc_x[i]='R'
        for i in range (0, len(list_nos)):
            if (list_restr_y[i]==1):
                list_forc_y[i]='R'

        #o programa cria o vetor de forças nodais para a estrutura
        list_F=[]
        for i in range (0, len(list_nos)):
            list_F.append(list_forc_x[i])
            list_F.append(list_forc_y[i])
        mat_F_t=np.array([list_F])
        mat_F=mat_F_t.reshape(len(list_F),1)

#Sub-Rotina 9: Cálculo dos Deslocamentos Nodais e Reações de Apoio

        #o programa gera o vetor de forças nodais para a resolução do
sistema F=K*d de acordo com a Regra do 0-1
        linha_zeros=[]
        for i in range (0, len(list_F)):
            termo=list_F[i]
            if (termo=='R'):
                list_F[i]=0.0
                z=i
                linha_zeros.append(z)
        mat_F_t_sis=np.array([list_F])
        mat_F_sis=mat_F_t_sis.reshape(len(list_F),1)

        #o programa gera a matriz de rigidez global para a resolução
do sistema F=K*d de acordo com a Regra do 0-1
        for i in range (0, len(linha_zeros)):
            y=linha_zeros[i]
            for j in range (0, ord_mat):
                if (y!=j):
                    list_K[y][j]=0.0
                else:
                    list_K[y][j]=1
            for j in range (0, ord_mat):
                if (y!=j):
                    list_K[j][y]=0.0
                else:
                    list_K[j][y]=1
        mat_K_sis=np.array([list_K])

        #o programa resolve o sistema F=K*d
        mat_K_sis_inv=np.linalg.inv(mat_K_sis)
        mat_dn=np.dot(mat_K_sis_inv, mat_F_sis)
        mat_F=np.dot(mat_K, mat_dn)

#Sub-Rotina 10: Separação dos Deslocamentos Nodais por Elemento

        #o programa organiza os deslocamentos e forças nodais em uma
lista única
        list_dn=[]
        list_forc=[]
        for i in range (0, len(list_F)):
            list_dn.append(mat_dn[0][i])
```

```
        list_forc.append(mat_F[0][i])

    #o programa organiza os deslocamentos nodais em listas de
deslocamentos horizontais e verticais
    list_dn_h=[]
    list_dn_v=[]
    cont_h=0
    cont_v=1
    while (cont_v<len(list_dn)):
        termo_h=list_dn[cont_h]
        termo_v=list_dn[cont_v]
        list_dn_h.append(termo_h)
        list_dn_v.append(termo_v)
        cont_h+=2
        cont_v+=2

    #o programa organiza em uma lista várias sub-listas, em que
cada uma representa os deslocamentos nodais para cada um dos
elementos
    list_dn_ele=[]
    for i in range (0, num_ele):
        dn_ele=[]
        no_1=list_ele[i][0]
        no_2=list_ele[i][1]
        no_3=list_ele[i][2]
        no_4=list_ele[i][3]
        dn_h_1=list_dn_h[(no_1)-1]
        dn_v_1=list_dn_v[(no_1)-1]
        dn_ele.append(dn_h_1)
        dn_ele.append(dn_v_1)
        dn_h_2=list_dn_h[(no_2)-1]
        dn_v_2=list_dn_v[(no_2)-1]
        dn_ele.append(dn_h_2)
        dn_ele.append(dn_v_2)
        dn_h_3=list_dn_h[(no_3)-1]
        dn_v_3=list_dn_v[(no_3)-1]
        dn_ele.append(dn_h_3)
        dn_ele.append(dn_v_3)
        dn_h_4=list_dn_h[(no_4)-1]
        dn_v_4=list_dn_v[(no_4)-1]
        dn_ele.append(dn_h_4)
        dn_ele.append(dn_v_4)
        list_dn_ele.append(dn_ele)

#Sub-Rotina 11: Cálculo das Deformações e Tensões Nodais

    #o programa define a Matriz de Elasticidade E
    cte=(mod)/(1-(poi**2))
    list_termos_E=[]
    e11=cte
    list_termos_E.append(e11)
    e12=cte*poi
    list_termos_E.append(e12)
    e13=0
    list_termos_E.append(e13)
    e21=cte*poi
```

