UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica
Curso de Engenharia Civil
Departamento de Mecânica Aplicada e Estruturas

MODELAGEM NUMÉRICA DE PLACAS ESPESSAS USANDO MÉTODO DOS ELEMENTOS FINITOS COM COMPUTAÇÃO PARALELA

RICARDO CALDEIRA DE OLIVEIRA

Projeto Final de Graduação apresentado ao corpo docente do Departamento de Mecânica Aplicada e Estruturas da Escola Politécnica da Universidade Federal do Rio de Janeiro, como requisito para obtenção do título de Engenheiro Civil.

Aprovado por:

José Antônio Fontes Santiago
D.Sc., COPPE/UFRJ (Orientador)

Gilberto Bruno Ellwanger
Prof. Associado, D.Sc., EP/UFRJ

Ricardo Valeriano Alves
Prof. Adjunto, D.Sc., EP/UFRJ

Novembro / 2009

Resumo da Dissertação apresentada ao DME/POLI/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Civil.

MODELAGEM NUMÉRICA DE PLACAS ESPESSAS USANDO MÉTODO DOS ELEMENTOS FINITOS COM COMPUTAÇÃO PARALELA

Ricardo Caldeira de Oliveira

Novembro/2009

Orientador: José Antônio Fontes Santiago

Curso: Engenharia Civil

O presente trabalho consiste no estudo e na aplicação do método dos elementos finitos à teoria de placas espessas. A formulação do método dos elementos finitos foi feita com base na fundamentação de placas espessas da teoria de *Mindlin*.

A aplicação numérica foi feita em linguagem *FORTRAN*, onde além da simples implementação do programa de elementos finitos de placa espessa, foi implementada a paralelização da programação para melhor desempenho das análises em computadores de arquitetura *multi-core*.

Exemplos numéricos foram formulados e comparados com resultados clássicos da literatura para validade de toda a formulação dos elementos e programa.

AGRADECIMENTOS

Agradeço aos meus pais Regina e Carlos, por acreditarem sempre em mim e pelo investimento no meu futuro. Apesar da distância geográfica, o apoio deles sempre foi fundamental.

À minha tia Liliana, que me forneceu abrigo, muito carinho e tudo mais que pudesse me faltar nos primeiros anos de faculdade intensivamente, e posteriormente, fez com que os dias do final de semana fossem os mais aguardados sempre.

Aos meus irmãos Renato, Matheus, Paulo César e José Aluizio. Incluo além do meu irmão de sangue Renato, aqueles que juntamente com este colaboraram para que meus dias fossem mais divertidos e compartilharam muitas passagens do meu viver.

À minha noiva Silvia Leal Soares, que definitivamente transformou minha vida me dando incentivo, carinho e atenção a todo instante. Sou muito grato a cada atitude e agradeço ainda sua colaboração, que foi imensurável não só para a minha formação como para a minha vida. Aproveito ainda para agradecer seus pais Miguel e Silvia pela cordialidade e hospitalidade, em especial, sua mãe por jamais ter deixado faltar o copo de mate e o café quando fazíamos os trabalhos da faculdade.

A todos os grandes amigos que conheci no Laboratório de Análise e Confiabilidade de Estruturas Offshore (LACEO). Em especial, ao fanático José Bazán e aos amigos das baias 205 e 206. Obrigado por me darem o prazer de trabalhar no Fundão.

Agradeço muito especialmente ao meu orientador José A. F. Santiago, que além de ser muito amigo, fez o trabalho de iniciação científica aguçar meus interesses na área acadêmica. Esta foi uma fase marcante que me deu ainda mais força e uma visão mais aberta para encarar os desafios da graduação.

Ao amigo Ricardo N. Deslandes Júnior que além de um grande amigo de faculdade me ajudou muito ultimamente a recuperar ânimo para treinar e completar a meia-maratona. A gente vai completar uma maratona em breve!

Ao amigo Fernando C. Marcos por nunca ter se esquecido de mim para o futebol de terça à noite. Agradeço ainda a todos os outros companheiros do futebol.

Ao Carlos Eugênio (Neném), que trouxe o esporte para minha vida, além de diversos outros hábitos e ensinamentos que me fizeram encontrar equilíbrio de mente, corpo e espírito para traçar, lutar e alcançar minhas metas.

Por fim, aos meus colegas de turma pela diversão, ajuda, carinho, conversas, churrascos, risadas, enfim, por literalmente tudo que vocês representam pra mim. Agradeço especialmente à ala niteroiense que de forma muito animada dividiu comigo muitos momentos de alegria durante o lazer e nas longas viagens no ônibus 998.

ÍNDICE

CAPÍTULO I

INTRODUÇÃO

I.1. Motivação

Atualmente as análises numéricas vêm tomando uma importância significativa, uma vez que os problemas e a ousadia humana no desenvolvimento da tecnologia são cada vez maiores. Juntamente com o aumento dos modelos numéricos, vem o desenvolvimento de ferramentas computacionais que possibilitem a análise desses problemas. Portanto, existem inúmeras pesquisas relacionadas à computação de alto desempenho para que possamos desfrutar sempre da capacidade máxima dos *hardwares* disponíveis.

O método dos elementos finitos é uma das ferramentas numéricas mais usadas modernamente para o desenvolvimento de pesquisas e projetos de engenharia. O desenvolvimento do método, juntamente com a disponibilidade cada vez maior de computadores de alto desempenho, possibilitou sua extensão para diversas áreas científicas.

Dentre as aplicações de elementos finitos relacionadas à elasticidade, um dos temas que oferece uma grande visibilidade no meio acadêmico e empresarial é a análise de flexão de placas. A aplicação do método dos elementos finitos permite a análise de problemas de geometria e carregamento complexos, onde soluções analíticas são difíceis e muitas vezes nem podem ser obtidas. Portanto, o método dos elementos finitos abre um amplo campo de pesquisas relacionadas a esse tipo de análise numérica.

I.2.Objetivo

Este trabalho tem por objetivo aplicar o método dos elementos finitos à análise estática linear de placa espessa. Será feita uma discussão com relação à

teoria de placas e, posteriormente, serão feitos alguns esclarecimentos com relação ao método dos elementos finitos para, então, aplicá-lo à teoria de placas espessas.

Este trabalho tem ainda como objetivo o desenvolvimento de um código computacional para aplicação do método dos elementos finitos para o problema em questão, fazendo o uso de linguagem *FORTRAN*. Neste código será feita ainda a implementação de trechos para execução em paralelo para máquinas de arquitetura *multi-core*, de forma buscar um maior desempenho na execução do programa.

I.3.Organização do Texto

O Capítulo II detalha alguns conceitos básicos com relação às teorias de placas. Posteriormente é feita uma abordagem mais detalhada do problema de placas espessas, que é o alvo deste trabalho.

O Capítulo III apresenta o embasamento teórico para o método dos elementos finitos de uma forma geral.

O Capítulo IV trata do desenvolvimento do método dos elementos finitos aplicado ao caso particular de placa espessa.

No Capítulo V são feitas algumas considerações em relação aos aspectos computacionais envolvidos no código computacional elaborado, citando como foi idealizada e executada a implementação da paralelização do código.

O Capítulo VI apresenta os resultados da aplicação prática do programa gerado.

O Capítulo VII aponta as conclusões que foram possíveis obter com a execução deste trabalho e sugestões para trabalhos futuros.

CAPÍTULO II

TEORIA DE FLEXÃO DE PLACAS

II.1. Introdução

As placas são elementos estruturais planos, submetidos a carregamentos transversais ao seu plano, com espessura muito menor que suas demais dimensões. Porém, através de uma forma mais genérica, é possível enquadrar ainda o caso de elementos carregados em seu próprio plano na teoria de flexão, quando os deslocamentos transversais passam a ser da ordem ou maiores que a espessura da placa, tornando os efeitos da não-linearidade geométrica significativos.

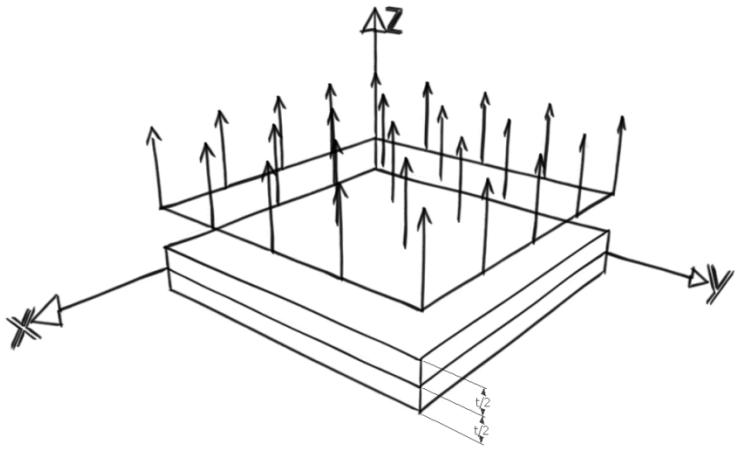


Figura II.1 – Representação do sistema de coordenadas cartesianas.

Segundo *Timoshenko*[19], o problema de flexão de placas depende fundamentalmente da espessura da placa quando comparada às outras dimensões e deve ser distinguido em três problemas distintos. Sendo eles:

1. Placas finas com pequenos deslocamentos transversais;

2. Placas finas com grandes deslocamentos transversais;

3. Placas espessas;

A consideração de placa fina com pequenos deslocamentos transversais (teoria clássica proposta por *Kirchhoff*) é válida se os deslocamentos transversais da placa são pequenos quando comparados à espessura *t*. Sendo assim, uma boa aproximação do comportamento de flexão da placa pode ser feita, considerando somente deformações devido à flexão, tomando as seguintes considerações:

i. O plano médio da placa pode ser considerado indeformável, permanecendo neutro durante a flexão. O que é válido se os deslocamentos transversais são pequenos quando comparados à espessura da placa;

ii. Retas normais ao plano médio da placa indeformada deverão continuar sendo consideradas normais ao plano médio deformado (desprezando assim as distorções γ_{xz} e γ_{yz});

iii. As tensões σ_z podem ser desprezadas.

Com essas considerações, o problema é traduzido matematicamente em um operador diferencial de quarta ordem, que pode ser solucionado impondo-se duas condições de contorno por bordo tendo, ainda, todas as componentes de tensão expressas em termos dos deslocamentos transversais *w*, que será um campo dependente somente das duas coordenadas no plano da placa.

Para placas finas com grandes deslocamentos transversais (placas muito esbeltas), existem teorias na literatura como a proposta por *Von-Kárman*, que consideram os efeitos da não-linearidade geométrica em sua formulação, incluindo o efeito dos carregamentos no plano da placa. Soluções a partir desta teoria podem ser aplicadas em análise de estabilidade elástica de placas e servem de base para geração de fórmulas muito utilizadas na análise de flambagem de

chapas em projeto de estruturas metálicas, frequentemente usadas, por exemplo, na indústria naval.

Em alguns casos a consideração das hipóteses anteriormente mencionadas não traduz satisfatoriamente o fenômeno analisado, principalmente quando a espessura da placa aumenta e quando há situações de carga concentrada. Nestes casos a teoria de placas espessas deve ser aplicada. Esta teoria considera o problema de flexão de placas um problema tridimensional da teoria de elasticidade, o que leva a uma análise de tensões muito mais complexa e fazendo com que as soluções analíticas fechadas sejam encontradas somente para casos muito particulares.

Por se tratar de um caso mais genérico (englobando placas finas e espessas com pequeno deslocamento transversal), a teoria de placas espessas foi escolhida para o desenvolvimento do trabalho. No entanto, será eventualmente mencionada a teoria de placas delgadas (clássica) em comparações de seus comportamentos. Sendo assim, a teoria escolhida na abordagem foi a teoria de *Mindlin*, que difere da teoria de *Reissner* em suas hipóteses iniciais.

II.2. Teoria de Mindlin

As placas espessas têm em seus campos de deslocamento parcelas relativas ao efeito de flexão e à deformação cisalhante (de forma análoga à viga de *Timoshenko*), originada das distorções fora do seu plano. Este problema, de acordo com as hipóteses mais conhecidas na literatura (*Reissner* e *Mindlin*), recai em um sistema de equações diferenciais, que pode ser solucionado impondo-se três condições de contorno por bordo.

Na teoria clássica (proposta por *Kirchhoff*), as retas normais ao plano médio indeformável antes da deformação permanecem retas e normais ao mesmo após a deformação, desprezando assim as contribuições das deformações cisalhantes. Na teoria de *Mindlin* essas retas não são mais necessariamente normais ao plano médio.

O problema a ser analisado envolve três campos de deslocamento ($w(x,y),\theta_{xz}(x,y),\theta_{yz}(x,y)$) e por conveniência foi adotado o sistema de convenções para sinais positivos mostrado na Figura II.2. Todo o problema será desenvolvido e escrito em função destes campos.

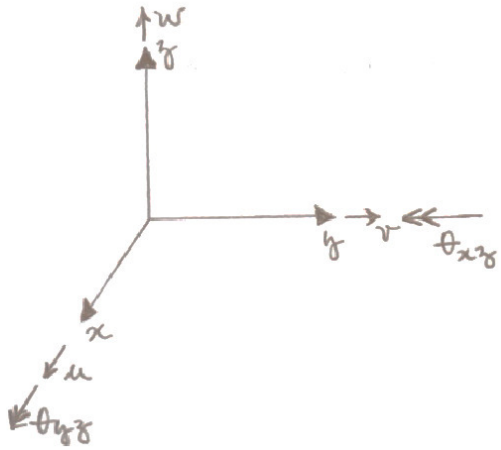


Figura II.2 – Conveção de sinais para deslocamentos e rotações.

II.2.1. Definição das hipóteses:

A proposta de *Mindlin* foi baseada nas seguintes hipóteses simplificadoras:

i. O plano médio da placa pode ser considerado indeformável, permanecendo neutro durante a flexão. O que é válido se os deslocamentos transversais são pequenos quando comparados à espessura da placa;

ii. Retas normais à superfície de referência indeformada permanecerão retas após a deformação, porém, não mais serão necessariamente normais ao plano médio;

iii. A componente de tensão normal σ_z é desprezível.

Sendo assim, o campo de deslocamentos fica definido como uma composição de deformações devido ao efeito de flexão e ao efeito de cisalhamento.

A teoria de *Reissner* irá recair nas mesmas expressões finais que a teoria de *Mindlin*, porém, as duas diferem nas suas hipóteses iniciais. *Reissner* considerou uma variação linear das tensões ao longo da espessura como hipótese inicial, enquanto *Mindlin* considerou uma variação linear das deformações.

II.2.2.Equações Geométricas

A hipótese de *Mindlin* considera uma variação uniforme de deformação ao longo da altura, como mostra a Figura II.3.

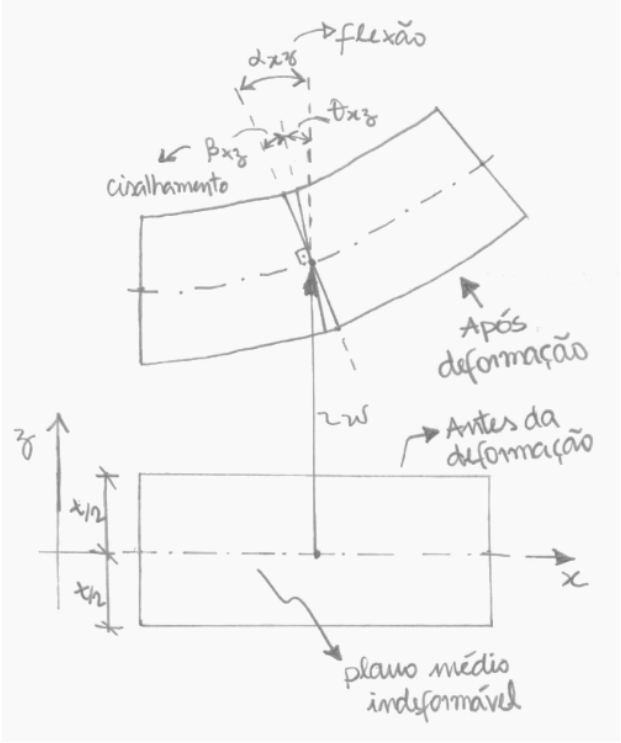


Figura II.3 –Deslocamentos e rotações de uma placa espessa no plano xz.

Através dessas considerações, é possível expressar os deslocamentos pontuais da placa, conforme a seguir.

$$u(x,y,z) = -z \cdot \theta_{xz}(x,y) \tag{II.1}$$

$$v(x,y,z) = -z \cdot \theta_{yz}(x,y) \tag{II.2}$$

$$w(x,y,z) = w(x,y) \tag{II.3}$$

Ou na seguinte forma vetorial:

$$\mathbf{u}(x,y,z) = \begin{bmatrix} u(x,y,z) \\ v(x,y,z) \\ w(x,y,z) \end{bmatrix} = \begin{bmatrix} -z \cdot \theta_{xz}(x,y) \\ -z \cdot \theta_{yz}(x,y) \\ w(x,y) \end{bmatrix} \tag{II.4}$$

Portanto, ficam definidos os deslocamentos u, v e w em função dos três campos de deslocamento desconhecidos.

Sendo θ definido por uma composição de rotações devido ao efeito de flexão (deixando a reta normal à superfície de referência) e rotações devido ao efeito de distorção, chega-se às relações mostradas abaixo.

$$\theta_{xz}(x,y) = \frac{\partial w(x,y)}{\partial x} - \beta_{xz}(x,y) \tag{II.5}$$

$$\theta_{yz}(x,y) = \frac{\partial w(x,y)}{\partial y} - \beta_{yz}(x,y) \tag{II.6}$$

Com isso, as relações de deslocamentos ficam mais claramente expressas em relação aos termos de flexão e distorção de forma distinta, conforme mostrado a seguir.

$$\mathbf{u}(x,y,z) = \begin{bmatrix} u(x,y,z) \\ v(x,y,z) \\ w(x,y,z) \end{bmatrix} = \begin{bmatrix} -z \cdot \left(\frac{\partial w(x,y)}{\partial x} - \beta_{xz}(x,y)\right) \\ -z \cdot \left(\frac{\partial w(x,y)}{\partial y} - \beta_{yz}(x,y)\right) \\ w(x,y) \end{bmatrix} \tag{II.7}$$

Fazendo uma breve analogia com a teoria de flexão clássica (*Kirchhoff*), houve a inclusão das distorções geradas pela deformação cisalhante $\beta_{xz}(x,y)$ e $\beta_{yz}(x,y)$.

Partindo das relações deformação-deslocamento definidas na teoria da elasticidade linear, é possível expressar as componentes de deformação da seguinte forma:

$$\varepsilon_x(x,y,z) = \frac{\partial u(x,y,z)}{\partial x} = -z \cdot \frac{\partial \theta_{xz}(x,y)}{\partial x} \quad \text{(II.8)}$$

$$\varepsilon_y(x,y,z) = \frac{\partial v(x,y,z)}{\partial y} - z \cdot \frac{\partial \theta_{yz}(x,y)}{\partial y} \quad \text{(II.9)}$$

$$\gamma_{xy}(x,y,z) = \frac{\partial u(x,y,z)}{\partial y} + \frac{\partial v(x,y,z)}{\partial x} = -z \cdot \left(\frac{\partial \theta_{xz}(x,y)}{\partial y} + \frac{\partial \theta_{yz}(x,y)}{\partial x}\right) \quad \text{(II.10)}$$

$$\gamma_{xz}(x,y,z) = \frac{\partial u(x,y,z)}{\partial z} + \frac{\partial w(x,y,z)}{\partial x} = -\theta_{xz}(x,y) + \frac{\partial w(x,y)}{\partial x} \quad \text{(II.11)}$$

$$\gamma_{yz}(x,y,z) = \frac{\partial v(x,y,z)}{\partial z} + \frac{\partial w(x,y,z)}{\partial y} = -\theta_{yz}(x,y) + \frac{\partial w(x,y)}{\partial y} \quad \text{(II.12)}$$

Lembrando que as deformações perpendiculares ao plano médio, ε_z, são nulas.

$$\varepsilon_z(x,y,z) = \frac{\partial w(x,y,z)}{\partial z} = \frac{\partial w(x,y)}{\partial z} = 0 \quad \text{(II.13)}$$

Substituindo as expressões (II.5 e II.6) na representação das deformações, é possível obter as equações a seguir (II.14 a II.18).

$$\varepsilon_x(x,y,z) = -z \cdot \left(\frac{\partial^2 w(x,y)}{\partial x^2} - \frac{\partial \beta_{xz}(x,y)}{\partial x}\right) \quad \text{(II.14)}$$

$$\varepsilon_y(x,y,z) = -z \cdot \left(\frac{\partial^2 w(x,y)}{\partial y^2} - \frac{\partial \beta_{yz}(x,y)}{\partial y}\right) \quad \text{(II.15)}$$

$$\gamma_{xy}(x,y,z) = -z \cdot \left(2 \cdot \frac{\partial^2 w(x,y)}{\partial x \partial y} - \frac{\partial \beta_{yz}(x,y)}{\partial x} - \frac{\partial \beta_{xz}(x,y)}{\partial y}\right) \quad \text{(II.16)}$$

$$\gamma_{xz}(x,y,z) = \beta_{xz}(x,y) \quad \text{(II.17)}$$

$$\gamma_{yz}(x,y,z) = \beta_{yz}(x,y) \quad \text{(II.18)}$$

As expressões de deformação não serão trabalhadas nessa forma no desenvolvimento posterior, embora na forma apresentada em (II.14 a II.18) as deformações fiquem mais elegantes e seja possível fazer uma analogia clara com as representações das mesmas na teoria clássica (*kirchhoff*), pois é possível perceber claramente que só houve a introdução das deformações cisalhantes.

Para uma representação mais compacta podemos trabalhar as expressões (II.8 a II.12) na forma vetorial. Logo, as deformações podem ser expressas por:

$$\boldsymbol{\varepsilon_b} = \begin{bmatrix} \varepsilon_x(x,y,z) \\ \varepsilon_y(x,y,z) \\ \gamma_{xy}(x,y,z) \end{bmatrix} = -z \cdot \begin{bmatrix} \frac{\partial \theta_{xz}(x,y)}{\partial x} \\ \frac{\partial \theta_{yz}(x,y)}{\partial y} \\ \left(\frac{\partial \theta_{xz}(x,y)}{\partial y} + \frac{\partial \theta_{yz}(x,y)}{\partial x}\right) \end{bmatrix} \tag{II.19}$$

$$\boldsymbol{\varepsilon_s} = \begin{bmatrix} \gamma_{xz}(x,y,z) \\ \gamma_{yz}(x,y,z) \end{bmatrix} = \begin{bmatrix} \frac{\partial u(x,y,z)}{\partial z} + \frac{\partial w(x,y,z)}{\partial x} \\ \frac{\partial v(x,y,z)}{\partial z} + \frac{\partial w(x,y,z)}{\partial y} \end{bmatrix} = \begin{bmatrix} -\theta_{xz}(x,y) + \frac{\partial w(x,y)}{\partial x} \\ -\theta_{yz}(x,y) + \frac{\partial w(x,y)}{\partial y} \end{bmatrix} \tag{II.20}$$

No entanto, será referenciado o vetor de deformações por flexão em função do vetor de curvaturas $\boldsymbol{\kappa_b}$. Esta representação simplifica muito o desenvolvimento mais adiante.

$$\boldsymbol{\varepsilon_b} = -z \cdot \boldsymbol{\kappa_b} \tag{II.21}$$

II.2.3.Equações Constitutivas

As equações constitutivas estabelecem a relação entre as tensões e as deformações do problema estudado. Portanto, tais relações irão fundamentalmente depender do material que está sendo estudado. Na grande maioria dos materiais, quando se trabalha com tensões bem abaixo da tensão de ruptura ou do patamar de escoamento e com pequenas deformações, pode ser dito que esta relação é linear.