```
    list_termos_E.append(e21)
    e22=cte
    list_termos_E.append(e22)
    e23=0
    list_termos_E.append(e23)
    e31=0
    list_termos_E.append(e31)
    e32=0
    list_termos_E.append(e32)
    e33=cte*((1-poi)/2)
    list_termos_E.append(e33)
    mat_E_linha=np.array([list_termos_E])
    mat_E=mat_E_linha.reshape(3,3)

    #o programa cria listas para armazenar as deformações e
tensões por elemento
    list_def_x_ele=[]
    list_def_y_ele=[]
    list_def_xy_ele=[]
    list_tens_x_ele=[]
    list_tens_y_ele=[]
    list_tens_xy_ele=[]

    #o programa calcula para cada elemento as deformações e
tensões em cada um dos nós
    for i in range (0, num_ele):
        list_def_x_i=[]
        list_def_y_i=[]
        list_def_xy_i=[]
        list_tens_x_i=[]
        list_tens_y_i=[]
        list_tens_xy_i=[]
        dn_ele=list_dn_ele[i]
        mat_dn_ele_linha=np.array(dn_ele)
        mat_dn_ele=mat_dn_ele_linha.reshape(len(dn_ele),1)
        for j in range (0, 4):
            #definição matriz B
            if (j==0):
                e=-1
                n=-1
                list_termos_B=[]
                b11=(n-1)/(4*a)
                list_termos_B.append(b11)
                b12=0
                list_termos_B.append(b12)
                b13=(-n+1)/(4*a)
                list_termos_B.append(b13)
                b14=0
                list_termos_B.append(b14)
                b15=(n+1)/(4*a)
                list_termos_B.append(b15)
                b16=0
                list_termos_B.append(b16)
                b17=(-n-1)/(4*a)
                list_termos_B.append(b17)
                b18=0
```

```
        list_termos_B.append(b18)
        b21=0
        list_termos_B.append(b21)
        b22=(e-1)/(4*b)
        list_termos_B.append(b22)
        b23=0
        list_termos_B.append(b23)
        b24=(-e-1)/(4*b)
        list_termos_B.append(b24)
        b25=0
        list_termos_B.append(b25)
        b26=(e+1)/(4*b)
        list_termos_B.append(b26)
        b27=0
        list_termos_B.append(b27)
        b28=(-e+1)/(4*b)
        list_termos_B.append(b28)
        b31=(e-1)/(4*b)
        list_termos_B.append(b31)
        b32=(n-1)/(4*a)
        list_termos_B.append(b32)
        b33=(-e-1)/(4*b)
        list_termos_B.append(b33)
        b34=(-n+1)/(4*a)
        list_termos_B.append(b34)
        b35=(e+1)/(4*b)
        list_termos_B.append(b35)
        b36=(n+1)/(4*a)
        list_termos_B.append(b36)
        b37=(-e+1)/(4*b)
        list_termos_B.append(b37)
        b38=(-n-1)/(4*a)
        list_termos_B.append(b38)
        mat_B_linha=np.array([list_termos_B])
        mat_B=mat_B_linha.reshape(3,8)
        mat_def_no=np.dot(mat_B,mat_dn_ele)
        mat_tens_no=np.dot(mat_E,mat_def_no)

        for k in range (0, 3):
            if(k==0):
                list_def_x_i.append(mat_def_no[k][0])
                list_tens_x_i.append(mat_tens_no[k][0])
            if(k==1):
                list_def_y_i.append(mat_def_no[k][0])
                list_tens_y_i.append(mat_tens_no[k][0])
            if(k==2):
                list_def_xy_i.append(mat_def_no[k][0])
                list_tens_xy_i.append(mat_tens_no[k][0])
    elif (j==1):
        e=1
        n=-1
        list_termos_B=[]
        b11=(n-1)/(4*a)
        list_termos_B.append(b11)
        b12=0
        list_termos_B.append(b12)
```

```
b13=(-n+1)/(4*a)
list_termos_B.append(b13)
b14=0
list_termos_B.append(b14)
b15=(n+1)/(4*a)
list_termos_B.append(b15)
b16=0
list_termos_B.append(b16)
b17=(-n-1)/(4*a)
list_termos_B.append(b17)
b18=0
list_termos_B.append(b18)
b21=0
list_termos_B.append(b21)
b22=(e-1)/(4*b)
list_termos_B.append(b22)
b23=0
list_termos_B.append(b23)
b24=(-e-1)/(4*b)
list_termos_B.append(b24)
b25=0
list_termos_B.append(b25)
b26=(e+1)/(4*b)
list_termos_B.append(b26)
b27=0
list_termos_B.append(b27)
b28=(-e+1)/(4*b)
list_termos_B.append(b28)
b31=(e-1)/(4*b)
list_termos_B.append(b31)
b32=(n-1)/(4*a)
list_termos_B.append(b32)
b33=(-e-1)/(4*b)
list_termos_B.append(b33)
b34=(-n+1)/(4*a)
list_termos_B.append(b34)
b35=(e+1)/(4*b)
list_termos_B.append(b35)
b36=(n+1)/(4*a)
list_termos_B.append(b36)
b37=(-e+1)/(4*b)
list_termos_B.append(b37)
b38=(-n-1)/(4*a)
list_termos_B.append(b38)
mat_B_linha=np.array([list_termos_B])
mat_B=mat_B_linha.reshape(3,8)
mat_def_no=np.dot(mat_B,mat_dn_ele)
mat_tens_no=np.dot(mat_E,mat_def_no)

for k in range (0, 3):
    if(k==0):
        list_def_x_i.append(mat_def_no[k][0])
        list_tens_x_i.append(mat_tens_no[k][0])
    if(k==1):
        list_def_y_i.append(mat_def_no[k][0])
        list_tens_y_i.append(mat_tens_no[k][0])
```

```
            if(k==2):
                list_def_xy_i.append(mat_def_no[k][0])
                list_tens_xy_i.append(mat_tens_no[k][0])
    elif (j==2):
        e=1
        n=1
        list_termos_B=[]
        b11=(n-1)/(4*a)
        list_termos_B.append(b11)
        b12=0
        list_termos_B.append(b12)
        b13=(-n+1)/(4*a)
        list_termos_B.append(b13)
        b14=0
        list_termos_B.append(b14)
        b15=(n+1)/(4*a)
        list_termos_B.append(b15)
        b16=0
        list_termos_B.append(b16)
        b17=(-n-1)/(4*a)
        list_termos_B.append(b17)
        b18=0
        list_termos_B.append(b18)
        b21=0
        list_termos_B.append(b21)
        b22=(e-1)/(4*b)
        list_termos_B.append(b22)
        b23=0
        list_termos_B.append(b23)
        b24=(-e-1)/(4*b)
        list_termos_B.append(b24)
        b25=0
        list_termos_B.append(b25)
        b26=(e+1)/(4*b)
        list_termos_B.append(b26)
        b27=0
        list_termos_B.append(b27)
        b28=(-e+1)/(4*b)
        list_termos_B.append(b28)
        b31=(e-1)/(4*b)
        list_termos_B.append(b31)
        b32=(n-1)/(4*a)
        list_termos_B.append(b32)
        b33=(-e-1)/(4*b)
        list_termos_B.append(b33)
        b34=(-n+1)/(4*a)
        list_termos_B.append(b34)
        b35=(e+1)/(4*b)
        list_termos_B.append(b35)
        b36=(n+1)/(4*a)
        list_termos_B.append(b36)
        b37=(-e+1)/(4*b)
        list_termos_B.append(b37)
        b38=(-n-1)/(4*a)
        list_termos_B.append(b38)
        mat_B_linha=np.array([list_termos_B])
```