A consideração de linearidade física do material simplifica consideravelmente as relações tensão-deformação e a solução dos problemas de elasticidade. No entanto, esta se trata de uma simplificação que irá impor uma limitação à análise de problemas particulares, surgindo eventualmente necessidade de se trabalhar com outras hipóteses.

De acordo com a Lei de *Hooke* e lembrando a hipótese de tensões transversais ao plano nulas (σ_z=0), podemos escrever as deformações e distorções em função das tensões normais e cisalhantes da seguinte forma desacoplada:

$$\begin{bmatrix}\varepsilon_x\\ \varepsilon_y\\ \gamma_{xy}\end{bmatrix} = \begin{bmatrix}1/E & -\nu/E & 0\\ -\nu/E & 1/E & 0\\ 0 & 0 & \frac{1}{G}\end{bmatrix} \cdot \begin{bmatrix}\sigma_x\\ \sigma_y\\ \tau_{xy}\end{bmatrix} \tag{II.22}$$

$$\begin{bmatrix}\gamma_{xz}\\ \gamma_{yz}\end{bmatrix} = \begin{bmatrix}\frac{1}{G} & 0\\ 0 & \frac{1}{G}\end{bmatrix} \cdot \begin{bmatrix}\tau_{xz}\\ \tau_{yz}\end{bmatrix} \tag{II.23}$$

Sendo G o módulo de elasticidade transversal dado por:

$$G = \frac{E}{2\cdot(1+\nu)} \tag{II.24}$$

Invertendo-se as matrizes que relacionam as tensões com as deformações se obtém:

$$\begin{bmatrix}\sigma_x\\ \sigma_y\\ \tau_{xy}\end{bmatrix} = \frac{E}{1-\nu^2} \cdot \begin{bmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & \frac{G\cdot(1-\nu^2)}{E}\end{bmatrix} \cdot \begin{bmatrix}\varepsilon_x\\ \varepsilon_y\\ \gamma_{xy}\end{bmatrix} \tag{II.25}$$

$$\begin{bmatrix}\tau_{xz}\\ \tau_{yz}\end{bmatrix} = \begin{bmatrix}G & 0\\ 0 & G\end{bmatrix} \cdot \begin{bmatrix}\gamma_{xz}\\ \gamma_{yz}\end{bmatrix} \tag{II.26}$$

Operando convenientemente as expressões descritas anteriormente, é possível expressar as tensões em termo dos campos independentes.

A distribuição de tensões normais e cisalhantes no plano da placa assume uma forma linear ao longo da espessura da placa, conforme a seguir.

$$\boldsymbol{\sigma_b} = \begin{bmatrix}\sigma_x\\ \sigma_y\\ \tau_{xy}\end{bmatrix} = \begin{bmatrix}-\frac{E}{1-\nu^2}\cdot z \cdot \left[\frac{\partial\theta_{xz}(x,y)}{\partial x} + \nu \cdot \frac{\partial\theta_{yz}(x,y)}{\partial y}\right]\\ -\frac{E}{1-\nu^2}\cdot z \cdot \left[\frac{\partial\theta_{yz}(x,y)}{\partial y} + \nu \cdot \frac{\partial\theta_{xz}(x,y)}{\partial x}\right]\\ -G \cdot z \cdot \left(\frac{\partial\theta_{xz}(x,y)}{\partial y} + \frac{\partial\theta_{yz}(x,y)}{\partial x}\right)\end{bmatrix} \tag{II.27}$$

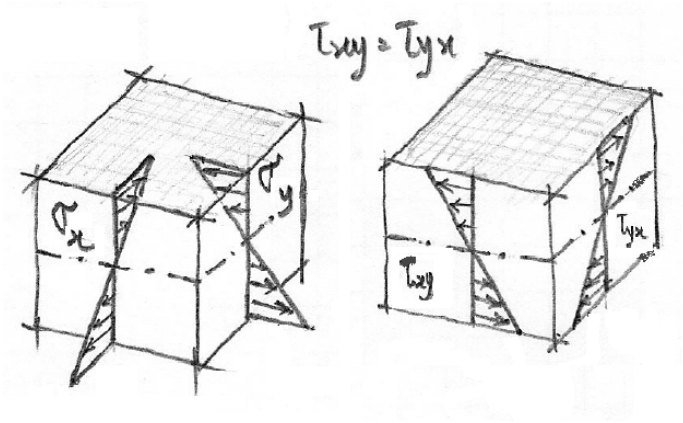


Figura II.4 – Distribuição de tensões na placa (Fonte: VALERIANO[20])

A mesma operação pode ser feita para as tensões cisalhantes fora do plano da placa, que leva a uma distribuição uniforme conforme a seguir:

$$\boldsymbol{\sigma_s} = \begin{bmatrix}\tau_{xz}\\ \tau_{yz}\end{bmatrix} = G \cdot \begin{bmatrix}-\theta_{xz}(x,y) + \frac{\partial w(x,y)}{\partial x}\\ -\theta_{yz}(x,y) + \frac{\partial w(x,y)}{\partial y}\end{bmatrix} \tag{II.28}$$

II.2.4.Solicitações

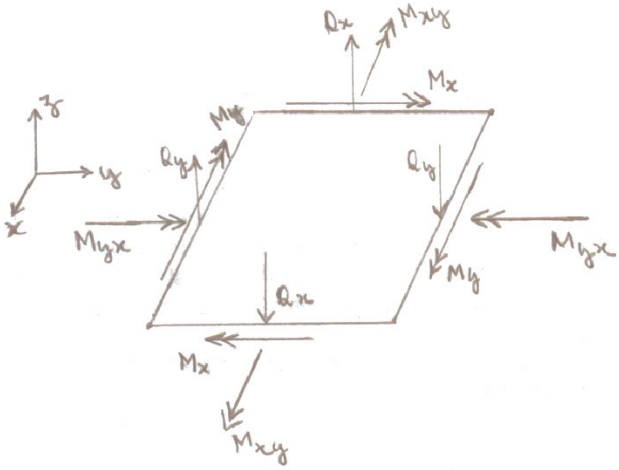


Figura II.5 – Convenção para os esforços.

Os esforços solicitantes, que são os momentos fletores, momentos torçores e cortantes, podem ser expressos integrando-se as tensões ao longo da

espessura da placa. Vale lembrar ainda que os esforços são dados por unidade de comprimento.

II.2.4.1.Momentos Fletores e de Torção

Integrando-se as tensões normais e cisalhantes no plano da placa ao longo da sua espessura, encontra-se (lembrando da distribuição linear de tensões dada na equação (II.27)):

$$M_x = \int_{-t/2}^{t/2} \sigma_x \cdot z \, dz = \frac{-E \cdot t^3}{12 \cdot (1-\nu^2)} \cdot \left[\frac{\partial \theta_{xz}(x,y)}{\partial x} + \nu \frac{\partial \theta_{yz}(x,y)}{\partial y} \right] \tag{II.29}$$

$$M_y = \int_{-t/2}^{t/2} \sigma_y \cdot z \, dz = \frac{-E \cdot t^3}{12 \cdot (1-\nu^2)} \cdot \left[\nu \frac{\partial \theta_{xz}(x,y)}{\partial x} + \frac{\partial \theta_{yz}(x,y)}{\partial y} \right] \tag{II.30}$$

$$M_{xy} = \int_{-t/2}^{t/2} \tau_{xy} \cdot z \, dz = \frac{-E \cdot t^3 \cdot (1-\nu)}{24 \cdot (1-\nu^2)} \cdot \left[\frac{\partial \theta_{xz}(x,y)}{\partial y} + \frac{\partial \theta_{yz}(x,y)}{\partial x} \right] \tag{II.31}$$

Com o auxílio das matrizes de elasticidade D_b, é possível explicitar os esforços de flexo-torção da seguinte forma:

$$\mathbf{D_b} = \frac{E \cdot t^3}{12 \cdot (1-\nu^2)} \cdot \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{(1-\nu)}{2} \end{bmatrix} \tag{II.32}$$

$$\mathbf{M} = \begin{bmatrix} M_x \\ M_y \\ M_{xy} \end{bmatrix} = -[D_b] \cdot [\kappa_b] \tag{II.33}$$

II.2.4.2.Esforços Cortantes

De acordo com a distribuição uniforme de tensões cisalhantes dada em (II.28), para obter os esforços cortantes basta integrar ao longo da espessura, ou apenas multiplicá-la pelas tensões cisalhantes, conforme mostrado a seguir.

$$Q_x = \int_{-t/2}^{t/2} \tau_{xz} \, dz = G \cdot t \cdot \left(-\theta_{xz}(x,y) + \frac{\partial w(x,y)}{\partial x} \right) \tag{II.34}$$

$$Q_y = \int_{-t/2}^{t/2} \tau_{yz} \, dz = G \cdot t \cdot \left(-\theta_{yz}(x,y) + \frac{\partial w(x,y)}{\partial y} \right) \tag{II.35}$$

Usando novamente o auxílio da matriz de elasticidade, encontramos os esforços na seguinte forma matricial.

$$\mathbf{D_s} = \frac{E \cdot t}{2 \cdot (1+\nu)} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = G \cdot t \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{II.36}$$

$$\mathbf{Q} = \begin{bmatrix} Q_x \\ Q_y \end{bmatrix} = [D_s] \cdot [\varepsilon_s] \tag{II.37}$$

II.2.4.3.Fator de cisalhamento

No cálculo dos esforços de cisalhamento é comum se utilizar o fator de cisalhamento para adequar os esforços calculados através das hipóteses simplificadoras à realidade.

Segundo o desenvolvimento teórico de *Mindlin,* a distribuição das tensões cisalhantes τ_{xz} e τ_{yz} é uniforme (visto que as distorções são lineares ao longo da espessura, como propôs *Mindlin*). Sabe-se, no entanto, que isto não corresponde à realidade, pois na prática esta distribuição assumiria uma forma aproximadamente parabólica.

Existem várias formas de se obter diversos fatores de cisalhamento para impor correções aos conhecidos erros teóricos. Um valor muito tomado na prática, que tem base na equivalência de energia de deformação para uma seção retangular, é o valor de 5/6. Portanto, será proposta essa correção da seguinte forma:

$$k = \frac{5}{6} \tag{II.38}$$

$$\mathbf{D_s} = k \cdot G \cdot t \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{II.39}$$

II.2.5.Equações de Equilíbrio

As equações de equilíbrio podem ser obtidas através do equilíbrio de um elemento infinitesimal de placa, sendo iguais às equações encontradas pela teoria clássica de flexão de placas (*Kirchhoff*).

$$\frac{\partial Q_x}{\partial x} + \frac{\partial Q_y}{\partial y} + q = 0 \quad \text{(II.40)}$$

$$\frac{\partial M_{xy}}{\partial x} + \frac{\partial M_y}{\partial y} - Q_y = 0 \quad \text{(II.41)}$$

$$\frac{\partial M_x}{\partial x} + \frac{\partial M_{xy}}{\partial y} - Q_x = 0 \quad \text{(II.42)}$$

Substituindo as equações (II.29 a II.31) e (II.34 e II.35) em (II.40 a II.42), obtém-se:

$$-D \cdot \left(\frac{\partial^2 \theta_{xz}}{\partial x^2} + \frac{1-\nu}{2} \cdot \frac{\partial^2 \theta_{xz}}{\partial y^2} + \frac{1+\nu}{2} \cdot \frac{\partial^2 \theta_{yz}}{\partial x \partial y}\right) - k \cdot G \cdot \left(\frac{\partial w}{\partial x} - \theta_{xz}\right) = 0 \quad \text{(II.43)}$$

$$-D \cdot \left(\frac{\partial^2 \theta_{yz}}{\partial y^2} + \frac{1-\nu}{2} \cdot \frac{\partial^2 \theta_{yz}}{\partial x^2} + \frac{1+\nu}{2} \cdot \frac{\partial^2 \theta_{xz}}{\partial x \partial y}\right) - k \cdot G \cdot \left(\frac{\partial w}{\partial y} - \theta_{yz}\right) = 0 \quad \text{(II.44)}$$

$$k \cdot G \cdot \left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} - \frac{\partial \theta_{xz}}{\partial x} - \frac{\partial \theta_{yz}}{\partial y}\right) - q = 0 \quad \text{(II.45)}$$

II.2.6.Condições de Contorno

O sistema de equações diferenciais mostrado anteriormente pode ser resolvido impondo-se três condições de contorno por bordo da placa em estudo. A seguir serão descritos como deverão ser impostas tais condições para alguns casos de condições de contorno típicas.

II.2.6.1.Bordo simplesmente apoiado

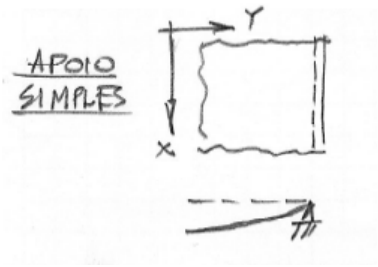


Figura II.6 – Representação de apoio simples (Fonte: VALERIANO [20])

O bordo simplesmente apoiado possui duas condições de contorno essenciais e uma condição de contorno natural que devem ser aplicadas conforme a Tabela 1.

Tabela 1 – Identificação das condições de contorno para apoio simples.

Condições de contorno essenciais	$w = 0$
	$\theta_{xz} = 0$
Condições de contorno naturais	$M_y = 0$

II.2.6.2.Bordo engastado

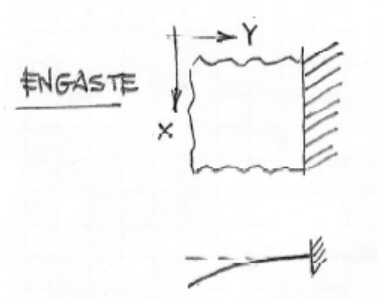


Figura II.7 – Representação de engaste (Fonte: VALERIANO [20])

O bordo engastado possui três condições de contorno essenciais que devem ser aplicadas conforme a Tabela 2.

Tabela 2 – Identificação das condições de contorno para engaste.

Condições de contorno essenciais	$w = 0$
	$\theta_{xz} = 0$
	$\theta_{yz} = 0$

II.2.6.3.Bordo livre

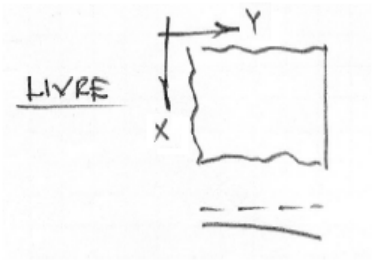


Figura II.8 – Representação de bordo livre (Fonte: VALERIANO [20])

O bordo livre possui três condições de contorno naturais que devem ser aplicadas conforme a Tabela 3.

Tabela 3 – Identificação das condições de contorno para bordo livre.

Condições de contorno naturais	$Q_y = 0$
	$M_y = 0$
	$M_{yx} = 0$

CAPÍTULO III

MÉTODO DOS ELEMENTOS FINITOS

III.1. Introdução

Segundo *Cook*[5] o método dos elementos finitos é um procedimento numérico para analisar problemas tratados pela mecânica do meio contínuo. Este método foi inicialmente desenvolvido por engenheiros, que aplicaram o sentimento físico de um problema estrutural e, posteriormente, foi desenvolvido por matemáticos através de métodos mais abstratos, estendendo sua aplicação a diversos problemas. Em todas as situações o analista quer calcular um determinado campo, que na análise estrutural é o campo de deslocamentos ou o campo de tensões; na análise térmica é o campo de temperaturas; em problemas de mecânica dos fluidos o campo é dado pelas velocidades do fluido e assim por diante. Portanto, o método dos elementos finitos é uma ferramenta numérica para se obter de forma aproximada, ou até algumas vezes exata, a solução de um problema regido por equações diferenciais.

De outra forma, pode-se dizer que o método dos elementos finitos divide o domínio em vários subdomínios ou elementos, conforme mostra a Figura III.1, descrevendo o comportamento de cada elemento de uma forma simplificada. Posteriormente esses elementos são reagrupados, fazendo-se a conexão dos elementos através de nós. Este procedimento resulta em um sistema de equações algébricas (que na análise de tensões são equações de equilíbrio) que, quando se torna muito grande sua resolução fica impraticável caso não seja utilizada uma implementação computacional.

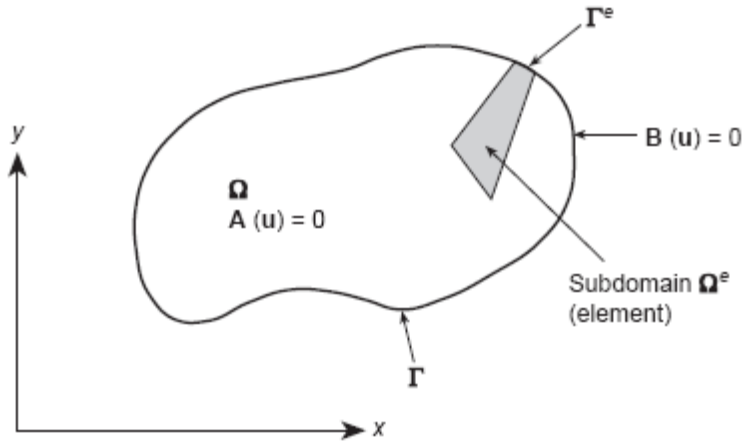


Figura III.1 – Domínio do problema Ω e contorno Γ (Fonte: Zienkiewicz[1]).

Após a resolução do sistema linear com obtenção dos deslocamentos nodais, existe ainda uma etapa de pós-processamento para cálculo das tensões, deformações, reações de apoio, etc.

Neste capítulo será feita uma abordagem sobre cada uma das etapas envolvidas em um problema de elasticidade linear típico.

III.2.Características do método dos elementos finitos

Os problemas analisados no método dos elementos finitos são regidos por equações diferenciais, que podem ser solucionadas impondo-se condições de contorno no domínio estudado. No entanto, tratar matematicamente esses problemas é bem mais complexo que tratar um problema de valor inicial, e soluções fechadas são, em geral, mais difíceis de serem encontradas ou às vezes nem existem, portanto, se faz necessária a adoção de métodos aproximados.

Para exemplificar o descrito anteriormente, pode ser tomado como exemplo o caso de uma barra engastada-livre, axialmente carregada com carga linearmente distribuída e comprimento L, conforme mostrado na Figura III.2.

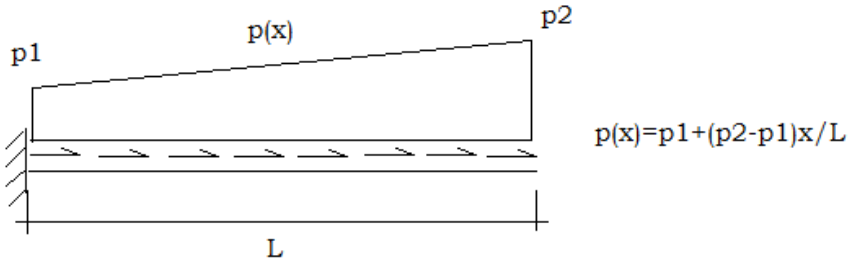


Figura III.2 –Representação esquemática de barra solicitada axialmente.

Este problema é regido pela seguinte equação diferencial:

$$E \cdot A \cdot \frac{d^2u}{dx^2} + p(x) = 0 \qquad \text{(III.1)}$$

Este problema é um dos mais simples dentro da teoria da elasticidade linear, tendo solução analítica relativamente fácil, obtida através de integrações sucessivas, impondo-se condições de contorno $u(0) = 0$ e $\frac{d}{dx}u(L) = 0$, dada na seguinte forma:

$$u(x) = \frac{-1}{E \cdot A} \cdot \left[\frac{-L \cdot (p1+p2)}{2} \cdot x + \frac{p1 \cdot x^2}{2} + \frac{p2-p1}{L} \cdot \frac{x^3}{6}\right] \qquad \text{(III.2)}$$

A solução é um polinômio do terceiro grau, pois a carga é linearmente distribuída.

Supondo que se queira obter uma solução aproximada para a equação diferencial (III.1) no domínio L. O método dos elementos finitos propõe soluções aproximadas onde o campo de deslocamentos será dado por combinação linear de funções, conforme a seguir:

$$u \cong \tilde{u} = \sum_{i=1}^{ngdl} N_i \cdot d_i \qquad \text{(III.3)}$$

Onde $ngdl$ é o número de graus de liberdade do elemento; a_i corresponde ao deslocamento no grau de liberdade i; N_i são funções que interpolam os valores um no grau de liberdade i, e zero em todos os outros, conhecidas como funções de forma.

Desta forma, se for adotado um elemento ligado por dois nós para discretizar o domínio do problema anterior, os deslocamentos no domínio do elemento poderiam ser representados de forma aproximada, por uma combinação linear das funções de forma representadas na Figura III.3.

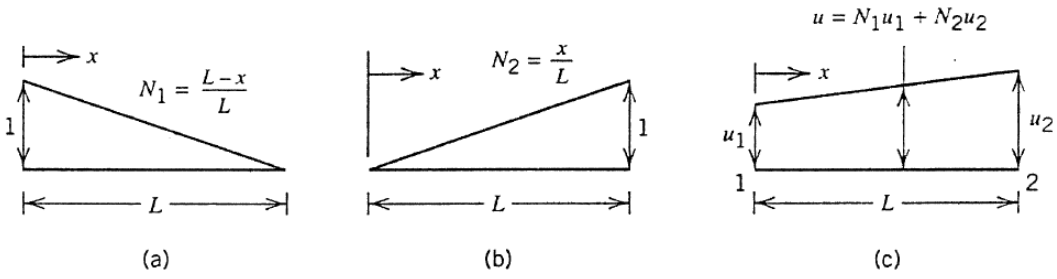


Figura III.3 –(a,b)Representação das funções de forma N1 e N2 do elemento de barra de dois nós, (c) Interpolação linear dos deslocamentos axiais (Fonte: Cook[4])

A seguir é mostrada uma solução deste problema dividindo o domínio L em um elemento.

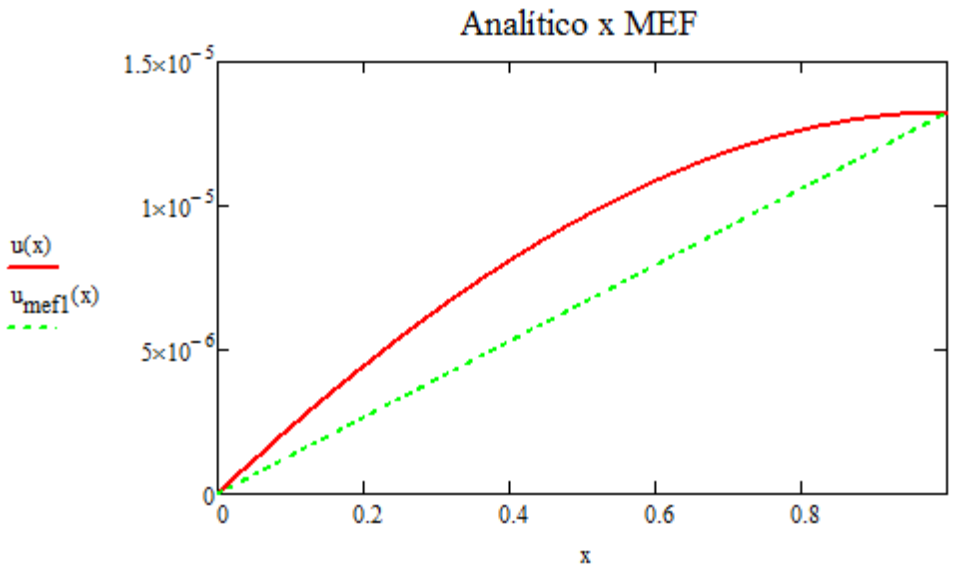


Figura III.4 –Deslocamentos axiais via MEF com um elemento vs. analítico.

Pode ser notado que a solução através do método dos elementos finitos coincidiu com a solução exata apenas nos pontos nodais, contendo resíduo no

campo de deslocamentos em todo o domínio do problema. Vale ressaltar que a solução nodal exata é um caso particular para este tipo de problema, e não é sempre que isso ocorre na aplicação do método dos elementos finitos.

Com o intuito de analisar a convergência da solução aproximada para a solução analítica, o problema anterior foi analisado com cinco pontos nodais ligados por quatro elementos. O resultado pode ser visto na Figura III.5.

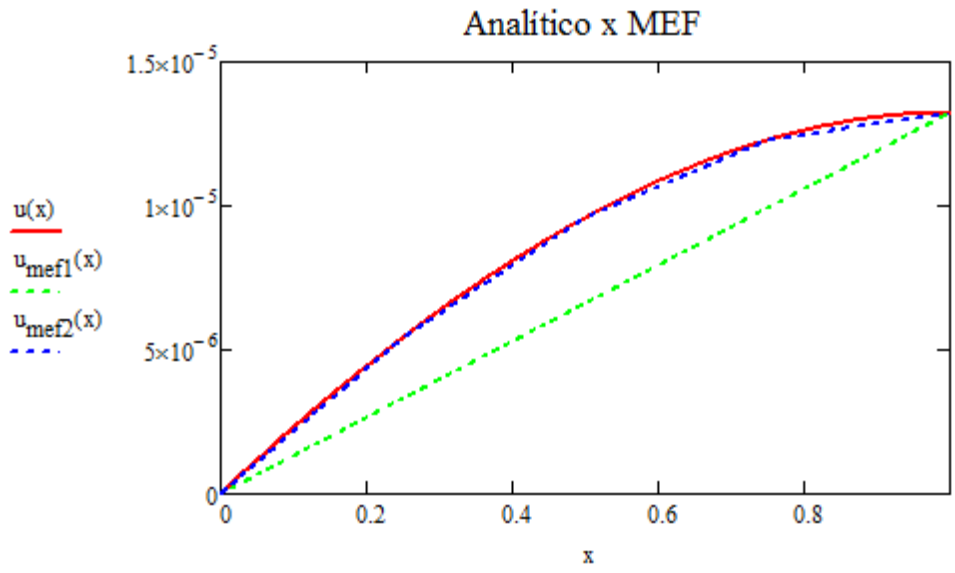


Figura III.5 – Deslocamentos axiais via MEF com um e quatro elementos vs. analítico.

O fato da solução aproximada ter apresentado menor erro com a maior discretização do problema é um dos pontos mais importantes no método dos elementos finitos. Ou seja, conforme o problema vai ficando mais refinado, a solução aproximada vai convergindo para a solução verdadeira do problema.

III.3.Elemento estrutural

O problema elástico dado no espaço tridimensional, sempre que possível, é transformado, por simplificação, em um problema bi ou unidimensional (ex: viga, placa, problema de estado plano de tensão ou deformação, etc.). Este

procedimento foi aplicado no capítulo anterior, onde foram integradas as tensões na placa e usaram-se esforços no desenvolvimento das expressões.

Portanto, assim fica definido o meio elástico que será estudado e são montadas as equações diferenciais para a representação do mesmo. Esta equação diferencial tem como solução os campos de deslocamentos envolvidos na análise, e para ser resolvida devem ser fornecidas as condições de contorno do problema. A representação aproximada desses deslocamentos é feita conforme descrito a seguir.

III.3.1.Funções de deslocamentos

No método dos elementos finitos o campo de deslocamentos no domínio dos elementos é dado por funções que interpolam os deslocamentos nodais da estrutura, devendo estas atender às condições de continuidade entre elementos que dependerão do problema estudado.

Portanto, dentro de um elemento o campo de deslocamentos pode ser definido pelos deslocamentos nodais do mesmo, conforme mostrado a seguir, onde $\boldsymbol{u}$ corresponde ao vetor com todos os campos de deslocamentos, d_i corresponde ao deslocamento de cada grau de liberdade do elemento (ou o valor do campo nodal de forma mais genérica), e N_i corresponde à função de interpolação associada ao grau de liberdade i, valendo um no mesmo e zero nos demais.

$$\mathbf{u} \cong \tilde{\mathbf{u}} = \sum_{i=1}^{ngdl} N_i \cdot d_i = \mathbf{N} \cdot \mathbf{d} \qquad \text{(III.4)}$$

Onde a função interpoladora (ou função de forma) terá um número de termos igual ao número de graus de liberdade do elemento, para que se tenha um sistema com solução única.

Conforme foi dito anteriormente, o mesmo vale ao comportamento global, ou seja, um elemento pode representar bem a variação do campo em determinada região, mas em regiões com grandes variações no campo pode ser

necessária uma malha mais refinada de elementos finitos, a fim de aumentar a precisão da resposta.

III.3.2.Deformações

Com os deslocamentos definidos em todos os pontos dentro dos elementos, pode ser usada a teoria da elasticidade para representar as deformações em cada ponto da estrutura, que pode ser escrita da seguinte forma, em notação matricial.

$$\boldsymbol{\varepsilon} = \mathbf{S} \cdot \tilde{\mathbf{u}} \tag{III.5}$$

Onde **S** é um operador diferencial.

Será utilizada uma notação para relacionar os deslocamentos nodais com o vetor de deformações, que é chamada de matriz de compatibilidade cinemática, definida conforme a seguir.

$$\mathbf{B} = \mathbf{S} \cdot \mathbf{N} \tag{III.6}$$

Substituindo-se a expressão (III.4) na equação (III.5) e usando-se a expressão (III.6), obtém-se:

$$\boldsymbol{\varepsilon} = \mathbf{B} \cdot \mathbf{d} \tag{III.7}$$

III.3.3.Tensões

As tensões podem ser encontradas com base nas equações constitutivas (ou relações tensão-deformação) que regem o problema. Sendo assim, têm-se as deformações em função das tensões a partir da matriz de elasticidade C, ou seja:

$$\boldsymbol{\varepsilon} = \mathbf{C} \cdot \boldsymbol{\sigma} \tag{III.8}$$

Invertendo-se a matriz constitutiva **C**, obtém-se as tensões expressas em relação às deformações da seguinte forma

$$\boldsymbol{\sigma} = \mathbf{D} \cdot \boldsymbol{\varepsilon} \tag{III.9}$$

Logo, é possível expressar as tensões em função do vetor de deslocamentos nodais da seguinte forma:

$$\boldsymbol{\sigma} = \mathbf{D} \cdot \mathbf{B} \cdot \mathbf{d} \tag{III.10}$$

III.4.Elementos isoparamétricos

A formulação de elementos isoparamétricos permite a extensão da formulação para elementos com geometria mais complexa. Por exemplo, um elemento quadrilátero plano poderá ser não retangular, permitindo a sua distorção e até a inclusão de lados curvos (no caso de elementos quadráticos ou de ordem superior).

O ponto mais importante na representação deste tipo de elemento é que na sua formulação são transferidas as coordenadas físicas do elemento para um sistema de coordenadas naturais. Portanto, neste domínio fica muito mais simples a avaliação das integrais numéricas, sendo depois necessária uma transformação inversa para o domínio original do problema. A matriz de transformação de coordenadas é a matriz *Jacobiana*.

Na construção da matriz de rigidez dos elementos devem ser encontradas as deformações do mesmo, que são obtidas através das derivadas do deslocamento com relação às coordenadas físicas do problema. Por sua vez, os deslocamentos agora serão representados nas coordenadas naturais ξ e η. Portanto, para transformar as derivadas paramétricas em derivadas cartesianas, é necessário o uso da regra da cadeia mostrado a seguir.

$$\frac{\partial N_i}{\partial \xi} = \frac{\partial x}{\partial \xi}\frac{\partial N_i}{\partial x} + \frac{\partial y}{\partial \xi}\frac{\partial N_i}{\partial y} \tag{III.11}$$

$$\frac{\partial N_i}{\partial \eta} = \frac{\partial x}{\partial \eta}\frac{\partial N_i}{\partial x} + \frac{\partial y}{\partial \eta}\frac{\partial N_i}{\partial y} \tag{III.12}$$

Que pode ser representada matricialmente por:

$$\begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} \tag{III.13}$$

Onde a matriz acima representada é denotada como matriz *Jacobiana*.

Se for desejada a obtenção das derivadas cartesianas em função das coordenadas naturais, basta inverter-se a matriz *Jacobiana*. É perfeitamente possível que sejam fornecidas coordenadas físicas inadmissíveis que condicionem mal a matriz *Jacobiana* e a tornem singular. Esta inversa é sensível a certos tipos de distorção e também às posições dos nós laterais. Segundo *Gallagher*[6], em elementos bi-quadráticos os nós laterais são melhores posicionados no meio dos nós adjacentes.

A implicação que a representação isoparamétrica produzirá na construção da matriz de rigidez é que as deformações agora serão expressas em termos das coordenadas naturais, e a própria integração também será realizada em coordenadas naturais, utilizando-se um fator de escala igual ao determinante da matriz *Jacobiana*, conforme a seguir.

$$dx\, dy = |\boldsymbol{J}|\, d\xi\, d\eta \tag{III.14}$$

Aplicando-se esta técnica à formulação típica de elementos planos, a obtenção da matriz de rigidez elementar pode ser feita conforme a seguir:

$$\mathbf{k_{elem}} = \int_A \mathbf{B}^{\mathrm{T}} \cdot \mathbf{D} \cdot \mathbf{B}\, dA = \int_{-1}^{1}\int_{-1}^{1} \mathbf{B}^{\mathrm{T}} \cdot \mathbf{D} \cdot \mathbf{B} \cdot |\boldsymbol{J}|\, d\xi\, d\eta \tag{III.15}$$

Estas idéias de formulação isoparamétrica podem ser estendidas naturalmente para problemas uni ou tri dimensionais (cubos).

Existem ainda, na literatura, elementos isoparamétricos com formas de triângulos e tetraedros. Estes elementos são muito úteis na geração de malhas

complexas, onde um quadrilátero ou um cubo não consegue ocupar determinadas áreas ou volumes. Para estes elementos a formulação é um pouco distinta. Este tipo de elemento não será abordado aqui neste texto, mas existe uma vasta literatura em elementos finitos tratando do assunto. No decorrer desta seção serão discutidos alguns tipos de elementos planos, mas antes serão abordados alguns aspectos de completividade polinomial.

III.4.1.Completividade polinomial

Para representar bem o comportamento de um determinado fenômeno no interior do elemento, o polinômio interpolador deve conter termos suficientemente ricos para garantir a convergência do método dos elementos finitos. De acordo com *Zienkiewicz*[1], de uma forma geral se procura obter expansões de elementos que disponham da ordem de um polinômio mais completo para o mínimo de graus de liberdade. Neste contexto é usual a análise do triângulo de *Pascal*, onde o número de termos que ocorrem no polinômio de duas variáveis pode ser tirado diretamente.

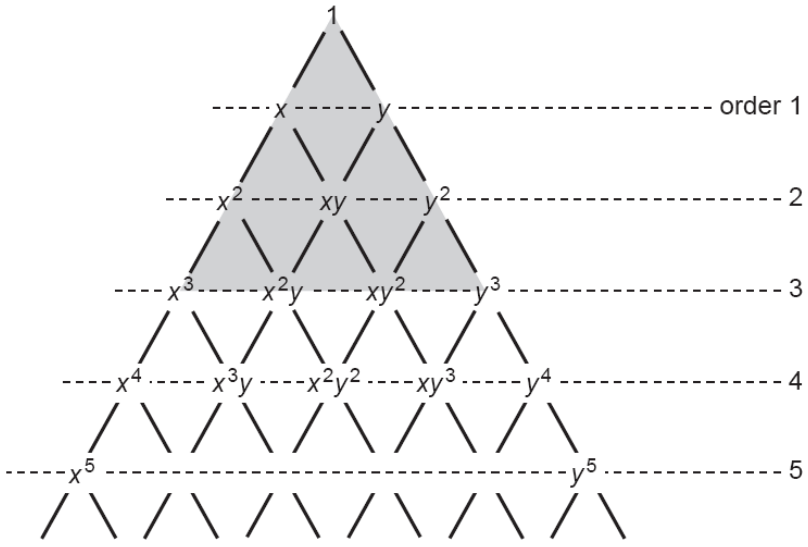


Figura III.6 – Representação do triângulo de Pascal (Fonte: Zienkiewicz[1]).

Cada linha do triângulo mostrado acima corresponde a uma ordem do polinômio interpolador. Portanto, polinômios completos de primeira ordem exigem três termos, de segunda ordem, seis, e assim por diante.

III.4.2.Elemento isoparamétrico plano Bi-linear

O procedimento descrito nesta seção generaliza a aplicação de qualquer forma de quadrilátero com quatro nós (com requerimento de continuidade C_0), dado que este é submetido a uma transformação de coordenadas e é transformado em um retângulo sem distorções em espaço parametrizado.

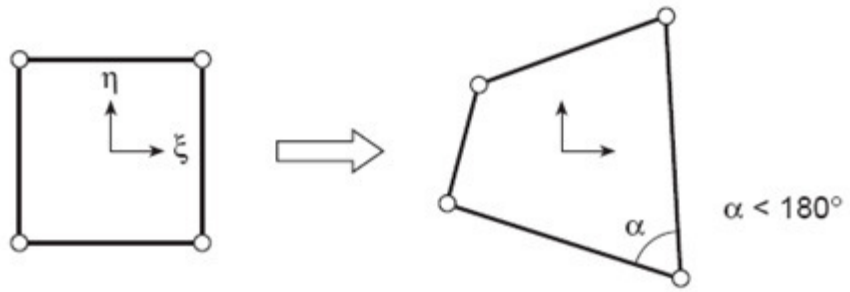


Figura III.7 – Representação de elemento Bi-linear em domínio parametrizado e original.

Este tipo de elemento faz parte da família de elementos de *Lagrange*. Nesta família as funções de forma podem ser obtidas através do produto dos polinômios de *Lagrange* em cada direção. Sendo assim, as funções de forma podem ser dadas por:

$$N_i = \frac{1}{4} \cdot (1 + \xi_i\, \xi) \cdot (1 + \eta_i\, \eta)\,; \quad i = 1 \ldots 4 \tag{III.16}$$

Que impondo as condições de interpolação de *Lagrange* nos pontos $\xi = \pm 1$ e $\eta = \pm 1$ podem ser escritas por:

$$N_1 = \frac{1}{4} \cdot (1 - \xi) \cdot (1 - \eta) \tag{III.17}$$

$$N_2 = \frac{1}{4} \cdot (1 + \xi) \cdot (1 - \eta) \tag{III.18}$$

$$N_3 = \frac{1}{4} \cdot (1 + \xi) \cdot (1 + \eta) \tag{III.19}$$

$$N_4 = \frac{1}{4} \cdot (1 - \xi) \cdot (1 + \eta) \tag{III.20}$$

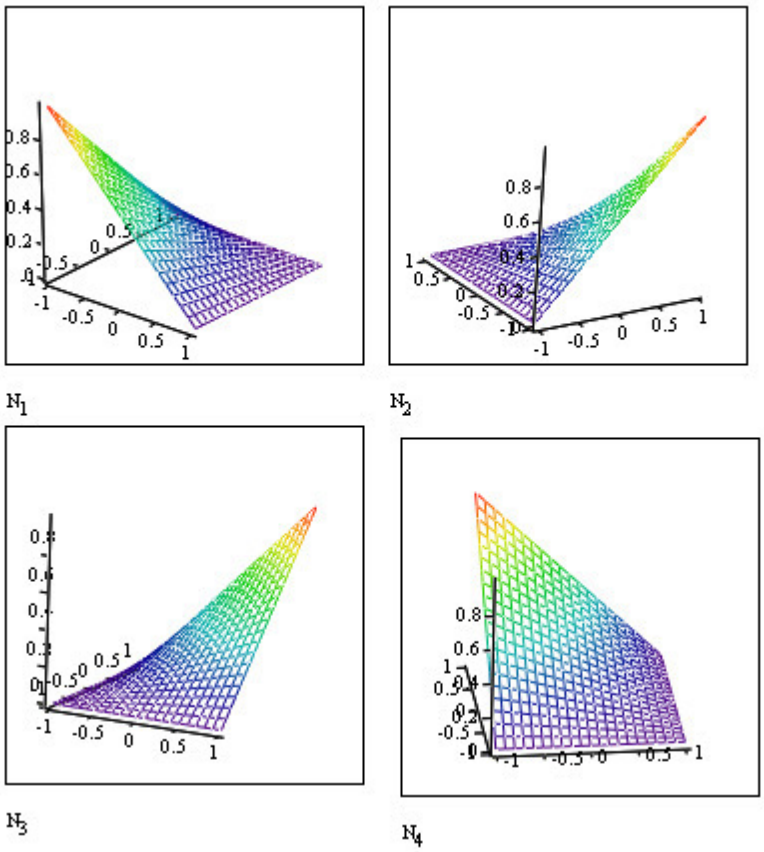


Figura III.8 – Funções de forma N_1, N_2, N_3 e N_4 do elementos Bi-linear.

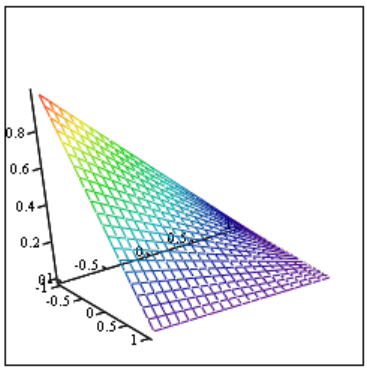
Estas funções interpolam o valor um no grau de liberdade i e zero em todos os outros, conforme mostra a Figura III.8. A partir do triângulo de pascal, é possível perceber que os termos presentes na função de forma são os termos de primeira ordem e um termo cruzado xy de ordem 2. Os termos de ordem um iriam definir um campo de deslocamentos plano, porém, com a adição de um termo cruzado, o elemento adquire uma forma diferenciada (mostrada na Figura III.8). Este ponto é muito importante, pois as deformações são expressas como derivadas das funções de forma e, portanto, para elementos onde a deformação é

dada pela primeira derivada da função de forma com relação à x ou h, esta apresentará uma variação na outra direção.

III.4.3.Elementos isoparamétricos planos quadráticos

Os elementos quadráticos podem pertencer à família de elementos de *Lagrange*, à família de elementos de *Serendipity* ou fazer parte de uma formulação híbrida. O elemento pertencente à família de *Lagrange* pode ter funções de forma obtidas através de interpolação polinomial de *Lagrange*. A geração de elementos segundo essa concepção traz a necessidade de os nós serem posicionados no interior do elemento, porém, os elementos da família de *Serendipity* suprem essa necessidade.

Será apresentada agora uma maneira sistemática de se obter elementos quadráticos planos, muito bem descrita por *Cook*[5]. Esta maneira está simplesmente baseada no acréscimo de alguma função que irá corresponder ao acréscimo de algum grau de liberdade no elemento. Sendo assim, considere a função de forma N_1 do elemento Bi-linear descrito anteriormente.



N1

N1(−1,1) = 0 N1(0,1) = 0 N1(1,1) = 0

N1(−1,0) = 0.5 N1(0,0) = 0.25 N1(1,0) = 0

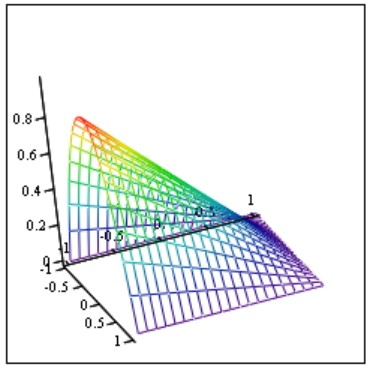
N1(−1,−1) = 1 N1(0,−1) = 0.5 N1(1,−1) = 0

Figura III.9 – Função de forma N_1 do elemento Bi-linear e avaliação nos pontos do domínio parametrizado.

O elemento Bi-linear possui quatro nós com coordenadas nodais combinadas nos pontos $\xi = \pm 1$ e $\eta = \pm 1$. Posicionando os nós laterais nas

coordenadas $\xi = 0$ e $\eta = 0$ é possível representar um lado curvo de um elemento quadrático. A geração do elemento com essas características pode ser feita facilmente por inspeção. As funções i do elemento Bi-linear valem um no grau de liberdade i e zero nos três demais. Portanto, devido ao fato da variação ser linear ao longo dos lados, essa função valerá 0.5 na coordenada $\xi = 0$ ou $\eta = 0$ correspondente àquele lado, conforme mostra a Figura III.9.

Se forem introduzidas agora funções que valham um no nó lateral associado e zero em todos os demais graus de liberdade, podemos, além de obter a função associada àquele grau de liberdade, introduzir as devidas correções nas funções correspondentes ao elemento Bi-linear. A seguir, são mostradas as funções que correspondem às funções N_5 e N_8 do elemento quadrático de oito nós.