```
        mat_B=mat_B_linha.reshape(3,8)
        mat_def_no=np.dot(mat_B,mat_dn_ele)
        mat_tens_no=np.dot(mat_E,mat_def_no)

        for k in range (0, 3):
            if(k==0):
                list_def_x_i.append(mat_def_no[k][0])
                list_tens_x_i.append(mat_tens_no[k][0])
            if(k==1):
                list_def_y_i.append(mat_def_no[k][0])
                list_tens_y_i.append(mat_tens_no[k][0])
            if(k==2):
                list_def_xy_i.append(mat_def_no[k][0])
                list_tens_xy_i.append(mat_tens_no[k][0])
    elif (j==3):
        e=-1
        n=1
        list_termos_B=[]
        b11=(n-1)/(4*a)
        list_termos_B.append(b11)
        b12=0
        list_termos_B.append(b12)
        b13=(-n+1)/(4*a)
        list_termos_B.append(b13)
        b14=0
        list_termos_B.append(b14)
        b15=(n+1)/(4*a)
        list_termos_B.append(b15)
        b16=0
        list_termos_B.append(b16)
        b17=(-n-1)/(4*a)
        list_termos_B.append(b17)
        b18=0
        list_termos_B.append(b18)
        b21=0
        list_termos_B.append(b21)
        b22=(e-1)/(4*b)
        list_termos_B.append(b22)
        b23=0
        list_termos_B.append(b23)
        b24=(-e-1)/(4*b)
        list_termos_B.append(b24)
        b25=0
        list_termos_B.append(b25)
        b26=(e+1)/(4*b)
        list_termos_B.append(b26)
        b27=0
        list_termos_B.append(b27)
        b28=(-e+1)/(4*b)
        list_termos_B.append(b28)
        b31=(e-1)/(4*b)
        list_termos_B.append(b31)
        b32=(n-1)/(4*a)
        list_termos_B.append(b32)
        b33=(-e-1)/(4*b)
        list_termos_B.append(b33)
```

```
                b34=(-n+1)/(4*a)
                list_termos_B.append(b34)
                b35=(e+1)/(4*b)
                list_termos_B.append(b35)
                b36=(n+1)/(4*a)
                list_termos_B.append(b36)
                b37=(-e+1)/(4*b)
                list_termos_B.append(b37)
                b38=(-n-1)/(4*a)
                list_termos_B.append(b38)
                mat_B_linha=np.array([list_termos_B])
                mat_B=mat_B_linha.reshape(3,8)
                mat_def_no=np.dot(mat_B,mat_dn_ele)
                mat_tens_no=np.dot(mat_E,mat_def_no)

                for k in range (0, 3):
                    if(k==0):
                        list_def_x_i.append(mat_def_no[k][0])
                        list_tens_x_i.append(mat_tens_no[k][0])
                    if(k==1):
                        list_def_y_i.append(mat_def_no[k][0])
                        list_tens_y_i.append(mat_tens_no[k][0])
                    if(k==2):
                        list_def_xy_i.append(mat_def_no[k][0])
                        list_tens_xy_i.append(mat_tens_no[k][0])
        list_def_x_ele.append(list_def_x_i)
        list_def_y_ele.append(list_def_y_i)
        list_def_xy_ele.append(list_def_xy_i)
        list_tens_x_ele.append(list_tens_x_i)
        list_tens_y_ele.append(list_tens_y_i)
        list_tens_xy_ele.append(list_tens_xy_i)

#Sub-Rotina 12: Cálculo das Deformações e Tensões Nodais Médias

    #o programa cria listas para armazenar as deformações e
tensões médias para cada um dos nós
    list_def_x_nod=[]
    list_def_y_nod=[]
    list_def_xy_nod=[]
    list_tens_x_nod=[]
    list_tens_y_nod=[]
    list_tens_xy_nod=[]

    #o programa percorre todos os nós da estrutura calculando a
média dos valores encontrados de deformações e tensões para cada
nó
    no=1
    while (no<=num_no):
        def_x=[]
        def_y=[]
        def_xy=[]
        tens_x=[]
        tens_y=[]
        tens_xy=[]
        cont_med=0
        for i in range (0, len(list_ele)):
```