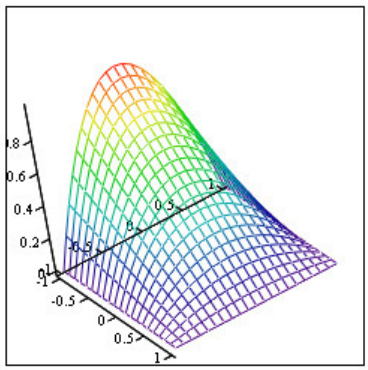


N5(−1,1) = 0 N5(0,1) = 0 N5(1,1) = 0

N5(−1,0) = 0 N5(0,0) = 0.5 N5(1,0) = 0

N5(−1,−1) = 0 N5(0,−1) = 1 N5(1,−1) = 0

N5



N8(−1,1) = 0 N8(0,1) = 0 N8(1,1) = 0

N8(−1,0) = 1 N8(0,0) = 0.5 N8(1,0) = 0

N8(−1,−1) = 0 N8(0,−1) = 0 N8(1,−1) = 0

N8

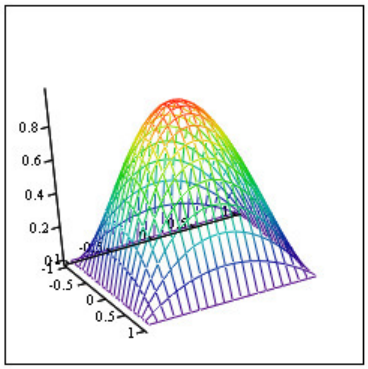
Figura III.10 – Função de forma N_5 e N_8 de elemento quadrático e avaliação nos pontos do domínio parametrizado.

Portanto, é possível aplicar uma correção de forma que a nova função N1 valha zero em todos os graus de liberdade do novo elemento, da forma mostrada a seguir.

$$N_1 = \frac{1}{4} \cdot (1 - \xi) \cdot (1 - \eta) - \frac{1}{2} \cdot N_5 - \frac{1}{2} \cdot N_8 \qquad \text{(III.21)}$$

O elemento que resultará na presença de oito nós será um elemento da família *Serendipity* que, para elemento quadrático, possui apenas nós no contorno do elemento.

É possível ainda adicionar um nó interno com coordenadas $\xi = 0$ e $\eta = 0$ e chegar a um elemento da família de *Lagrange*. A função de forma correspondente a esse grau de liberdade é mostrada a seguir e é conhecida como "função bolha" (do inglês, "*bubble function"*).



$N9(-1,1) = 0 \quad N9(0,1) = 0 \quad N9(1,1) = 0$

$N9(-1,0) = 0 \quad N9(0,0) = 1 \quad N9(1,0) = 0$

$N9(-1,-1) = 0 \quad N9(0,-1) = 0 \quad N9(1,-1) = 0$

Figura III.11 – Função de forma N_9 para gerar elemento quadrático de Lagrange ("bubble function").

Com a inclusão deste conceito é possível montar a tabela mostrada a seguir, de onde podem ser retirados elementos retangulares com lados quadráticos e com números de nós podendo variar de quatro a nove.

Tabela 4 – Esquema de introdução de nós em elemento Bi-linear.

	Aplicar coluna se o nó i estiver presente no elemento				
	i = 5	i = 6	i = 7	i = 8	i = 9
$N_1 = 1/4 \,.\, (1 - \xi)(1 - \eta)$	$-1/2 \,.\, N_5$			$-1/2 \,.\, N_8$	$+1/4 \,.\, N_9$
$N_2 = 1/4 \,.\, (1 + \xi)(1 - \eta)$	$-1/2 \,.\, N_5$	$-1/2 \,.\, N_6$			$+1/4 \,.\, N_9$
$N_3 = 1/4 \,.\, (1 + \xi)(1 + \eta)$		$-1/2 \,.\, N_6$	$-1/2 \,.\, N_7$		$+1/4 \,.\, N_9$
$N_4 = 1/4 \,.\, (1 - \xi)(1 + \eta)$			$-1/2 \,.\, N_7$	$-1/2 \,.\, N_8$	$+1/4 \,.\, N_9$
$N_5 = 1/2 \,.\, (1 - \xi^2)(1 - \eta)$					$-1/2 \,.\, N_9$
$N_6 = 1/2 \,.\, (1 + \xi)(1 - \eta^2)$					$-1/2 \,.\, N_9$
$N_7 = 1/2 \,.\, (1 - \xi^2)(1 + \eta)$					$-1/2 \,.\, N_9$
$N_8 = 1/2 \,.\, (1 - \xi)(1 - \eta^2)$					$-1/2 \,.\, N_9$
$N_9 = \quad (1 - \xi^2)(1 - \eta^2)$					

Através da tabela acima é possível ainda formular elementos híbridos que contenham lados com nós e sem nós, que são especialmente úteis quando é feita uma transição de malha com elementos quadráticos para uma malha com elementos lineares, como mostrados a seguir.

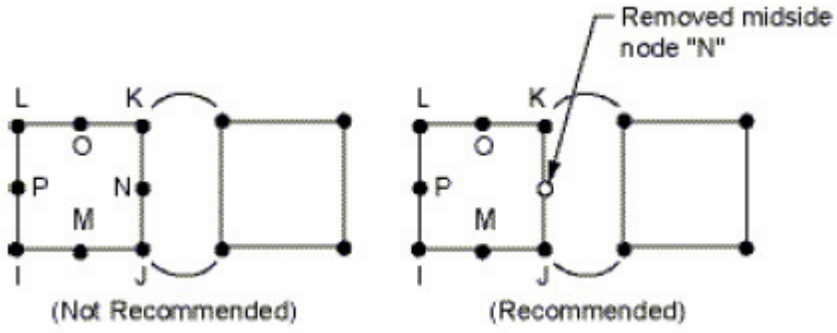


Figura III.12 – Esquema de conexão de elemento quadrático híbrido com elemento linear (Fonte: ANSYS – Reference Guide[37])

Usando o mesmo raciocínio, a formulação pode ser ainda estendida para elementos cúbicos e até mesmo de ordem superior.

CAPÍTULO IV

MÉTODO DOS ELEMENTOS FINITOS APLICADO À TEORIA DE PLACAS ESPESSAS

IV.1.Introdução

Segundo *Zienkiewicz*[2], as aplicações do método dos elementos finitos à teoria de placas e suas extensões para cascas, foram um dos primeiros assuntos de aplicação, e começaram a ser estudadas antes mesmo de 1960. Naquele tempo, várias dificuldades foram encontradas e estas aplicações não foram completamente estudadas, logo, este tópico ainda é um assunto de pesquisa.

Os assuntos básicos de teoria de placas já foram discutidos no segundo capítulo, onde foi visto que na teoria clássica de *Kirchhoff* (placas delgadas) é possível representar o estado de deformações em função de apenas um campo, que representa os deslocamentos fora do plano da placa. No entanto, desenvolvendo-se as expressões de deformações, são encontradas derivadas de segunda ordem do campo de deslocamentos. Tal fato impõe uma exigência de continuidade não só no campo de deslocamentos, como também nas suas derivadas (continuidade C_1). Logo, é necessário impor continuidade de deslocamentos e rotações nas interfaces dos nós dos elementos. A determinação das funções de forma com esse tipo de requerimento é bem mais complexa que aquelas que exigem continuidade do tipo C_0 (somente no campo).

Segundo a teoria de placas espessas de *Mindlin*, o estado de deformações é representado em função de três campos de deslocamento, conforme visto no segundo capítulo. Portanto, no desenvolvimento das expressões de deformações, são encontradas apenas derivadas de primeira ordem, fazendo com que o problema exija apenas uma continuidade do tipo C_0. Desta forma, pode-se dizer que, em termos de implementação de elementos finitos, é mais simples utilizar a

teoria de placas espessas do que a teoria clássica, ao contrário das soluções analíticas fechadas.

Neste capítulo será apresentada a formulação de elementos finitos de placa espessa, além da apresentação das expressões utilizadas para a geração das matrizes de rigidez em elementos retangulares isoparamétricos.

IV.2.Formulação de Elementos Finitos de Placa Espessa

Conhecendo-se as equações geométricas e constitutivas do problema, é possível ser feita uma abordagem através dos princípios de energia. Para solução analítica a partir desta formulação, devem ser aplicados conceitos do Cálculo Variacional que, desenvolvido adequadamente, indicará a equação que governa o problema. No entanto, obtida a expressão da energia potencial total do sistema, é possível obter-se soluções aproximadas, desde que seja admitida uma configuração cinematicamente admissível que extremize a energia potencial total. Este princípio é muito usado como formulação variacional no método dos elementos finitos.

A energia total do sistema (Π) é dada pela soma das parcelas de energia de deformação (U), geradas pela configuração deformada da estrutura, e da parcela de energia potencial gerada pelas forças externas (W).

$$\Pi = U - W \tag{IV.1}$$

A energia potencial gerada pelas forças externas concentradas é fornecido pela expressão a seguir:

$$W = \mathbf{F} \cdot \mathbf{u} \; ; \tag{IV.2}$$

Sendo **F** a força e **u** deslocamento.

No entanto, nos problemas de elasticidade, além de forças concentradas, é comum se trabalhar com forças distribuídas por unidade de comprimento, área

ou volume. Nesses casos, a energia potencial gerada pelas forças externas pode ser dada por:

$$W = \int \mathbf{q} \cdot \mathbf{u}(x)\ dx\ ; \tag{IV.3}$$

Sendo q = força / unidade de comprimento;

$$W = \iint \mathbf{q} \cdot \mathbf{u}(x,y)\ dA\ ; \tag{IV.4}$$

Sendo q = força / unidade de área (A);

$$W = \iiint \mathbf{q} \cdot \mathbf{u}(x,y,z) dV\ ; \tag{IV.5}$$

Sendo q = força / unidade de volume (V)

A energia de deformação específica pode ser dada por:

$$U^* = \int \sigma\, d\varepsilon \tag{IV.6}$$

Para materiais que obedecem à relação linear entre tensão e deformação, a energia de deformação do sistema pode ser dada por:

$$U = \int_V\ U^*\, dV = \frac{1}{2}\int_V\ \sigma \cdot \varepsilon\, dV \tag{IV.7}$$

Aplicando-se esta teoria ao problema estudado, considerando apenas as deformações não nulas, tem-se:

$$U = \frac{1}{2}\int_V\ (\sigma_x \cdot \varepsilon_x + \sigma_y \cdot \varepsilon_y + \tau_{xy} \cdot \gamma_{xy} + \tau_{xz} \cdot \gamma_{xz} + \tau_{yz} \cdot \gamma_{yz})\, dV \tag{IV.8}$$

A expressão (IV.8) pode ser escrita também através da seguinte notação matricial:

$$U = \frac{1}{2}\int_V\ (\boldsymbol{\sigma}_\mathbf{b} \cdot \boldsymbol{\varepsilon}_\mathbf{b} + \boldsymbol{\sigma}_\mathbf{s} \cdot \boldsymbol{\varepsilon}_\mathbf{s})\, dV \tag{IV.9}$$

Usando-se as equações constitutivas para expressar a tensão em função da deformação e integrando-se ao longo da espessura da placa, é possível representar a energia da seguinte forma:

$$U = \frac{1}{2}\int_A\ \boldsymbol{\kappa}_\mathbf{b}^{\mathrm{T}} \cdot \mathbf{D}_\mathbf{b} \cdot \boldsymbol{\kappa}_\mathbf{b} + \boldsymbol{\varepsilon}_\mathbf{s}^{\mathrm{T}} \cdot \mathbf{D}_\mathbf{s} \cdot \boldsymbol{\varepsilon}_\mathbf{s}\, dA \tag{IV.10}$$

Portanto, para o problema de flexão de placas espessas, a energia potencial total pode ser expressa da seguinte forma:

$$\Pi = \frac{1}{2}\int_A \ \boldsymbol{\kappa_b}^{\mathrm{T}} \cdot \mathbf{D_b} \cdot \boldsymbol{\kappa_b} + \boldsymbol{\varepsilon_s}^{\mathrm{T}} \cdot \mathbf{D_s} \cdot \boldsymbol{\varepsilon_s}\, dA - \int_A \ q \cdot w(x,y) dA - \sum F_i w_i \tag{IV.11}$$

A partir do funcional acima, já é possível desenvolver todo o procedimento para a geração das matrizes de rigidez e vetor de forças nodais. O único ponto a ser mudado de um tipo para outro de elemento, são as funções de interpolação e troca de domínio de integração.

IV.3.Elementos Finitos Isoparamétricos para placas Espessas

Conforme já descrito na formulação de elementos isoparamétricos, os mesmos podem fazer parte de diversas famílias e grau de precisão. Neste trabalho foi elaborada uma formulação geral e foram aplicados os seguintes elementos para estudo e desenvolvimento.

i. Elemento de placa espessa Bi-linear (4 nós);

ii. Elemento de placa espessa Quadrático (8 nós – Família *Serendipity*);

iii. Elemento de placa espessa Quadrático (9 nós – Família de *Lagrange*);

Portanto, neste momento serão avaliados os aspectos comuns de todas as formulações.

IV.3.1.Funções de deslocamento

O campo de deslocamentos será representado por três variáveis independentes, e a avaliação desses campos nos domínios do elemento será feita com auxílio das funções de forma, que irão interpolar os valores nodais da

variável independente. Tem-se, portanto, a seguinte equação geral (onde, n corresponde ao número de nós do elemento).

$$\tilde{\mathbf{u}} = \begin{bmatrix} \tilde{w} \\ \tilde{\theta}_{xz} \\ \tilde{\theta}_{yz} \end{bmatrix} = \sum_{i=1}^{n} \begin{bmatrix} N_i & 0 & 0 \\ 0 & N_i & 0 \\ 0 & 0 & N_i \end{bmatrix} \cdot \begin{bmatrix} w_i \\ \theta_{xz_i} \\ \theta_{yz_i} \end{bmatrix} \tag{IV.12}$$

IV.3.2.Deformações e curvaturas

Usando as equações da teoria da elasticidade e as funções de interpolação do elemento, é possível expressar as deformações no interior do mesmo, conforme a seguir.

$$\boldsymbol{\varepsilon}_{\mathbf{b}} = -z \cdot \begin{bmatrix} \frac{\partial \theta_{xz}(x,y)}{\partial x} \\ \frac{\partial \theta_{yz}(x,y)}{\partial y} \\ \left(\frac{\partial \theta_{xz}(x,y)}{\partial y} + \frac{\partial \theta_{yz}(x,y)}{\partial x} \right) \end{bmatrix} \cong -z \cdot \sum_{i=1}^{n} \begin{bmatrix} 0 & \frac{\partial N_i}{\partial x} & 0 \\ 0 & 0 & \frac{\partial N_i}{\partial y} \\ 0 & \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} \end{bmatrix} \cdot \begin{bmatrix} w_i \\ \theta_{xz_i} \\ \theta_{yz_i} \end{bmatrix} \tag{IV.13}$$

Com a matriz de compatibilidade cinemática B, pode-se reescrever as curvaturas da seguinte forma:

$$\mathbf{B}_{\mathbf{b}} = \sum_{i=1}^{n} \begin{bmatrix} 0 & \frac{\partial N_i}{\partial x} & 0 \\ 0 & 0 & \frac{\partial N_i}{\partial y} \\ 0 & \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} \end{bmatrix} \tag{IV.14}$$

$$\boldsymbol{\kappa}_{\mathbf{b}} = \mathbf{B}_{\mathbf{b}} \cdot \begin{bmatrix} w_i \\ \theta_{xz_i} \\ \theta_{yz_i} \end{bmatrix} \tag{IV.15}$$

Realizando o mesmo procedimento para as deformações cisalhantes, podemos escrever:

$$\boldsymbol{\varepsilon}_{\mathbf{s}} = \begin{bmatrix} -\theta_{xz}(x,y) + \frac{\partial w(x,y)}{\partial x} \\ -\theta_{yz}(x,y) + \frac{\partial w(x,y)}{\partial y} \end{bmatrix} \cong \sum_{i=1}^{n} \begin{bmatrix} \frac{\partial N_i}{\partial x} & -N_i & 0 \\ \frac{\partial N_i}{\partial y} & 0 & -N_i \end{bmatrix} \cdot \begin{bmatrix} w_i \\ \theta_{xz_i} \\ \theta_{yz_i} \end{bmatrix} \tag{IV.16}$$

$$\boldsymbol{\varepsilon_s} = \mathbf{B_s} \cdot \begin{bmatrix} w_i \\ \theta_{xz_i} \\ \theta_{yz_i} \end{bmatrix} \tag{IV.17}$$

IV.3.3.Esforços

Utilizando-se as relações anteriores e substituindo-as nas expressões dos esforços apresentadas no segundo capítulo, podemos expressá-los da seguinte forma:

$$\mathbf{M} = \begin{bmatrix} M_x \\ M_y \\ M_{xy} \end{bmatrix} = -\mathbf{D_b} \cdot \boldsymbol{\kappa_b} = -\mathbf{D_b} \cdot \mathbf{B_b} \cdot \begin{bmatrix} w_i \\ \theta_{xz_i} \\ \theta_{yz_i} \end{bmatrix} \tag{IV.18}$$

$$\mathbf{Q} = \begin{bmatrix} Q_x \\ Q_y \end{bmatrix} = \mathbf{D_s} \cdot \boldsymbol{\varepsilon_s} = \mathbf{D_s} \cdot \mathbf{B_s} \cdot \begin{bmatrix} w_i \\ \theta_{xz_i} \\ \theta_{yz_i} \end{bmatrix} \tag{IV.19}$$

IV.3.4.Integração Parametrizada

Substituindo as expressões de $\boldsymbol{\kappa_b}$, e $\boldsymbol{\varepsilon_s}$ em (IV.12), podemos re-expressar a energia de deformação da seguinte forma:

$$\mathrm{U} = \frac{1}{2}\int_A \tilde{\mathbf{u}}^{\mathrm{T}} \cdot \mathbf{B_b}^{\mathrm{T}} \cdot \mathbf{D_b} \cdot \mathbf{B_b} \cdot \tilde{\mathbf{u}} + \tilde{\mathbf{u}}^{\mathrm{T}} \cdot \mathbf{B_s}^{\mathrm{T}} \cdot \mathbf{D_s} \cdot \mathbf{B_s} \cdot \tilde{\mathbf{u}}\ dA \tag{IV.20}$$

A minimização da parcela do funcional relativa à energia de deformação com relação aos deslocamentos nodais do elemento fornecerá as matrizes de rigidez de flexão e de cisalhamento, conforme mostrado a seguir:

$$\int_A \mathbf{B_b}^{\mathrm{T}} \cdot \mathbf{D_b} \cdot \mathbf{B_b} \cdot \tilde{\mathbf{u}} + \mathbf{B_s}^{\mathrm{T}} \cdot \mathbf{D_s} \cdot \mathbf{B_s} \cdot \tilde{\mathbf{u}}\ dA = (\mathbf{K_b} + \mathbf{K_s}) \cdot \tilde{\mathbf{u}} \tag{IV.21}$$

Já o vetor de forças nodais equivalentes, para carregamento vertical distribuído, pode ser obtido através seguinte equação:

$$\int_A q \cdot w(x,y) dA = \int_A N_i \cdot q\ dA \tag{IV.22}$$

A formulação envolve o tratamento de integrais no domínio físico do problema. Trabalhar com essas integrais no seu domínio original gera uma complexidade muito grande na avaliação das mesmas, portanto, são realizadas transformações de coordenadas para um domínio de coordenadas naturais, onde a avaliação das mesmas nesse domínio pode ser feita de forma genérica com auxílio da quadratura de *Gauss*.

Desta forma, a avaliação das matrizes de rigidez e de cisalhamento pode ser feita em domínio parametrizado da seguinte forma:

$$\mathbf{K_b} = \int_A \mathbf{B_b}^{\mathrm{T}} \cdot \mathbf{D_b} \cdot \mathbf{B_b}\ dA = \int_{-1}^{1}\int_{-1}^{1} \mathbf{B_b}^{\mathrm{T}} \cdot \mathbf{D_b} \cdot \mathbf{B_b} \cdot |\boldsymbol{J}|\ d\xi\ d\eta \quad \text{(IV.23)}$$

$$\mathbf{K_s} = \int_A \mathbf{B_s}^{\mathrm{T}} \cdot \mathbf{D_s} \cdot \mathbf{B_s}\ dA = \int_{-1}^{1}\int_{-1}^{1} \mathbf{B_s}^{\mathrm{T}} \cdot \mathbf{D_s} \cdot \mathbf{B_s} \cdot |\boldsymbol{J}|\ d\xi\ d\eta \quad \text{(IV.24)}$$

$$\mathbf{K_T} = \mathbf{K_b} + \mathbf{K_s} \quad \text{(IV.25)}$$

Com esta filosofia, podem ser gerados diversos elementos isoparamétricos retangulares, que serão tratados no capítulo com os exemplos de aplicação.

CAPÍTULO V

ASPECTOS COMPUTACIONAIS E PARALELIZAÇÃO

V.1.Introdução

Neste capítulo serão comentados alguns aspectos com relação à implementação computacional do programa construído para a realização deste trabalho. Serão discutidas também algumas técnicas de paralelização do programa, utilizadas para obtenção de um melhor desempenho, em termos de tempo de processamento.

De uma forma geral, o programa desenvolvido para este trabalho funciona exatamente com a mesma estrutura que qualquer programa de elementos finitos. Desta forma, as etapas podem ser resumidas conforme o fluxograma a seguir.

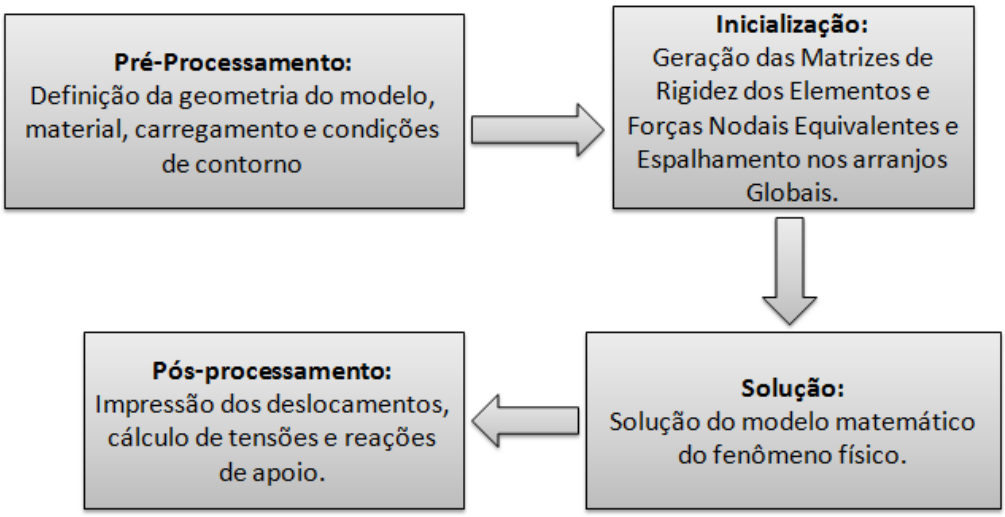


Figura V.1 – Fluxograma com atividades típicas de um programa com o MEF.