```
            for j in range (0, 4):
                if (list_ele[i][j]==no):
                    def_x.append(list_def_x_ele[i][j])
                    def_y.append(list_def_y_ele[i][j])
                    def_xy.append(list_def_xy_ele[i][j])
                    tens_x.append(list_tens_x_ele[i][j])
                    tens_y.append(list_tens_y_ele[i][j])
                    tens_xy.append(list_tens_xy_ele[i][j])
                    cont_med+=1
        soma_def_x=0
        soma_def_y=0
        soma_def_xy=0
        soma_tens_x=0
        soma_tens_y=0
        soma_tens_xy=0
        for i in range (0, len(def_x)):
            soma_def_x=soma_def_x+def_x[i]
            soma_def_y=soma_def_y+def_y[i]
            soma_def_xy=soma_def_xy+def_xy[i]
            soma_tens_x=soma_tens_x+tens_x[i]
            soma_tens_y=soma_tens_y+tens_y[i]
            soma_tens_xy=soma_tens_xy+tens_xy[i]
        med_def_x=soma_def_x/cont_med
        med_def_y=soma_def_y/cont_med
        med_def_xy=soma_def_xy/cont_med
        med_tens_x=soma_tens_x/cont_med
        med_tens_y=soma_tens_y/cont_med
        med_tens_xy=soma_tens_xy/cont_med
        list_def_x_nod.append(med_def_x)
        list_def_y_nod.append(med_def_y)
        list_def_xy_nod.append(med_def_xy)
        list_tens_x_nod.append(med_tens_x)
        list_tens_y_nod.append(med_tens_y)
        list_tens_xy_nod.append(med_tens_xy)
        no+=1

#Sub-Rotina 13: Saída de Dados

    # o programa exporta os resultados obtidos em arquivo.txt

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

        saida.write('\n')
        saida.write('   RESULTADOS OBTIDOS - ANÁLISE ESTÁTICA DE
CHAPAS VIA MÉTODO DOS ELEMENTOS FINITOS\n')
        saida.write('\n')
        saida.write('   Nome do arquivo: '+ nome +'\n')
        saida.write('\n')
        saida.write('   DESLOCAMENTOS NODAIS\n')
        saida.write('\n')
        saida.write('   Nó  Deslocamento x ('+str(un_metrica)+')\n')
        cont_desl=0
        for i in range (0, num_no):
            saida.write('                                '+str(i+1)+'
'+str('{0:.8f}'.format(float(list_dn[cont_desl])))+'\n')
#formato de 8 casas decimais
```

```
            cont_desl+=2
        saida.write('\n')
        saida.write('   Nó  Deslocamento y ('+str(un_metrica)+')\n')
        cont_desl=1
        for i in range (0, num_no):
            saida.write('                                '+str(i+1)+'
'+str('{0:.8f}'.format(float(list_dn[cont_desl])))+'\n')
            cont_desl+=2
        saida.write('\n')

        saida.write('   FORÇAS NODAIS\n')
        saida.write('\n')
        saida.write('   Nó  Força x ('+str(un_forca)+')\n')
        cont_forc=0
        for i in range (0, num_no):
            saida.write('                                '+str(i+1)+'
'+str('{0:.8f}'.format(float(list_forc[cont_forc])))+'\n')
            cont_forc+=2
        saida.write('\n')
        saida.write('   Nó  Força y ('+str(un_forca)+')\n')
        cont_forc=1
        for i in range (0, num_no):
            saida.write('                                '+str(i+1)+'
'+str('{0:.8f}'.format(float(list_forc[cont_forc])))+'\n')
            cont_forc+=2
        saida.write('\n')

        saida.write('   DEFORMAÇÕES NORMAIS DIREÇÃO X\n')
        saida.write('\n')
        saida.write('                Nó          Deformação     x
('+str(un_metrica)+'/'+str(un_metrica)+')\n')
        for i in range (0,num_no):
            saida.write('                                '+str(i+1)+'
'+str('{0:.8f}'.format(float(list_def_x_nod[i])))+'\n')
        saida.write('\n')

        saida.write('   DEFORMAÇÕES NORMAIS DIREÇÃO Y\n')
        saida.write('\n')
        saida.write('                Nó          Deformação     y
('+str(un_metrica)+'/'+str(un_metrica)+')\n')
        for i in range (0,num_no):
            saida.write('                                '+str(i+1)+'
'+str('{0:.8f}'.format(float(list_def_y_nod[i])))+'\n')
        saida.write('\n')

        saida.write('   DEFORMAÇÕES ANGULARES\n')
        saida.write('\n')
        saida.write('                Nó          Deformação     xy
('+str(un_metrica)+'/'+str(un_metrica)+')\n')
        for i in range (0,num_no):
            saida.write('                                '+str(i+1)+'
'+str('{0:.8f}'.format(float(list_def_xy_nod[i])))+'\n')
        saida.write('\n')

        saida.write('   TENSÕES NORMAIS DIREÇÃO X\n')
        saida.write('\n')
```

```
        saida.write('                  Nó           Tensão      x
('+str(un_forca)+'/'+str(un_metrica)+str(2)+')\n')
        for i in range (0,num_no):
            saida.write('                              '+str(i+1)+'
'+str('{0:.8f}'.format(float(list_tens_x_nod[i])))+'\n')
        saida.write('\n')

        saida.write('   TENSÕES NORMAIS DIREÇÃO Y\n')
        saida.write('\n')
        saida.write('                  Nó           Tensão      y
('+str(un_forca)+'/'+str(un_metrica)+str(2)+')\n')
        for i in range (0,num_no):
            saida.write('                              '+str(i+1)+'
'+str('{0:.8f}'.format(float(list_tens_y_nod[i])))+'\n')
        saida.write('\n')

        saida.write('   TENSÕES CISALHANTES\n')
        saida.write('\n')
        saida.write('                    Nó       Tensão       xy
('+str(un_forca)+'/'+str(un_metrica)+str(2)+')\n')
        for i in range (0,num_no):
            saida.write('                              '+str(i+1)+'
'+str('{0:.8f}'.format(float(list_tens_xy_nod[i])))+'\n')
        saida.write('\n')