Portanto, serão brevemente descritas a seguir cada uma das sub-rotinas presentes no programa, e depois serão feitas as considerações com relação à paralelização do código.

V.2.Descrição das sub-rotinas

Nesta seção, será brevemente discutida cada sub-rotina do programa desenvolvido na realização deste trabalho.

V.2.1.Sub-rotina Multiplica

Esta sub-rotina realiza apenas a multiplicação de matrizes da seguinte forma.

$$\mathrm{C}(m,q) = \mathrm{A}(m,n) \cdot \mathrm{B}(n,q) \tag{V.1}$$

V.2.2.Sub-rotina Transposta

Esta subrotina calcula a transposta de uma matriz. No entanto, a matriz não precisa ser necessariamente quadrada. O procedimento básico de transposição se baseia em realizar a operação a seguir para todos os elementos da matriz.

$$\mathrm{A}^T(j,i) = \mathrm{A}(i,j) \tag{V.2}$$

V.2.3.Sub-rotina Título

Esta sub-rotina realiza apenas a impressão de um cabeçalho na tela, no instante em que o programa é chamado.

V.2.4.Sub-rotina Dados

A sub-rotina dados realiza a impressão dos dados lidos no arquivo de saída de forma formatada. Este fato é uma boa prática de programação, pois se torna muito mais fácil checar se os parâmetros de entrada estão corretos no caso de um erro na execução do programa.

V.2.5.Sub-rotina Apontador

Esta sub-rotina tem a função de avaliar as posições dos elementos da diagonal principal no armazenamento em perfil da matriz de rigidez. Este esquema de apontamento pode ser encontrado de outras formas na literatura, mas a filosofia do *Skyline* é sempre a mesma.

Para isto, obviamente a sub-rotina leva em consideração as posições que serão ocupadas de acordo com a matriz de incidência nodal fornecida no arquivo de entrada.

V.2.6.Sub-rotina Monta-Matriz

Esta sub-rotina é responsável pela chamada de avaliação e espalhamento das matrizes de rigidez dos elementos na matriz de rigidez global. Convém lembrar mais uma vez que o armazenamento da matriz de rigidez global é em perfil (*Skyline*).

V.2.7.Sub-rotina Mat_elm

Esta é a sub-rotina mais importante do programa. Ela é responsável pela geração das matrizes de rigidez dos elementos, que é uma das tarefas que envolvem o maior nível de complexidade dentro de um programa de elementos finitos.

Esta sub-rotina é ainda a parte mais particular do programa, quando comparado a outro programa de elementos finitos padrão, pois todas as outras etapas são muito parecidas em qualquer análise de elementos finitos.

V.2.8.Sub-rotina Cholesky

Esta sub-rotina é responsável pela resolução do sistema linear. A resolução é feita de forma direta, realizando a fatoração de *Cholesky* da matriz A, e depois realizando uma etapa de dupla retro-substituição.

V.2.9.Sub-rotina Contorno

Esta sub-rotina é responsável pela imposição das condições de contorno de acordo com o método do número grande. Vale ressaltar que o número grande utilizado no programa é 10e+20 e, como o método do número grande fornece soluções aproximadas, problemas sérios podem ser encontrados se a ordem da rigidez dos termos da matriz de rigidez global estiver próxima deste número. Este número também não deve ser exageradamente grande para não ocasionar problemas de *over-flow*.

V.2.10.Sub-rotina Saídamatriz

Esta sub-rotina é apenas para o controle do programa (se for solicitada) e realiza a impressão do vetor apontador e vetor de trabalho A para checagem de geração.

V.2.11.Sub-rotina Reações

Esta sub-rotina é responsável pelo cálculo das reações de apoio, e leva em consideração que o esquema de imposição das condições de contorno é o método do número grande.

V.2.12.Sub-rotina Deslocamentos

Esta sub-rotina é responsável apenas pela impressão dos deslocamentos no arquivo de saída, de forma formatada.

V.2.13.Sub-rotina Gera_temp

Esta sub-rotina é responsável por gerar um arquivo temporário para a execução do programa. Este arquivo temporário nada mais é do que o arquivo de entrada sem a inclusão dos comentários. Os comentários ajudam o usuário a entrar com os dados corretamente. No final do programa o arquivo temporário é fechado e excluído.

V.3.Como montar um arquivo de entrada?

Foi elaborado um esquema para o arquivo de entrada de forma que este possuísse linhas de comentários. Desta forma, toda a linha que possui o caractere "#", constitui uma linha comentada. Todas as linhas comentadas podem ser alteradas de forma livre e sem cautela.

A criação de linhas comentadas fornece uma liberdade muito grande para inserir indicadores com a seqüência do que deve ser fornecido ao programa em cada linha. Esta prática torna a tarefa do usuário do programa muito ágil na modificação de algum parâmetro, ou até na geração de um novo arquivo de entrada.

O arquivo de entrada deve possuir extensão ".INP" (que significa, input).

Os parâmetros fornecidos no programa por linha (ou bloco) são os seguintes:

- Número de nós;

- Coordenadas nodais;
- Número de propriedades do material;
- Propriedades do material;
- Número de elementos;
- Conectividades dos elementos (incidências);
- Número de condições de contorno;
- Condições de contorno;
- Número de nós carregados;
- Cargas nodais;
- Número de elementos carregados;
- Carregamento nos elementos.

A geração de coordenadas nodais pode ser feita de uma forma simples com o auxílio do *Excel* para malhas regulares e elementos com geometria em faixas de retângulos. Para a geração de malhas mais complexas, deve ser utilizado um software com gerador de malhas, como o *GID*. Muitos destes *softwares* são gratuitos e possuem versões acadêmicas. Pacotes de elementos finitos como o *ANSYS* também possuem geradores de malhas que podem exportar dados de coordenadas nodais e incidência de elementos.

As condições de contorno essenciais são impostas apontando o nó que deve ser restringido e indicando os graus de liberdade através do índice 1, no caso de grau de liberdade restringido.

O carregamento nodal (ou condição de contorno natural) é especificado de forma semelhante à condição de contorno essencial, porém, agora deve ser especificada a carga nodal em cada grau de liberdade do nó.

O carregamento no elemento é fornecido indicando-se o índice do elemento e a carga distribuída vertical por unidade de superfície.

V.4.Paralelização do código

A paralelização de um programa envolve atividades um pouco complexas, pois é necessário distribuir a análise entre vários processadores e distribuir ainda o armazenamento das variáveis em memória. Portanto, ao invés de ser investido na paralelização de todo o programa, é mais interessante investir naquelas regiões que realmente demandam muito processamento. Estas regiões são conhecidas como "*hot spots*".

Logo, antes de implementar paralelismo desgovernadamente no problema, deve ser realizada uma prévia análise de quais regiões irão acarretar em um ganho de desempenho real na execução do programa. Na maioria dos programas de elementos finitos existem duas etapas mais críticas, são elas: Geração das matrizes de rigidez dos elementos com espalhamento na matriz de rigidez global e solução do sistema linear. Ao longo desta seção será discutido como foi feita a otimização de cada uma dessas etapas.

A paralelização utilizada na implementação computacional gerada é um tipo de paralelização baseada em memória compartilhada através do uso de diretivas e bibliotecas baseadas em *OpenMP*. Outro tipo de paralelismo muito utilizado é baseado em programação com utilização de memória distribuída. Existem muitas aplicações deste tipo de paralelização, muitas delas fazendo o uso de bibliotecas de *MPI* (*Message Passing Interface*).

V.4.1.Paralelismo com uso de *OpenMP*

O *OpenMP* é uma interface para programação em paralelo cujas características são baseadas em facilitar aplicações de programação em paralelo com compartilhamento de memória. Se pretende com o *OpenMP* ser

conveniente para implementação num alcance amplo de arquiteturas como as de máquinas *multi-core* e processadores de *multi-threading* que têm se espalhado no mercado crescentemente.

O sucesso do *OpenMP* pode ser atribuído a vários fatores. Um fator forte é sua ênfase em programação paralela estruturada. Outra é que o *OpenMP* é comparativamente simples de usar, visto que o trabalho pesado dos detalhes da programação em paralelo são feitas pelo compilador. Existe ainda a vantagem de ser amplamente utilizado, de modo que uma aplicação de *OpenMP* é capaz de ser executada em várias plataformas.

De uma forma esquemática, as diretivas *OpenMP* são capazes de gerar regiões a serem executadas de forma paralela através de um esquema do tipo *FORK-JOIN*, conforme exemplifica a *Figura V.2*, onde é exemplificada a execução do programa com trechos de compartilhamento de trabalho pelas *threads* e trechos onde a execução é seqüencial (somente a *thread master* em trabalhando).

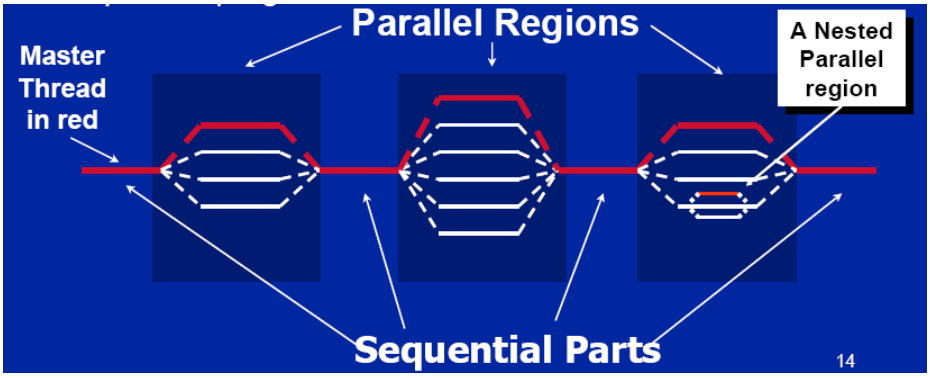


Figura V.2 – Representação de modelo Fork-Join para paralelismo.

Inicialmente, implementar alguns testes de execução de código em paralelo é muito simples e imediato através das diretivas *OpenMP*. No entanto, em aplicações mais complexas a atividade de paralelizar e exercer o controle das variáveis se torna muito complexo.

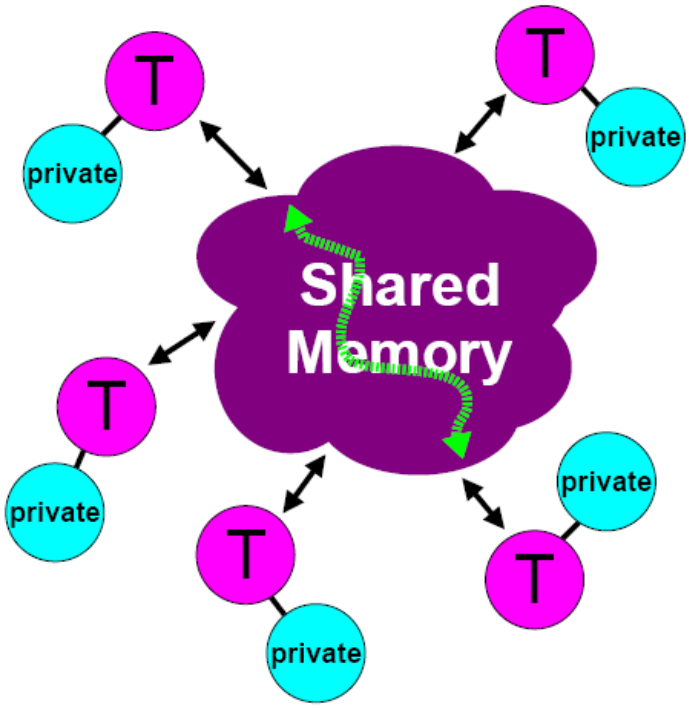


Figura V.3 – Representação das regiões privadas e compartilhadas de memória para cada Thread.

Conforme indicado na *Figura V.3*, devem ser especificadas as regiões de memória que serão compartilhadas e as regiões que serão privadas de cada *thread*. Regiões de memória compartilhada podem ser acessadas por todas as *threads*, enquanto que a região privada só pode ser executada pela *thread* que comanda aquela cópia.

Em aplicações simples o controle dessas variáveis é muito simples de ser feito, no entanto, para rotinas com muitas variáveis envolvidas o problema de especificar as variáveis se torna muito confuso.

V.4.2.Paralelização da Geração de matrizes dos elementos

A sub-rotina responsável pela chamada da geração das matrizes de rigidez dos elementos e espalhamento destas na matriz de rigidez global é a sub-rotina Montamatriz.

A sub-rotina é baseada em um laço com duas etapas principais. Este laço itera com todos os elementos e é responsável pela chamada da rotina de geração de matriz de rigidez do elemento (Matelm) e espalhamento na matriz de rigidez global em cada iteração. As matrizes de rigidez dos elementos são armazenadas em memória somente até o espalhamento na matriz global, posteriormente não é possível acessar os valores das matrizes do elemento sem chamar a rotina de geração das mesmas. Isso faz com que haja um congestionamento muito menor de memória durante a execução do programa.

O trabalho de gerar a matriz de rigidez de um elemento genérico i, e espalhamento na matriz de rigidez global é uma tarefa que independente em cada iteração do processo. Tal característica sugere uma idéia de que o trabalho pode ser dividido entre vários processos, criando um paralelismo nas atividades.

Portanto, através da utilização de diretivas *OpenMP* foi criado o paralelismo na geração das matrizes de rigidez dos elementos. O loop que itera nos elementos é "dividido" pelo número de processadores do computador (de arquitetura *multi-core*) e é possível ganhar agilidade na geração da matriz de rigidez global.

Uma particularidade importante dentre os comandos gerados na paralelização do código é no momento de se espalhar as matrizes de rigidez dos elementos na matriz de rigidez global. Como existem posições da matriz de rigidez global que compartilham rigidez advindas de mais de um elemento, existe a possibilidade de haver um conflito entre os processadores no momento em que esta posição é atualizada. Se isto ocorrer, certamente a atividade não será realizada de forma adequada e acarretará em erros na execução do programa.

Contudo, é importante ressaltar que tal erro (muito comum em computação paralela) tem caráter aleatório, pois depende do tempo de execução (*runtime*). Ou seja, diante de um problema desse tipo o programa pode ser executado perfeitamente em determinado instante ou cometer erros catastróficos em outros.

A forma adotada para prevenir este tipo de concorrência ao acesso de determinada área de memória foi a utilização da diretiva *ATOMIC*. Esta diretiva estabelece que o comando seguinte à esta especificação deva ser executado por uma *thread* de cada vez. Ou seja, se outra *thread* já estiver executando tal comando, a *thread* que chega deve esperar até que possa ela fazer o comando sem concorrência.

Esta tarefa impõe que determinada tarefa seja executada de forma seqüencial. A utilização destes comandos em todas as regiões paralelas iria impor uma condição de execução seqüencial em vários processadores. Portanto, deve ser tomado um cuidado muito grande na utilização desses comandos e estabelecimento de regiões críticas, pois estas medidas podem piorar muito o desempenho.

V.4.3.Paralelismo na solução do sistema de equações

O paralelismo da solução do sistema linear foi feito utilizando bibliotecas do pacote *Math Kernel Library* (*MKL*), presente no compilador *Intel Visual Studio 2008*. Esta biblioteca permite possibilita um ganho de desempenho significativo em aplicações científicas, de engenharia e de finanças resolvendo problemas computacionais muito robustos. Este pacote contém rotinas de álgebra linear, transformadas rápidas de *Fourier* (*FFT*), geração de números aleatórios, entre outras. Este pacote oferece ainda bom desempenho para processadores de outras marcas (não *Intel*) e é gerado através de programação paralela com uso de *OpenMP*, oferecendo compatibilidade com variáveis de ambiente desta origem.

Será feito um breve comentário sobre resolução de sistemas lineares e esquemas de armazenamento de matrizes esparsas.

Em geral, nos problemas de elasticidade a tarefa de um programa de elementos finitos é calcular os deslocamentos nodais de um determinado meio. Tais deslocamentos, em uma análise estática linear consistem na resolução de um sistema de equações lineares da seguinte forma:

$$\mathbf{K} \cdot \mathbf{d} = \mathbf{F} \tag{V.3}$$

A solução de sistemas de equações é um dos procedimentos matemáticos mais utilizados em aplicações práticas. Portanto, existem diversos procedimentos computacionais para resolver o sistema e isso é alvo de pesquisa em diversos institutos de matemática.

No problema em questão, o sistema possui uma matriz real simétrica positiva definida. A matriz de rigidez é real porque ela possui coeficientes reais. A mesma é positiva definida por possuir a característica do produto vetor transposto x matriz x vetor ser sempre positivo e não nulo. Por conseguinte é possível realizar a fatoração de *Cholesky*, encontrando uma matriz triangular inferior que multiplicada por sua transposta a matriz original é obtida da seguinte forma:

$$\mathbf{K} \cdot \mathbf{d} = \mathbf{F} \quad \rightarrow \quad \mathbf{L} \cdot \mathbf{L}^{\mathrm{T}} \cdot \mathbf{d} = \mathbf{F} \tag{V.4}$$

Após esta fatoração, o sistema pode ser resolvido através de duas retro-substituições sucessivas da seguinte forma:

$$\mathbf{L} \cdot \mathbf{b} = \mathbf{F} \quad \rightarrow \quad \mathbf{L}^{\mathrm{T}} \cdot \mathbf{d} = \mathbf{b} \tag{V.5}$$

V.4.3.1. Tipos de armazenamento de matrizes esparsas

Em muitas aplicações de resolução de sistemas de equações, o problema envolve uma matriz muito esparsa. A esparsidade é maior quanto maior for o número de termos nulos na matriz do sistema. Nesses casos, muitas operações com termos nulos são geradas de forma desnecessária, mas estão de acordo com o algoritmo alvo.

No entanto, existem técnicas na resolução de sistemas para tirar vantagem da característica esparsa da matriz do sistema. Estas técnicas tornam a solução do sistema muito mais ágil, por diminuir o número de operações de ponto flutuante.

Na aplicação destas técnicas, é necessária uma estratégia de armazenamento da matriz. Geralmente, o que se faz é armazenar os termos da matriz em um vetor e criar um esquema de apontamento dessas posições para auxiliar o acesso no espalhamento e solução do sistema.

V.4.3.1.1.Esquema de armazenamento *Skyline*

Este esquema tira vantagem da característica simétrica da matriz de rigidez e armazena somente os coeficientes da parte triangular superior da matriz de rigidez. Para exemplificar o armazenamento considere a matriz simétrica a seguir.

$$\mathbf{C} = \begin{bmatrix} 1 & -1 & -3 & 0 & 0 \\ -1 & 5 & 0 & 0 & 0 \\ -3 & 0 & 4 & 6 & 4 \\ 0 & 0 & 6 & 7 & 0 \\ 0 & 0 & 4 & 0 & -5 \end{bmatrix} \tag{V.6}$$

O esquema de armazenamento do tipo *Skyline*, armazena a matriz coluna por coluna, considerando o armazenamento do primeiro coeficiente não nulo da coluna até a diagonal principal da matriz. Desta forma, cada coluna possui uma altura de armazenamento e a matriz armazenada em um vetor ficaria da seguinte forma.

$$\mathbf{A_{Skyline}} = [1 \quad -1 \quad 5 \quad -3 \quad 0 \quad 4 \quad 6 \quad 7 \quad 4 \quad 0 \quad -5] \tag{V.7}$$

Para auxiliar o acesso aos termos é necessária a adoção de um esquema de apontamento dos elementos. Existem vários esquemas de apontamento para o elemento da diagonal principal da matriz ou apontar o primeiro termo não nulo da coluna. *BATHE*[3], sugere um esquema de armazenamento com esta filosofia um pouco diferente, onde o vetor **A** contém os termos de **C** armazenados de baixo para cima e apontando o elemento da diagonal principal. Já na biblioteca do *Math Kernel Library*, os termos de **A** são armazenados da mesma forma com o apontamento do primeiro termo não nulo da coluna. Neste trabalho, o esquema de armazenamento é o mesmo que o anterior com o apontamento do elemento da

diagonal principal da matriz. Desta forma, pode-se montar o seguinte vetor apontador:

$$\mathbf{p} = [1 \quad 3 \quad 6 \quad 8 \quad 11] \tag{V.8}$$

Porém, para utilizar as rotinas que usam o armazenamento *Skyline* do pacote *MKL*, este esquema de apontamento deve ser modificado, resultando então no vetor apontador mostrado a seguir.

$$\mathbf{p} = [1 \quad 2 \quad 4 \quad 7 \quad 9 \quad 12] \tag{V.9}$$

V.4.3.1.2.Esquema de armazenamento *CSR*

Outro tipo de armazenamento é o armazenamento do tipo *CSR (Compressed Sparsed Row)*. Este esquema é um dos mais utilizados e eficientes na solução de matrizes esparsas. São criados três vetores de trabalho, sendo que um vetor contém os termos da matriz (Vetor $\mathbf{A_{CSR}}$) e é responsável pelo armazenamento dos termos não nulos seguindo um critério de armazenamento da esquerda para a direita. Existe ainda um vetor que indica as colunas de cada termo da matriz (**JA**) e outro que indica onde são iniciadas as linhas da matriz (**IA**). Segundo esta filosofia e tomando vantagem da simetria da matriz **C**, podemos representá-la com da seguinte forma:

$$\mathbf{A_{CSR}} = [1 \quad -1 \quad -3 \quad 5 \quad 4 \quad 6 \quad 4 \quad 7 \quad -5] \tag{V.10}$$

$$\mathbf{JA} = [1 \quad 2 \quad 3 \quad 2 \quad 3 \quad 4 \quad 5 \quad 4 \quad 5] \tag{V.11}$$

$$\mathbf{IA} = [1 \quad 4 \quad 5 \quad 8 \quad 9 \quad 10] \tag{V.12}$$

V.4.3.2.Estratégia para Resolução do Sistema

A forma de armazenamento inicialmente utilizada no programa era do tipo *Skyline*. Utilizar uma estratégia de armazenamento do tipo *CSR* desde o início do programa iria acarretar mudanças grandes em várias rotinas. Logo, impor as alterações somente na parte de solução do sistema foi a forma mais

rápida e eficiente de buscar um bom ganho de desempenho sem impor alterações em toda a extensão do código.

Desta forma, toda a estrutura de geração da matriz de rigidez com armazenamento do tipo *Skyline* com apontamento dos elementos da diagonal principal foi mantida e foram impostas as seguintes etapas como forma de troca da rotina inicial de solução do sistema.

- Criação de novo vetor apontador (tipo *MKL*);
- Conversão de esquema de armazenamento *Skyline* para *CSR*;
- Resolução do sistema com armazenamento *CSR*.

A razão da conversão do esquema de armazenamento *Skyline* para o tipo *CSR* é a não existência de rotina para solução de sistema com matriz não triangularizada no pacote *MKL.*

Após a conversão para o esquema de armazenamento *CSR* foi utilizada a rotina para solução direta paralela de matrizes esparsas, a rotina *PARDISO®* (*Parallel Direct Solver*), presente no pacote *MKL* desenvolvido pela *Intel®* para *Fortran*.