    print('ANÁLISE COMPLETA! ABRA O ARQUIVO SAIDA_DE_DADOS.TXT
PARA OBTER OS RESULTADOS.')
```

APÊNDICE B – ARQUIVO DE SAÍDA DO EXEMPLO 1 NA MALHA DE 16 ELEMENTOS CSQ

RESULTADOS OBTIDOS – ANÁLISE ESTÁTICA DE CHAPAS VIA MÉTODO DOS ELEMENTOS FINITOS

Nome do arquivo: Exemplo 1 malha 5x5

DESLOCAMENTOS NODAIS

Nó	Deslocamento x (cm)
1	0.00000000
2	0.00042518
3	0.00082992
4	0.00125252
5	0.00167643
6	0.00000000
7	0.00039974
8	0.00083049
9	0.00125342
10	0.00167677
11	0.00000000
12	0.00039548
13	0.00082743
14	0.00125429
15	0.00167682
16	0.00000000
17	0.00039974
18	0.00083049
19	0.00125342
20	0.00167677
21	0.00000000
22	0.00042518
23	0.00082992
24	0.00125252
25	0.00167643

Nó	Deslocamento y (cm)
1	0.00000000
2	0.00013215
3	0.00013006
4	0.00012737
5	0.00012644
6	0.00000000
7	0.00006137
8	0.00006772
9	0.00006340
10	0.00006318
11	0.00000000
12	0.00000000

13	0.00000000
14	0.00000000
15	0.00000000
16	0.00000000
17	-0.00006137
18	-0.00006772
19	-0.00006340
20	-0.00006318
21	0.00000000
22	-0.00013215
23	-0.00013006
24	-0.00012737
25	-0.00012644

FORÇAS NODAIS

Nó	Força x (kN)
1	-6.51426591
2	-0.00000000
3	-0.00000000
4	0.00000000
5	5.56000000
6	-10.54416838
7	0.00000000
8	0.00000000
9	-0.00000000
10	11.12000000
11	-10.36313142
12	-0.00000000
13	0.00000000
14	0.00000000
15	11.12000000
16	-10.54416838
17	-0.00000000
18	-0.00000000
19	0.00000000
20	11.12000000
21	-6.51426591
22	0.00000000
23	-0.00000000
24	0.00000000
25	5.56000000

Nó	Força y (kN)
1	-2.62979384
2	0.00000000
3	0.00000000
4	0.00000000
5	0.00000000

6	-0.53819113
7	-0.00000000
8	0.00000000
9	-0.00000000
10	-0.00000000
11	0.00000000
12	-0.00000000
13	-0.00000000
14	-0.00000000
15	0.00000000
16	0.53819113
17	0.00000000
18	0.00000000
19	0.00000000
20	-0.00000000
21	2.62979384
22	-0.00000000
23	-0.00000000
24	-0.00000000
25	0.00000000

DEFORMAÇÕES NORMAIS DIREÇÃO X

Nó	Deformação x (cm/cm)
1	0.00003348
2	0.00003267
3	0.00003257
4	0.00003333
5	0.00003338
6	0.00003148
7	0.00003270
8	0.00003361
9	0.00003332
10	0.00003333
11	0.00003114
12	0.00003258
13	0.00003381
14	0.00003344
15	0.00003327
16	0.00003148
17	0.00003270
18	0.00003361
19	0.00003332
20	0.00003333
21	0.00003348
22	0.00003267
23	0.00003257
24	0.00003333
25	0.00003338

DEFORMAÇÕES NORMAIS DIREÇÃO Y

Nó	Deformação y (cm/cm)
1	0.00000000
2	-0.00001115
3	-0.00000982
4	-0.00001007
5	-0.00000996
6	0.00000000
7	-0.00001041
8	-0.00001024
9	-0.00001003
10	-0.00000996
11	0.00000000
12	-0.00000967
13	-0.00001066
14	-0.00000998
15	-0.00000995
16	0.00000000
17	-0.00001041
18	-0.00001024
19	-0.00001003
20	-0.00000996
21	0.00000000
22	-0.00001115
23	-0.00000982
24	-0.00001007
25	-0.00000996

DEFORMAÇÕES ANGULARES

Nó	Deformação xy (cm/cm)
1	0.00001041
2	0.00000111
3	-0.00000010
4	-0.00000000
5	-0.00000002
6	0.00000483
7	0.00000033
8	-0.00000012
9	-0.00000004
10	0.00000001
11	0.00000000
12	0.00000000
13	-0.00000000
14	0.00000000
15	-0.00000000

16	-0.00000483
17	-0.00000033
18	0.00000012
19	0.00000004
20	-0.00000001
21	-0.00001041
22	-0.00000111
23	0.00000010
24	0.00000000
25	0.00000002

TENSÕES NORMAIS DIREÇÃO X

Nó	Tensão x (kN/cm2)
1	0.76097701
2	0.66667490
3	0.67342022
4	0.68883035
5	0.69077221
6	0.71543791
7	0.67223800
8	0.69410914
9	0.68892698
10	0.68979822
11	0.70782126
12	0.67453961
13	0.69580095
14	0.69203176
15	0.68838929
16	0.71543791
17	0.67223800
18	0.69410914
19	0.68892698
20	0.68979822
21	0.76097701
22	0.66667490
23	0.67342022
24	0.68883035
25	0.69077221

TENSÕES NORMAIS DIREÇÃO Y

Nó	Tensão y (kN/cm2)
1	0.22829310
2	-0.03053593
3	-0.00103652
4	-0.00173023
5	0.00117819

6	0.21463137
7	-0.01355660
8	-0.00359577
9	-0.00076189
10	0.00100466
11	0.21234638
12	0.00244428
13	-0.01185416
14	0.00110890
15	0.00070065
16	0.21463137
17	-0.01355660
18	-0.00359577
19	-0.00076189
20	0.00100466
21	0.22829310
22	-0.03053593
23	-0.00103652
24	-0.00173023
25	0.00117819

TENSÕES CISALHANTES

Nó	Tensão xy (kN/cm2)
1	0.08278000
2	0.00885888
3	-0.00077569
4	-0.00000048
5	-0.00015516
6	0.03844569
7	0.00260649
8	-0.00092806
9	-0.00031318
10	0.00011477
11	0.00000000
12	-0.00000000
13	-0.00000000
14	0.00000000
15	-0.00000000
16	-0.03844569
17	-0.00260649
18	0.00092806
19	0.00031318
20	-0.00011477
21	-0.08278000
22	-0.00885888
23	0.00077569
24	0.00000048
25	0.00015516