Apesar da introdução de algumas atividades a mais antes de realizar a solução do sistema, o ganho final de desempenho foi muito interessante, conforme será apresentado no capítulo a seguir com exemplos de aplicação.

CAPÍTULO VI

EXEMPLOS DE APLICAÇÃO

VI.1.Introdução

Neste capítulo serão mostrados os resultados da aplicação do método dos elementos finitos à análise de placas espessas testando vários tipos de elementos, em programa gerado em *FORTRAN*, verificando o comportamento com relação aos tipos de integração utilizados.

Várias análises foram realizadas de forma a compor gráficos que se encontram muito presentes na literatura (*Cook*[5], *Zienkiewicz*[2], entre outros), que trata do comportamento de elementos isoparamétricos retangulares.

Com esses resultados serão feitas algumas avaliações do comportamento dos elementos com relação ao travamento da malha de elementos finitos.

Após essas considerações, será feita uma breve avaliação do desempenho computacional da análise para implementação com o elemento quadrático *Lagrangeano*. Serão feitos comentários sobre os ganhos de desempenho e da eficiência do paralelismo.

VI.2.Elemento Bi-linear

Nesta seção, será comentada aplicação do elemento finito isoparamétrico de placa espessa Bi-linear, mostrado na Figura VI.1.

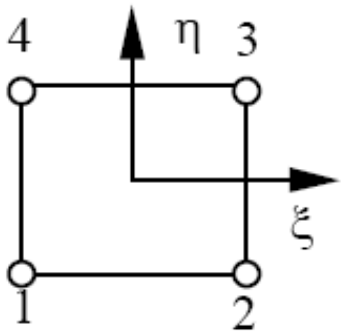


Figura VI.1 – Representação de elemento Bi-linear em domínio parametrizado.

Quanto ao processo de integração para obtenção das matrizes de rigidez e esforços nodais equivalentes, este será realizado em domínio parametrizado. No entanto, com relação ao número de pontos de Gauss serão testadas duas hipóteses, integração completa e integração seletiva, como pode ser visto na Tabela 5. A razão para as duas hipóteses é de verificar o comportamento do elemento com relação ao travamento que será discutido mais adiante.

Tabela 5 – Especificação do tipo de integração (Elemento Bi-linear).

ELEMENTO BILINEAR		Matriz de Rigidez de Flexão	Matriz de Rigidez de Cisalhamento
N° de pontos de Gauss	Integração Completa	2 x 2	2 x 2
	Integração Seletiva	2 x 2	1 x 1

As funções de interpolação deste elemento podem ser obtidas por interpolação *Lagrangeana*, que resultam as funções de forma a seguir:

$$N_1 = \frac{1}{4} \cdot (1 - \xi) \cdot (1 - \eta) \tag{VI.1}$$

$$N_2 = \frac{1}{4} \cdot (1 + \xi) \cdot (1 - \eta) \tag{VI.2}$$

$$N_3 = \frac{1}{4} \cdot (1 + \xi) \cdot (1 + \eta) \tag{VI.3}$$

$$N_4 = \frac{1}{4} \cdot (1 - \xi) \cdot (1 + \eta) \tag{VI.4}$$

Para testar este fenômeno com esses dois elementos (integração completa e seletiva), foi feito um teste e comparados os resultados com a teoria de placas finas (*Kirchhoff*). Para este teste, foi analisada uma placa quadrada engastada em todos os bordos utilizando uma malha regular com 64 elementos. Este teste é exatamente o mesmo empregado por *Cook*[5] para verificar o mesmo fenômeno. *Zienkiewicz*[2] também propôs um teste muito parecido. Desta forma, puderam-se comparar os resultados obtidos com os resultados apresentados na literatura, onde além de verificar o travamento era validado o programa desenvolvido neste trabalho.

As integrais necessárias no processo de geração das matrizes de rigidez dos elementos envolvem no máximo tais funções de interpolação ao quadrado, resultando em funções bidimensionais quadráticas. Portanto, de acordo com as regras de integração gaussiana de um polinômio são necessários no mínimo dois pontos de Gauss em cada dimensão para garantir integração exata da função.

Conforme a relação L/t (vão/espessura) aumenta, os resultados encontrados com as duas teorias devem coincidir, visto que as parcelas de deformações cisalhantes ficarão cada vez menores frente as deformações devido ao efeito de flexão. No entanto, se forem integradas ambas as matrizes de rigidez (flexão e cisalhamento) com dois pontos de integração os resultados divergem muito da solução de placas finas conforme se aumenta a relação L/t. Isto ocorre por conta do fenômeno do travamento.

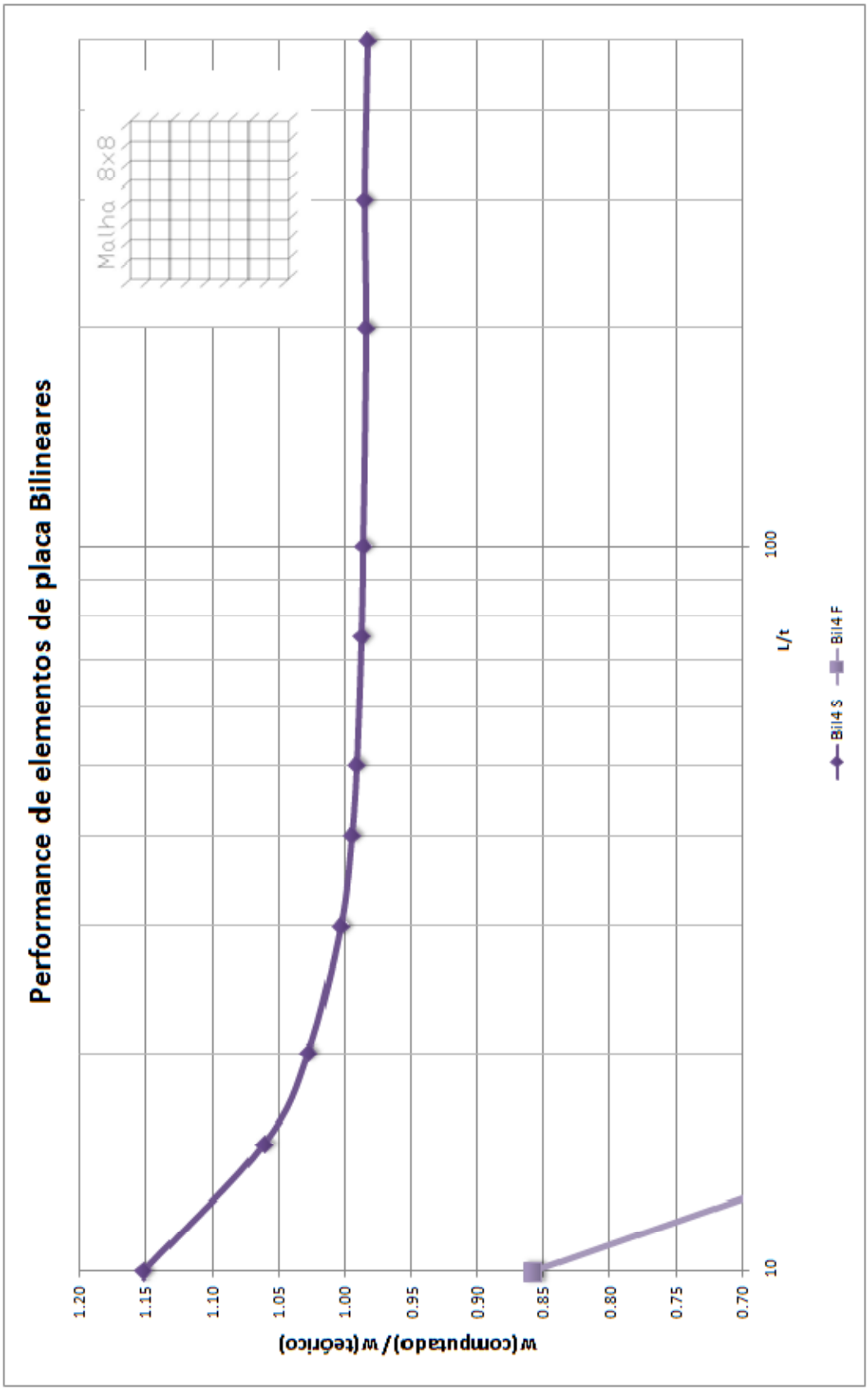


Figura VI.2 – Deflexão central de uma placa quadrada uniformemente carregada de razão lado/espessura L/t. Elementos Bi-lineares com integração seletiva e completa.

Conforme visto na Figura VI.2, os resultados encontrados para o elemento com integração completa foram muito ruins. Já com a integração seletiva os resultados se aproximam muito bem da teoria de placas finas conforme se aumenta a relação L/t. No entanto, é possível notar que mesmo com integração seletiva os resultados não coincidiram com a teoria de placas finas, apresentando então uma rigidez além de esperado no comportamento.

VI.3.Elemento Quadrático de *Serendipity* (8 nós)

A seguir, foi testado outro elemento isoparamétrico com interpolação quadrática, mostrado na Figura VI.3.

Nesta seção será testado o elemento com interpolação quadrática de *Serendipity*. Este elemento possui polinômios de interpolação quadrática e possui nós (graus de liberdade) apenas nos cantos e nas arestas (que podem ser curvas). O elemento quadrático com interpolação *Lagrangeana* (que possui nó no seu interior) também foi testado e será discutido na próxima seção.

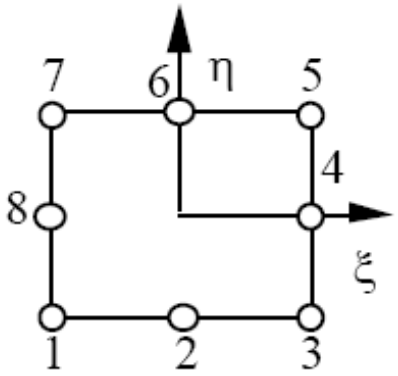


Figura VI.3 – Representação de elemento quadrático (Serendipity) em domínio parametrizado.

Quanto ao processo de integração serão testadas três hipóteses, integração completa, integração seletiva e integração reduzida, conforme Tabela 6.

Tabela 6 – Especificação do tipo de integração (Elemento Quadrático de Serendipity).

ELEMENTO SERENDIPITY (8 nodes)		Matriz de Rigidez de Flexão	Matriz de Rigidez de Cisalhamento
N° de pontos de Gauss	Integração Completa	3 x 3	3 x 3
	Integração Seletiva	3 x 3	2 x 2
	Integração Reduzida	2 x 2	2 x 2

As funções de interpolação deste elemento podem ser obtidas de acordo com a tabela a apresentada em capítulo anterior (Tabela 4).

As integrais necessárias no processo de geração das matrizes de rigidez dos elementos envolvem no máximo tais funções de interpolação ao quadrado, resultando em funções bidimensionais de quarto grau. Portanto, de acordo com as regras de integração gaussiana de um polinômio (vistas em capítulo anterior) são necessários no mínimo três pontos de *Gauss* em cada dimensão para garantir integração exata da função.

Para testar este fenômeno do travamento, foi realizado o mesmo teste do elemento Bi-linear empregando uma malha de 64 elementos para avaliar o comportamento de uma placa quadrada engastada em todos os bordos. Os resultados encontrados são mostrados no gráfico na Figura VI.4.

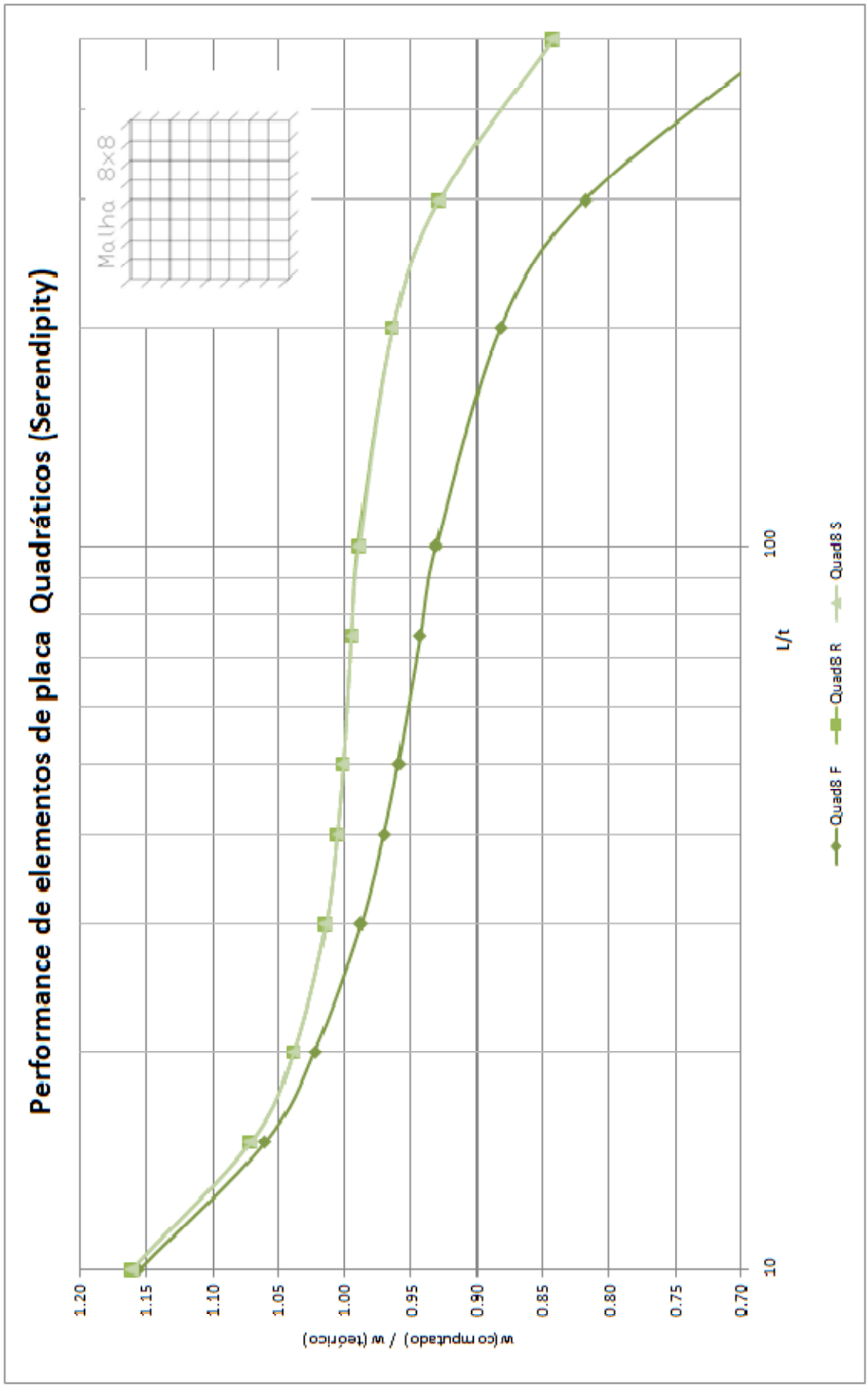


Figura VI.4 – Deflexão central de uma placa quadrada uniformemente carregada de razão lado/espessura L/t. Elementos Serendipity com integração red., sel. e completa.

Portanto, puderam ser verificados que para qualquer tipo de integração o elemento apresenta o travamento da malha de elementos finitos com o aumento da relação L/t. Outro ponto interessante é que mesmo com integração 2 x 2 para a flexão e para cisalhamento (integração reduzida) apresentou resultados muito próximos aos encontrados com integração seletiva . Isto ocorre porque os termos a serem integrados na geração da matriz de rigidez de flexão não demandam tantos pontos assim de integração por envolverem mais derivadas das funções de interpolação.

VI.4.Elemento Quadrático de *Lagrange* (9 nós)

Nesta seção será avaliado o elemento quadrático com interpolação *Lagrangeana*, mostrado na Figura VI.5.

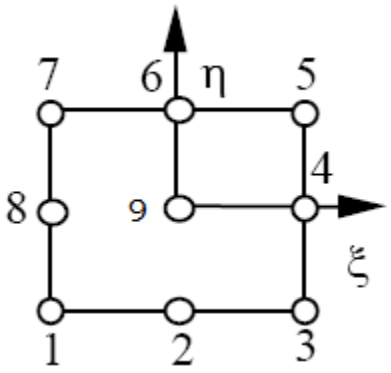


Figura VI.5 – Representação de elemento quadrático (Lagrange) em domínio parametrizado.

Quanto ao processo de integração serão testadas três hipóteses, integração completa, integração seletiva e integração reduzida, que podem ser vistas na Tabela 7.

Tabela 7 – Especificação do tipo de integração (Elemento Quadrático de Lagrange).

ELEMENTO LAGRANGEANO (9 nodes)		Matriz de Rigidez de Flexão	Matriz de Rigidez de Cisalhamento
N° de pontos de Gauss	Integração Completa	3 x 3	3 x 3
	Integração Seletiva	3 x 3	2 x 2
	Integração Reduzida	2 x 2	2 x 2

As funções de interpolação deste elemento podem ser obtidas por interpolação *Lagrangeana*, ou de acordo com a mesma tabela apresentada na seção anterior.

Realizando os mesmos testes realizados para avaliar os elementos Bi-linear e *Serendipity*, obtiveram-se os resultados mostrados na Figura VI.6.

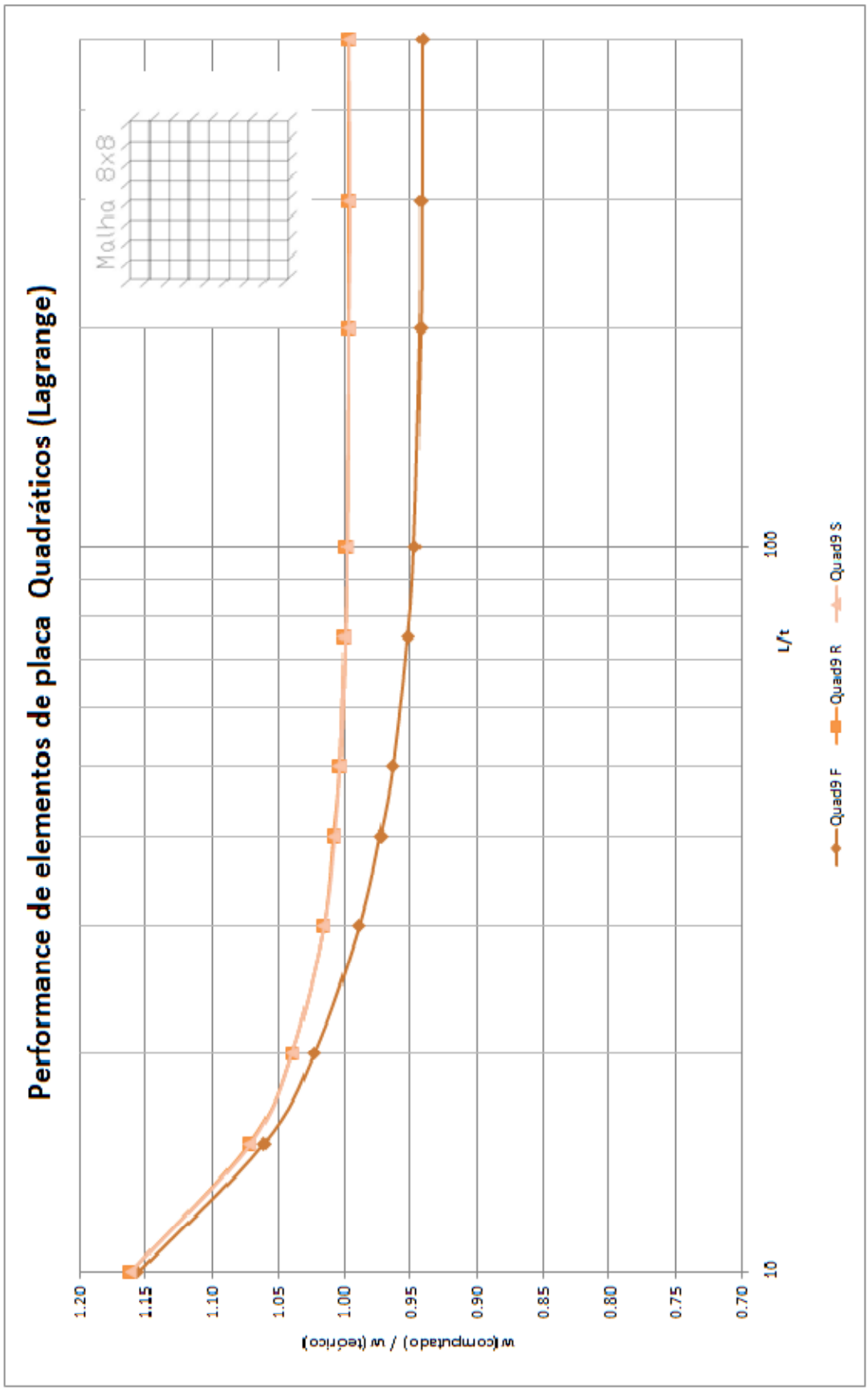


Figura VI.6 – Performance de elementos Lagrange com integração reduzida, seletiva e completa.

Verificou-se que usando integração completa o elemento *Lagrangeano* ainda apresentou um comportamento fora do esperado. No entanto, esta divergência não foi tão acentuada quanto àquela encontrada para o elemento quadrático de oito nós. Com a integração reduzida e seletiva os resultados encontrados foram excelentes para este elemento.

O elemento quadrático *Lagrangeano* apresentou as melhores respostas. Para explicitar ainda mais esse bom resultado é mostrado na Figura VI.7 um gráfico, onde pode ser comparada a performance de cada um dos elementos empregados no trabalho.

Na literatura (*Cook*[5]), ainda é discutido outro tipo de elemento que se comporta muito bem com relação ao travamento. Este elemento é chamado nessas bibliografias de elemento *Heterosis* e possui 8 nós, como o elemento de *Serendipity* apenas para os graus de liberdade referentes ao deslocamento transversal, para as rotações ele possui 9 nós resultando em um elemento de 26 graus de liberdade. Segundo *Cook*[5], é possível ainda formular outros tipos de elementos híbridos com bom comportamento com relação ao travamento.

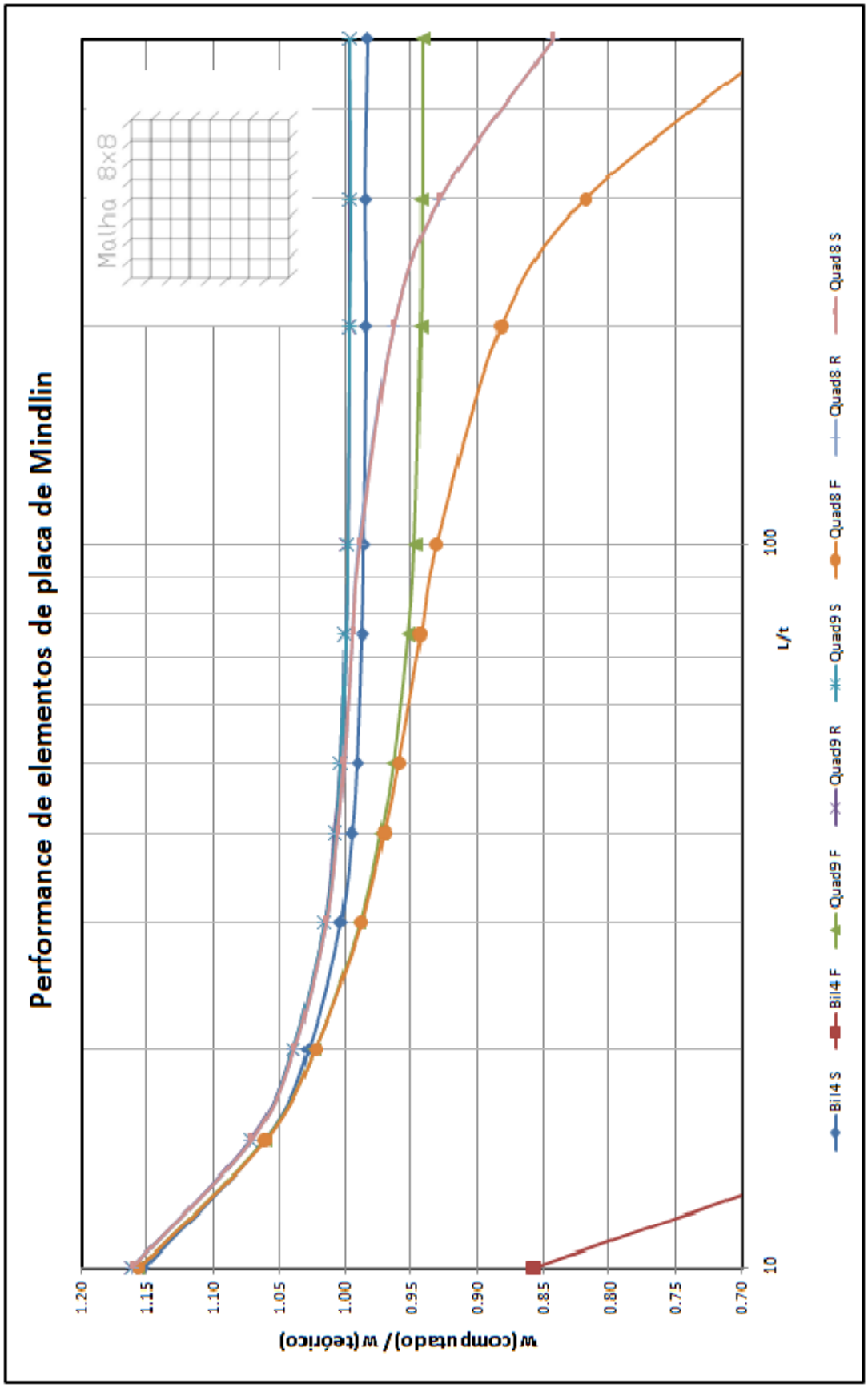


Figura VI.7 – Performance de todos os elementos analisados com relação ao travamento.

VI.5.Travamento da Malha

O travamento do esforço cortante ocorre pelo aparecimento de termos muito altos dentro da matriz de rigidez de cisalhamento. Segundo *Cook*[5], cada ponto de integração impõe duas restrições ao elemento, uma associada às deformações γ_{xz} e outra associada às deformações γ_{yz}. Desta forma, a redução do número de pontos de integração faz com que seja reduzido o número de restrições adicionais ao sistema. Este mesmo fenômeno ocorre nos elementos de viga formulados pela teoria de *Timoshenko*.

É importante ainda mencionar que a realização de uma análise modal do problema em questão pode nos levar a conclusões interessantes. A análise modal do sistema irá nos fornecer os autovalores e autovetores associados à matriz de rigidez **K**. Tais autovetores têm significado físico e representam os vários possíveis modos de deformação que um elemento pode ter. Associados a cada um destes autovetores existem autovalores que representam uma espécie de rigidez do elemento contra àquele modo de deformação. Como a matriz de rigidez do elemento sem restrições é singular, sempre existirão autovalores nulos que representarão deslocamentos de corpo rígido ao elemento com energia de deformação nula. Ao usarem-se integrações com números reduzidos de pontos, surgem outros autovalores nulos que representam modos espúrios sem energia de deformação e sem significado físico algum. Por estas razões pode-se concluir que o processo de integração reduzida realmente impõe menos restrições ao elemento. *Lima*[42] realizou análises modais do elemento quadrático *Serendipity* e verificou o aparecimento de apenas mais um modo espúrio de energia nula com uso de integração reduzida. Logo, pode-se verificar que para este elemento, mesmo que seja utilizada a integração reduzida, não são introduzidas liberações suficientes para o não travamento da malha, justificando os maus resultados encontrados para este tipo de elemento.

VI.6.Desempenho computacional usando paralelização

Conforme o comentado no capítulo anterior, estudou-se o ganho de desempenho computacional nas duas etapas mais dispendiosas do processo de análise do problema. Estas etapas correspondem às etapas de geração de matrizes de rigidez dos elementos com espalhamento na matriz de rigidez global e solução do sistema de equações.

Executando a análise usada para a geração dos resultados mostrados para o elemento quadrático de *Lagrange* foram feitos os seguintes testes:

- Avaliação do desempenho sem nenhum tipo de paralelismo e utilizando a rotina de solução do sistema de equações original(que faz uso do método de *Cholesky*). Este exemplo será chamado de SEQ para simplificação e apresentação dos resultados a seguir;

Para as demais avaliações foi utilizado n processadores para geração das matrizes dos elementos e solução do sistema de equações (fazendo uso do pacote. *MKL*, *Intel*®). Estes exemplo serão chamados de:

- PARA1, n=1 (um processador);
- PARA2, n=2 (dois processadores);
- PARA4, n=4 (quatro processadores);
- PARA8, n=8 (oito processador).

Com a geração destes resultados foi montada a Tabela 8, de onde podem ser tiradas algumas conclusões.

Tabela 8 –Tempo de processamento com múltiplos processadores.

TEMPO (seg)	Geração de Matriz de Rigidez	Solução do Sistema de Equações	TOTAL
SEQ	0.063313	4.985188	5.141812
PARA1	0.046375	0.171875	0.343125
PARA2	0.032656	0.173781	0.313781
PARA4	0.015625	0.125015	0.28125
PARA8	0.012178	0.126469	0.345594

Na tabela anterior pode ser notado que simplesmente a mudança de rotina responsável pela resolução do sistema de equações lineares utilizada, acarretou em um ganho significativo na etapa de solução do sistema de equações. Isto foi devido à resolução do sistema com o armazenamento *CSR* e utilização da rotina *Pardiso*® (Pacote *MKL*®). Vale lembrar ainda que o armazenamento do tipo *Skyline* guarda muitos coeficientes nulos da matriz esparsa, o que torna o armazenamento *CSR* ainda mais vantajoso. Com o método *Skyline*, os elementos, que devem ser conectados em nove nós, podem encontrar-se muito distantes em termos matriciais, acarretando uma altura de perfil muito grande e aumentando muito o número de operações na fatoração.

A resolução do sistema de equações foi um pouco mais eficiente com o aumento do número de processadores, mas não foi possível notar um grande acréscimo de desempenho e isso se deu por dois motivos. O primeiro é a magnitude do problema estudado que consistiu em uma análise relativamente pequena e portanto a demanda de processamento não era grande o suficiente para tornar a divisão em processos eficientes. Outro aspecto é que o solver do *Pardiso*® consiste em uma resolução direta paralela do sistema de equações, portanto, existem as etapas de fatoração e retro-substituição, sendo que a segunda por razões óbvias não pode ser executada em paralelo acarretando em um trecho seqüencial na rotina.

Com relação à etapa geração das matrizes de rigidez dos elementos e montagem da matriz de rigidez global, esta foi mais rápida conforme o número de processadores foi aumentando. No entanto, conforme mostra o gráfico da Figura VI.8, o *Speedup* não foi igual ao número e processadores.

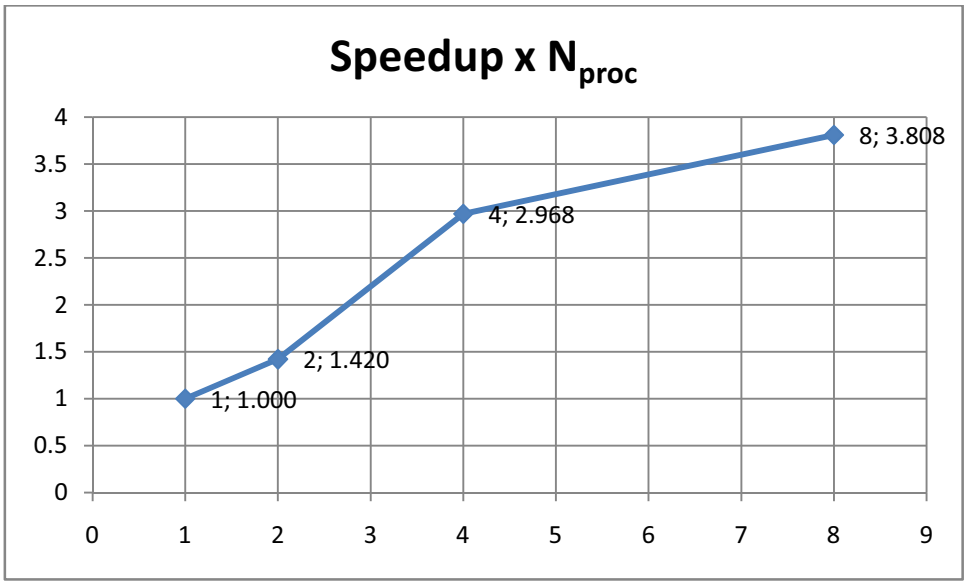


Figura VI.8 – Gráfico com avaliação da eficiência computacional com o aumento do número de processadores (Speedup).

O *Speedup* é uma forma de medir a eficiência de uma aplicação em paralelo que corresponde ao tempo de execução serial dividido pelo tempo de execução paralelo. Um *Speedup* igual ao número de processadores seria o paralelismo com eficiência de 100%, mas sempre há uma perda de eficiência com o aumento da divisão do processamento.

Outra conclusão que pode ser tirada do gráfico da Figura VI.8 é que a eficiência com quatro processadores foi maior do que a encontrada com dois processadores, já que houve uma inclinação mais acentuada no segundo trecho. Normalmente essa eficiência tenderia a diminuir, mas em alguns casos, quando são feitas as análises em computadores com arquitetura de muitos processadores (quatro ou oito), e são solicitados apenas dois para processamento paralelo, os resultados de desempenho podem não corresponder às expectativas. Para estes casos, já foram encontrados em outras oportunidades melhores desempenho em computadores *Dual Core* somente.

CAPÍTULO VII

CONCLUSÕES

Neste capítulo serão feitas as conclusões finais com relação ao trabalho desenvolvido e por fim serão feitas sugestões para trabalhos futuros.

VII.1.Conclusões

Durante a execução deste trabalho foi realizada de forma muito bem sucedida um estudo de revisão geral de método dos elementos finitos (MEF) com um pouco mais de detalhes e com um pouco menos de pressão existente normalmente em um curso de graduação ou mestrado. Este estudo preliminar foi muito importante na geração dos alicerces para a aplicação ao problema de flexão de placas.

Com relação à teoria de placas espessas, foi realizada uma extensa pesquisa na literatura, já que não é muito comum encontrar um bom desenvolvimento de todos os passos da teoria. Portanto, um dos objetivos do segundo capítulo foi cumprido, que era o de gerar um bom texto no tratamento do problema de flexão de placas espessas.

A implementação computacional do método dos elementos finitos se tratou de um trabalho muito valioso. O programa foi feito tomando como base um programa que fazia a análise via MEF de estruturas com elementos de grelha. Portanto, foi aproveitada a estrutura do programa e modificações foram necessárias em quase todas as rotinas presentes no programa, excluindo a resolução do sistema de equações lineares que em um primeiro instante foi

aproveitada integralmente. Isto fez com que fosse criada uma intimidade valiosa com um código padrão de elementos finitos.

Após a geração de códigos para análise dos três tipos de elementos isoparamétricos presentes neste trabalho (Bi-linear e quadráticos de *Serendipity* e *Lagrange*), foi realizada a implementação do paralelismo nas rotinas de geração de matrizes de rigidez dos elementos e de resolução do sistema de equações lineares, obtendo bons resultados em termos de desempenho para os exemplos rodados. A tarefa de geração de matrizes de rigidez dos elementos já havia sido implementada em outras aplicações com sucesso e os resultados obtidos tiveram o comportamento esperado. Com relação à resolução do sistema linear, a implementação do paralelismo com as rotinas do *MKL* superou as expectativas e a estratégica desenvolvida neste trabalho já está sendo usada inclusive em outras aplicações práticas de elementos finitos.

VII.2.Sugestões para trabalhos futuros

O programa desenvolvido neste trabalho abre caminhos de aplicação de análise de placas espessas para diversos tipos de análise com elementos finitos. Serão citadas a seguir algumas sugestões de implementações para o desenvolvimento de trabalhos futuros.

Uma aplicação futura interessante é a implementação de uma análise dinâmica no domínio do tempo. Para criar este tipo de análise, seria necessária a introdução de rotinas para a geração de matrizes de massa (massa efetiva ou consistente) e de amortecimento. Seria necessário ainda adotar um método de integração no tempo como o método da diferença finita central (explícito) ou método de *Newmark* (implícito). Se esta análise consistir em uma análise dinânica linear, só haveria a necessidade de avaliar uma vez a etapa de geração das matrizes de rigidez, massa e amortecimento, deixando o sistema efetivo com as mesmas características em todas as iterações. Logo, será necessária apenas a solução do sistema efetivo em cada iteração da integração no tempo, solução

esta que através da utilização do pacote *MKL* utilizada neste trabalho melhoraria muito o desempenho das análises. Sendo a análise uma análise não linear, seria necessário mudar a matriz efetiva a cada iteração, porém, fazendo o uso de diretivas *OpenMP* é possível obter um bom ganho computacional.

Ainda pensando em uma análise dinâmica, poderia ser implementada uma análise modal do sistema, obtendo os modos e as freqüências naturais de vibração da placa. Com a análise destes modos e freqüências pode ser utilizado ainda um método de redução de base para a solução de problemas inerciais (somente com componentes de vibração de baixa freqüência). Ou ainda, pode ser analisado um problema de propagação de ondas usando um método explícito de integração.

Outra avaliação futura que poderia ser feita é a consideração hipótese de uma não-linearidade física no material (desobedecendo a Lei de *Hooke* clássica). Esta não-linearidade pode ser associada a algum efeito de plasticidade ou um efeito reológico. Sendo uma não-linearidade dependente das deformações, deve ser implementada uma solução iterativa até convergir para uma configuração de equilíbrio. Outro tipo de consideração é a de um material visco-elástico, onde a relação constitutiva é montada com componentes de tensão, deformação e tempo, incluindo portanto efeitos reológicos do material.

Em termos de implementação de computação de alto desempenho, poderia ser elaborado um novo código utilizado para paralelismo em clusters baseado em memória distribuída (com *MPI – Message Passing Interface*). Com a utilização deste tipo de processamento seria possível resolver problemas realmente grandes, distanciando muito da análise estática linear que não iria demandar tanto processamento assim, a não ser que fosse feita uma análise com geometria extremamente exagerada de tamanho e discretização.

Outra sugestão é a modificação do código para ele ser iniciado e conter o endereçamento do espalhamento global já com a filosofia de armazenamento de matriz esparsa *CSR*. Isso pouparia o trabalho de ter de converter o formato de armazenamento antes da resolução do sistema linear com o solver *Pardiso*®.

Com relação ao paralelismo, existe ainda oportunidade de estender o aprendizado de paralelismo com memória compartilhada (*OpenMP*) para outras aplicações. Uma aplicação que poderia apresentar bons resultados seria a integração do Método de Monte Carlo para análise de Confiabilidade, uma vez que essa análise demanda um grande esforço computacional (para casos onde a probabilidade de falha é muito pequena principalmente), com geração de números aleatórios em determinados domínios que poderiam estar sendo gerados de forma paralela e ganhando muito em tempo de processamento da integração numérica.

REFERÊNCIAS BIBLIOGRÁFICAS

1. **Finite Element Method - The Basis,** O. C. Zienkiewicz and R.L Taylor, 5th Edition, Butterworth Heinemann.

2. **Finite Element Method - Solid mechanics,** O. C. Zienkiewicz and R.L Taylor, 5th Edition, Butterworth Heinemann.

3. **Finite Element Procedures,** K.J. Bathe, Prentice Hall, Inc.

4. **Finite Element Modeling for Stress Analysis,** R. D. Cook John Wiley and Sons, Inc.

5. **Concepts and Applications of Finite Element Analysis,** R. D. Cook, D. S. Malkus, M. E. Plesha, 3rd Edition, John Wiley and Sons, Inc.

6. **Finite Element Analysis - Fundamentals,** R. H.Gallagher, Prentice Hall, Inc.

7. **Finite Element Programming,** E. Hinton, D. R. Owen.

8. **Método dos Elementos Finitos,** Primeira Edição, Álvaro F. M. Azeredo, 2003.

9. **The Finite Element Method in Engineering** – S. S. Rao, 4th Edition, Elsevier Science & Technology Books, 2004

10. **Numerical Solutions of partial differential equations by the finite element method by the Finite Element Method** – Johnson, Claes, Cambridge University Press.

11. **Método dos Elementos Finitos em Análise de Estruturas –** Humberto Lima Soriano, Edusp, 2003.

12. **Notas de Aula (Curso de MEF)** – Luiz Eloy Vaz (DME-POLI/UFRJ).

13. **Notas de Aula (Curso de MEF)** – Silvio de Souza Lima (DME-POLI/UFRJ).

14. **Notas de Aula (Curso de MEF)** – Fernando Ribeiro (COPPE/UFRJ).

15. **Análise de Estruturas em Computadores: Estruturas Reticuladas** – Humberto Lima Soriano e Silvio de Souza Lima, 2ª Edição, Volume I.

16. **Análise de Estruturas** – Humberto Soriano e Silvio de Souza Lima, 2ª Edição, Editora Ciência Moderna.

17. **Mechanical Vibrations** – S. S. RAO, 3rd Edition, Addison-Wesley Publishing Company.

18. **Elementary Diferential Equations and Boundary Value Problems** – Boyce & Diprima, 7th Edition.

19. **Theory of Plates and Shells,** S. Timoshenko and S. Woinowsky-Krieger 2nd Edition, McGraw-Hill Book Company.

20. **Vigas, Placas e Cascas, Notas de Aula (Curso de Elasticidade II) -** R. V. Alves (DME-POLI/UFRJ).

21. **Notas de Aula (Placas e Cascas) –** Lúcia Maria Dinis (FEUP).

22. **Energy Principles and Variational Methods in Applied Mechanics** – J. N. Reedy, John Wiley and Sons, Inc.

23. **Theories and Applications of Plate Analysis** – Rudolph Szilard, John Wiley and Sons, Inc, 2004.

24. **Mecânica dos Sólidos 1 -** S. Timoshenko, Gere, Livros Técnicos e Científicos Editora.

25. **Mecânica dos Sólidos 2 -** S. Timoshenko, Gere, Livros Técnicos e Científicos Editora.

26. **Resistencia de Materiales -** S. Timoshenko, Espasa-Calpe, S.A. 1957.

27. **Notas de Aula (Métodos Aproximados em Mecânica Aplicada) -** R. V. Alves.

28. **Cálculo Numérico** – Márcia A. Gomes Ruggiero e Vera Lúcia R. Lopes, 2ª Edição, Pearson Makron Books.

29. **Direct Methods for Sparse Linear Systems –** Timothy A. Davis, Society for Industrial and Applied Mathematics Philadelphia.

30. **Using OpenMP (Portable Shared Memory Parallel Programmimg)** – Barbara Chapman, Gabriele Jost and Ruud Van Der Paas, The MIT Press.

31. **Programação OpenMP (Apostila)** – Ricardo Almeida de Mendonça Küsel, Versão: Junho/2008.

32. www.openmp.org , acessado em 19/09/2009.

33. https://computing.llnl.gov/tutorials/openMP/, acessado em 20/09/2009.

34. **Intel® Visual Fortran Compiler Help.**

35. **Intel® Math Kernel Library - Reference Manual.**

36. **Intel® Pardiso - User Guide Version 3.2 .**

37. **Release 11.0 Documentation for ANSYS.**

38. **Análise de Estruturas formadas pela associação de placas e chapas utilizando o Método dos Elementos de Contorno** - Monnerat, Daniel Dias. Teses de Msc., COPPE/UFRJ.

39. **Formulação Variacional e Aproximação por Elementos Finitos dos Modelos de Placas de Kirchhoff e Reissner-Mindlin** - Pereira, Carlos Eduardo Leite. Tese de Msc., UNICAMP.

40. **Uma Resolução de Placas com a Teoria de Mindlin através do Método dos Elementos de Contorno –** Sanches, Luis C. F.. Tese de Msc., UNICAMP.

41. **Análise da Interação Solo-Estrutura de Fundações do Tipo Estaca para Plataformas Offshore –** Aguiar, Cristiano Santos. Projeto de Tese de Doutorado, COPPE/UFRJ.

42. **Análise de Placas à Flexão pelo Método dos Elementos Finitos** - LIMA, Diego de Amorim. Projeto de Graduação, Engenharia Civil, UFRJ.

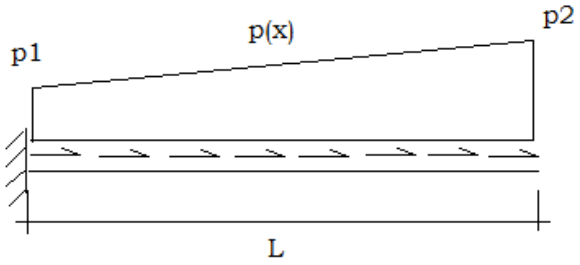
43. **Modelagem Numérica de Flexão de Placas retangulares segundo a Teoria de Kirchhoff** – Monnerat, Daniel Dias. Projeto de Graduação, Engenharia Civil, UFRJ.

ANEXOS

ANEXO A

ANÁLISE DE ELEMENTO DE 1 gdl por nó via MEF

Nesta planilha, será feito apenas o desenvolvimento de uma análise via método dos elementos finitos de uma barra axialmente carregada mostrada na figura abaixo. Os resultados deste exemplo foram apresentados no capítulo II.



p(x)=p1+(p2-p1)x/L

Equação diferencial para representação matemática do fenômeno físico:

$$E \cdot A \cdot \frac{d^2u}{dx^2} + p(x) = 0$$

Solução Analítica:

A solução analítica pode ser obtida através de integrações sucessivas e impondo as condições de contorno do problema.

$$p(x) := p1 + (p2 - p1)\cdot\frac{x}{L} \qquad u''(x) := \frac{-p(x)}{E\cdot A}$$

$$u'(x) := \int_0^x u''(x)\,dx + C1 \text{ expand} \rightarrow C1 - \frac{p1\cdot x}{A\cdot E} + \frac{p1\cdot x^2}{2\cdot A\cdot E\cdot L} - \frac{p2\cdot x^2}{2\cdot A\cdot E\cdot L}$$

$$u(x) := \int_0^x u'(x)\,dx + C2 \text{ expand} \rightarrow C2 + C1\cdot x - \frac{p1\cdot x^2}{2\cdot A\cdot E} + \frac{p1\cdot x^3}{6\cdot A\cdot E\cdot L} - \frac{p2\cdot x^3}{6\cdot A\cdot E\cdot L}$$

Condições de contorno:

Condição de contorno essencial:	Condição de contorno natural:
$u(0) = 0$	$u'(L) = 0$

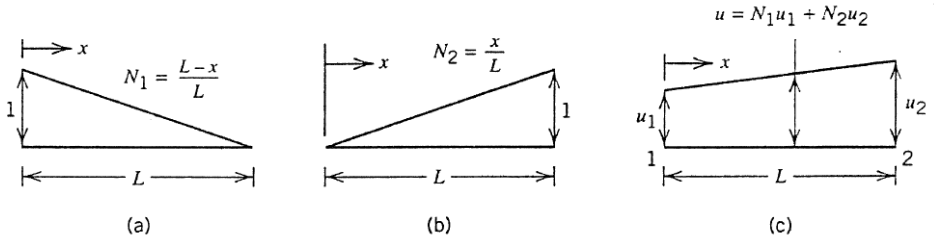
Portanto, a solução analítica pode ser expressa da seguinte forma:

$$u(x) \rightarrow x\cdot\left(\frac{L\cdot p1}{2\cdot A\cdot E}+\frac{L\cdot p2}{2\cdot A\cdot E}\right)-\frac{p1\cdot x^{2}}{2\cdot A\cdot E}+\frac{p1\cdot x^{3}}{6\cdot A\cdot E\cdot L}-\frac{p2\cdot x^{3}}{6\cdot A\cdot E\cdot L}$$

Solução Via MEF:

Solução utilizando apenas um elemento:

Usando um elemento linear de dois nós, podemos interpolar os deslocamentos no interior do elemento da seguinte forma:



$$N_1(x) := \frac{L-x}{L} \qquad N_2(x) := \frac{x}{L}$$

$$u_{mef1}(x) := u_1\cdot N_1(x) + u_2\cdot N_2(x) \rightarrow \frac{u_2\cdot x}{L}+\frac{u_1\cdot(L-x)}{L}$$

$$\varepsilon_{mef1}(x) := \frac{d}{dx}u_{mef1}(x) \rightarrow \frac{u_2}{L}-\frac{u_1}{L} \qquad \text{(Teoria da elasticidade)}$$

$$\sigma_{mef1}(x) := E\cdot\varepsilon_{mef1}(x) \rightarrow -E\cdot\left(\frac{u_1}{L}-\frac{u_2}{L}\right) \qquad \text{(Lei de Hooke)}$$

$$U := \frac{1}{2}\cdot A\cdot\int_0^L \sigma_{mef1}(x)\cdot\varepsilon_{mef1}(x)\,dx \rightarrow \frac{A\cdot E\cdot(u_1-u_2)^2}{2\cdot L} \qquad \text{(Energia Potencial de Deformação)}$$

Trabalho realizado pela carga distribuída, de acordo com o princípio dos trabalhos virtuais.

$$W := \int_0^L p(x)\cdot u_{mef1}(x)\,dx + F_1\cdot u_1 + F_2\cdot u_2$$

$$W \text{ expand } \rightarrow F_1\cdot u_1 + F_2\cdot u_2 + \frac{L\cdot p1\cdot u_1}{3} + \frac{L\cdot p1\cdot u_2}{6} + \frac{L\cdot p2\cdot u_1}{6} + \frac{L\cdot p2\cdot u_2}{3}$$

$$\Pi := U + W \qquad \text{(Energia Potencial Total)}$$

A configuração de equilíbrio representa um valor estacionário para a energia potencial, logo, podemos escrever:

$$\frac{d}{du_1}\Pi \text{ expand } \rightarrow F_1 + \frac{L\cdot p1}{3} + \frac{L\cdot p2}{6} + \frac{A\cdot E\cdot u_1}{L} - \frac{A\cdot E\cdot u_2}{L}$$

$$\frac{d}{du_2}\Pi \text{ expand } \rightarrow F_2 + \frac{L\cdot p1}{6} + \frac{L\cdot p2}{3} - \frac{A\cdot E\cdot u_1}{L} + \frac{A\cdot E\cdot u_2}{L}$$

Podemos expressar o desenvolvido acima matricialmente da seguinte forma:

Matriz de Rigidez — Vetor de forças nodais equivalentes

$$\begin{pmatrix} \frac{E\cdot A}{L} & -\frac{E\cdot A}{L} \\ -\frac{E\cdot A}{L} & \frac{E\cdot A}{L} \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} + \begin{pmatrix} \frac{L\cdot p1}{3} + \frac{L\cdot p2}{6} \\ \frac{L\cdot p1}{6} + \frac{L\cdot p2}{3} \end{pmatrix}$$

Com matriz de rigidez nessa forma não é possível computar sua inversa, pois existe deslocamento de corpo rígido, sendo necessário impor as condições de contorno.

Condição de contorno essencial:	Condição de contorno natural:
$u_1 = 0$	$F_2 = 0$

Trabalhando apenas com a segunda linha do sistema, podemos chegar a:

$$u_2 := \frac{\frac{L\cdot p1}{6} + \frac{L\cdot p2}{3}}{\frac{E\cdot A}{L}} \text{ simplify } \rightarrow \frac{L^2\cdot(p1 + 2\cdot p2)}{6\cdot A\cdot E}$$

Logo, o campo de deslocamentos aproximado pelo método dos elementos finitos fica dado por:

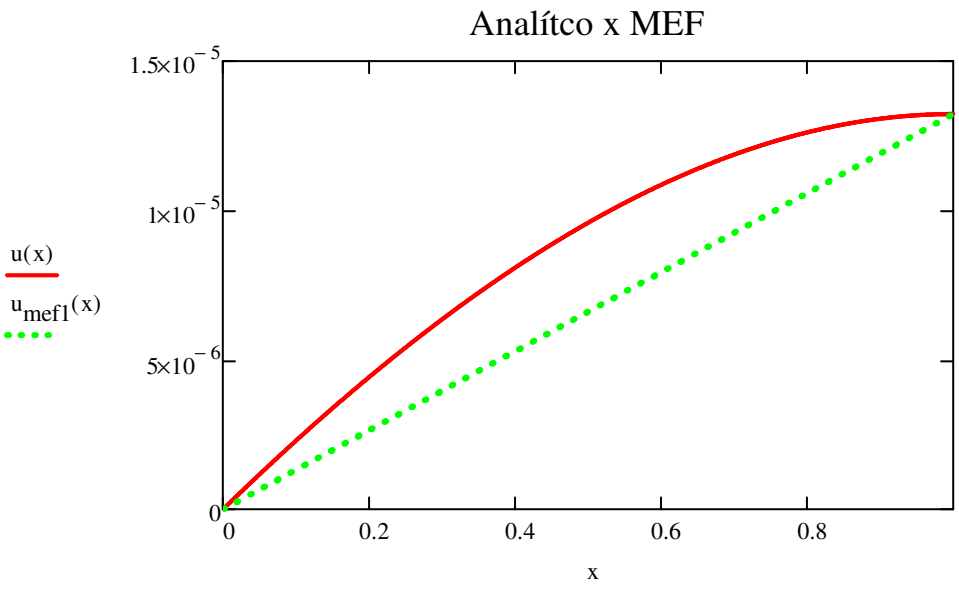
$$u_{mef1}(x) := u_1\cdot N_1(x) + u_2\cdot N_2(x) \rightarrow \frac{L\cdot x\cdot(p1 + 2\cdot p2)}{6\cdot A\cdot E}$$

Substituição de valores numéricos para avaliação dos resultados:

$E := 2.1 \cdot 10^{7}$ $A := 0.3$ $p1 := 100$ $p2 := 200$ $L := 1$

Solução Analítica: $u(x) := x \cdot \left(\frac{L \cdot p1}{2 \cdot A \cdot E} + \frac{L \cdot p2}{2 \cdot A \cdot E} \right) - \frac{p1 \cdot x^{2}}{2 \cdot A \cdot E} + \frac{p1 \cdot x^{3}}{6 \cdot A \cdot E \cdot L} - \frac{p2 \cdot x^{3}}{6 \cdot A \cdot E \cdot L}$

Solução pelo Método dos Elementos Finitos: $u_{mef1}(x) := \frac{L \cdot x \cdot (p1 + 2 \cdot p2)}{6 \cdot A \cdot E}$



Solução utilizando 4 elementos:

$p(x) := p1 + (p2 - p1) \cdot \frac{x}{L}$

$k := \frac{E \cdot A}{\frac{L}{4}}$

$u_2 := 1$ $f1 := 1$ $u_3 := 1$ $u_4 := 2$ $u_5 := 1$

Given

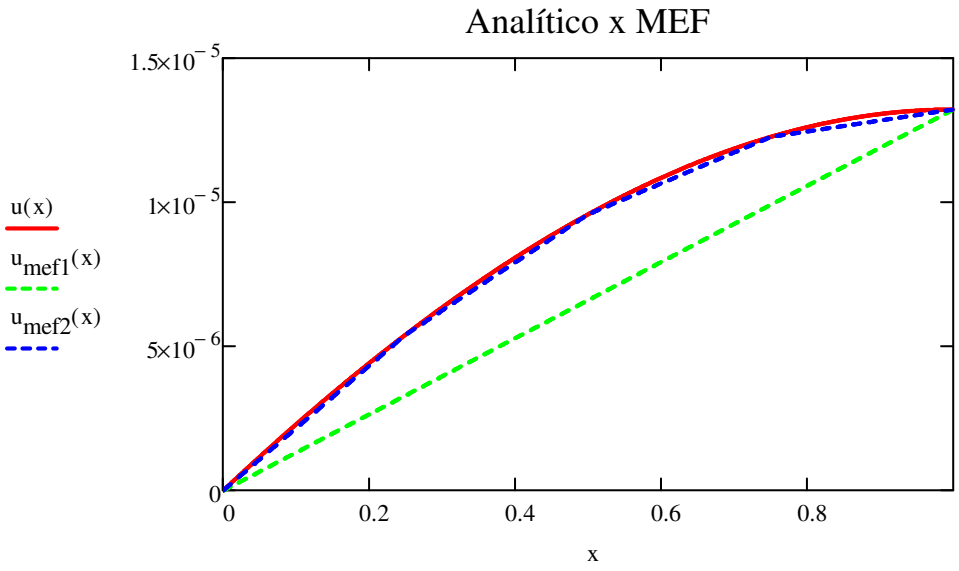
$$\begin{pmatrix} k & -k & 0 & 0 & 0 \\ -k & 2k & -k & 0 & 0 \\ 0 & -k & 2k & -k & 0 \\ 0 & 0 & -k & 2k & -k \\ 0 & 0 & 0 & -k & k \end{pmatrix} \cdot \begin{pmatrix} 0 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix} = \begin{pmatrix} f1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} \frac{p(0) \cdot 0.25}{3} + \frac{p(0.25) \cdot 0.25}{6} \\ \frac{p(0) \cdot 0.25}{6} + 2 \cdot \frac{p(0.25) \cdot 0.25}{3} + \frac{p(0.5) \cdot 0.25}{6} \\ \frac{p(0.25) \cdot 0.25}{6} + 2 \cdot \frac{p(0.5) \cdot 0.25}{3} + \frac{p(0.75) \cdot 0.25}{6} \\ \frac{p(0.5) \cdot 0.25}{6} + 2 \cdot \frac{p(0.75) \cdot 0.25}{3} + \frac{p(1) \cdot 0.25}{6} \\ \frac{p(1) \cdot 0.25}{3} + \frac{p(0.75) \cdot 0.25}{6} \end{pmatrix}$$

$$\begin{pmatrix} f1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix} := \text{Find}(f1, u_2, u_3, u_4, u_5) = \begin{pmatrix} -150 \\ 5.415 \times 10^{-6} \\ 9.59 \times 10^{-6} \\ 1.228 \times 10^{-5} \\ 1.323 \times 10^{-5} \end{pmatrix}$$

Solução do sistema de equações.

$$N_1(x) := 1 - \frac{4 \cdot x}{L} \qquad N_2(x) := \frac{4x}{L} \qquad u_1 := 0$$

$$u_{mef2}(x) := \begin{| u_{fem2} \leftarrow u_1 \cdot N_1(x) + u_2 \cdot N_2(x) \text{ if } x \geq 0 \wedge x < 0.25 \\ u_{fem2} \leftarrow u_2 \cdot N_1(x - 0.25) + u_3 \cdot N_2(x - 0.25) \text{ if } x \geq 0.25 \wedge x < 0.5 \\ u_{fem2} \leftarrow u_3 \cdot N_1(x - 0.5) + u_4 \cdot N_2(x - 0.5) \text{ if } x \geq 0.5 \wedge x < 0.75 \\ u_{fem2} \leftarrow u_4 \cdot N_1(x - 0.75) + u_5 \cdot N_2(x - 0.75) \text{ if } x \geq 0.75 \wedge x \leq 1 \\ u_{fem2}$$



ANEXO B

GERAÇÃO DE FUNÇÕES DE FORMA PARA ELEMENTOS QUADRÁTICOS

$ORIGIN := 1$

Esta planilha realiza um esquema de inclusão de nós sugerido no livro do Robert Cook, *Concepts and Applications of Finite Element Analysis*.

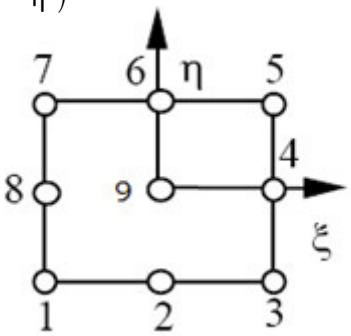
$$N1(\xi,\eta) := \frac{1}{4}(1-\xi)(1-\eta) \qquad N5(\xi,\eta) := \frac{1}{2}\left(1-\xi^2\right)(1-\eta) \qquad N9(\xi,\eta) := \left(1-\xi^2\right)\left(1-\eta^2\right)$$

$$N2(\xi,\eta) := \frac{1}{4}(1+\xi)(1-\eta) \qquad N6(\xi,\eta) := \frac{1}{2}(1+\xi)\left(1-\eta^2\right)$$

$$N3(\xi,\eta) := \frac{1}{4}\cdot(1+\xi)(1+\eta) \qquad N7(\xi,\eta) := \frac{1}{2}\cdot\left(1-\xi^2\right)(1+\eta)$$

$$N4(\xi,\eta) := \frac{1}{4}\cdot(1-\xi)(1+\eta) \qquad N8(\xi,\eta) := \frac{1}{2}\cdot(1-\xi)\left(1-\eta^2\right)$$

Seleção de nós presentes no elemento:



$$i6 := 1$$

$$i8 := 1 \quad i9 := 1 \quad i7 := 1$$

$$i5 := 1$$

$$vecN(\xi,\eta) := \begin{pmatrix} N1(\xi,\eta) \\ N2(\xi,\eta) \\ N3(\xi,\eta) \\ N4(\xi,\eta) \\ N5(\xi,\eta) \\ N6(\xi,\eta) \\ N7(\xi,\eta) \\ N8(\xi,\eta) \\ N9(\xi,\eta) \end{pmatrix}$$

$$fun := \begin{pmatrix} 1 & 0 & 0 & 0 & \frac{-1}{2}\cdot i5 & 0 & 0 & \frac{-1}{2}\cdot i8 & \frac{1}{4}\cdot i9 \\ 0 & 1 & 0 & 0 & \frac{-1}{2}\cdot i5 & \frac{-1}{2}\cdot i6 & 0 & 0 & \frac{1}{4}\cdot i9 \\ 0 & 0 & 1 & 0 & 0 & \frac{-1}{2}\cdot i6 & \frac{-1}{2}\cdot i7 & 0 & \frac{1}{4}\cdot i9 \\ 0 & 0 & 0 & 1 & 0 & 0 & \frac{-1}{2}\cdot i7 & \frac{-1}{2}\cdot i8 & \frac{1}{4}\cdot i9 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \frac{-1}{2}\cdot i9 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \frac{-1}{2}\cdot i9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{-1}{2}\cdot i9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{-1}{2}\cdot i9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\cdot i9 \end{pmatrix}$$

$$N(\xi,\eta) := fun\cdot vecN(\xi,\eta)$$

$$N(\xi,\eta)\ \text{simplify} \rightarrow \begin{bmatrix} \frac{\xi\cdot\eta\cdot(\xi-1)\cdot(\eta-1)}{4} \\ \frac{\xi\cdot\eta\cdot(\xi+1)\cdot(\eta-1)}{4} \\ \frac{\xi\cdot\eta\cdot(\xi+1)\cdot(\eta+1)}{4} \\ \frac{\xi\cdot\eta\cdot(\xi-1)\cdot(\eta+1)}{4} \\ -\frac{\eta\cdot(\eta-1)\cdot\left(\xi^2-1\right)}{2} \\ -\frac{\xi\cdot(\xi+1)\cdot\left(\eta^2-1\right)}{2} \\ -\frac{\eta\cdot(\eta+1)\cdot\left(\xi^2-1\right)}{2} \\ -\frac{\xi\cdot(\xi-1)\cdot\left(\eta^2-1\right)}{2} \\ \left(\xi^2-1\right)\cdot\left(\eta^2-1\right) \end{bmatrix}$$

$N1(\xi,\eta) := N(\xi,\eta)_1$ $N4(\xi,\eta) := N(\xi,\eta)_4$ $N7(\xi,\eta) := N(\xi,\eta)_7$

$N2(\xi,\eta) := N(\xi,\eta)_2$ $N5(\xi,\eta) := N(\xi,\eta)_5$ $N8(\xi,\eta) := N(\xi,\eta)_8$

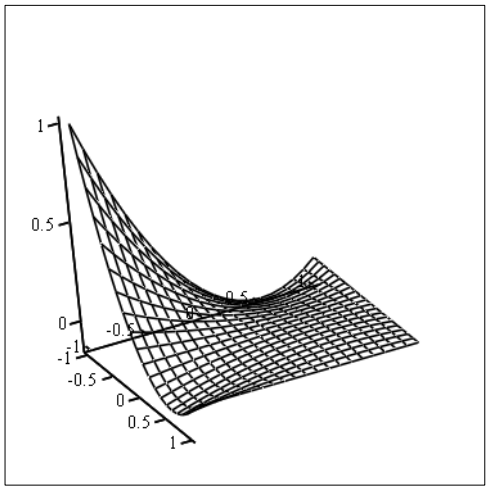
$N3(\xi,\eta) := N(\xi,\eta)_3$ $N6(\xi,\eta) := N(\xi,\eta)_6$ $N9(\xi,\eta) := N(\xi,\eta)_9$

FUNÇÃO DE FORMA N1:

$N1(-1,1) = 0$ $N1(0,1) = 0$ $N1(1,1) = 0$

$N1(-1,0) = 0$ $N1(0,0) = 0$ $N1(1,0) = 0$

$N1(-1,-1) = 1$ $N1(0,-1) = 0$ $N1(1,-1) = 0$



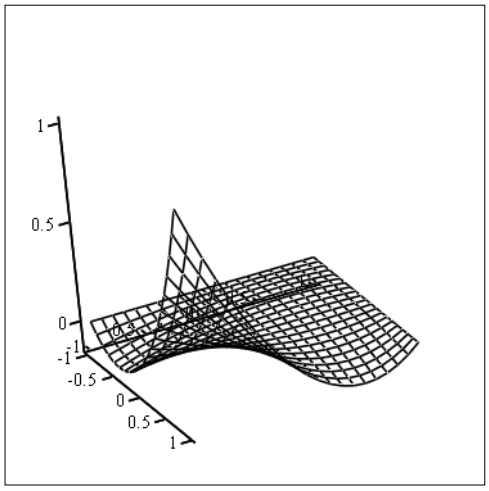
N1

FUNÇÃO DE FORMA N2:

$N2(-1,1) = 0 \quad N2(0,1) = 0 \quad N2(1,1) = 0$

$N2(-1,0) = 0 \quad N2(0,0) = 0 \quad N2(1,0) = 0$

$N2(-1,-1) = 0 \quad N2(0,-1) = 0 \quad N2(1,-1) = 1$



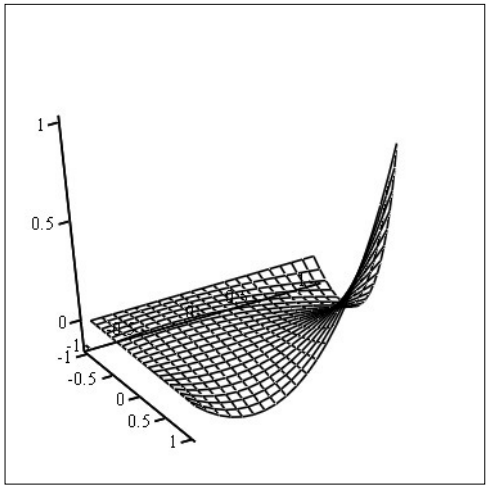
N2

FUNÇÃO DE FORMA N3:

$N3(-1,1) = 0 \quad N3(0,1) = 0 \quad N3(1,1) = 1$

$N3(-1,0) = 0 \quad N3(0,0) = 0 \quad N3(1,0) = 0$

$N3(-1,-1) = 0 \quad N3(0,-1) = 0 \quad N3(1,-1) = 0$



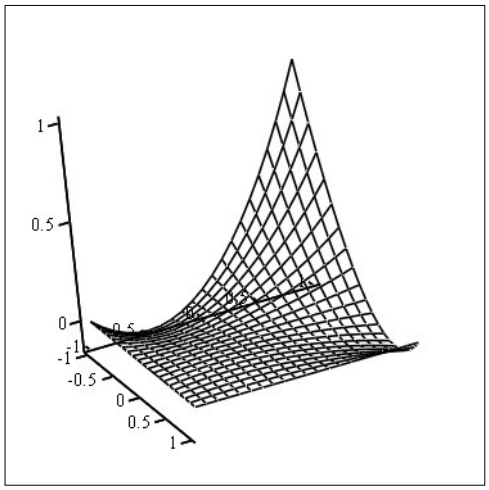
N3

FUNÇÃO DE FORMA N4:

$N4(-1,1) = 1 \quad N4(0,1) = 0 \quad N4(1,1) = 0$

$N4(-1,0) = 0 \quad N4(0,0) = 0 \quad N4(1,0) = 0$

$N4(-1,-1) = 0 \quad N4(0,-1) = 0 \quad N4(1,-1) = 0$



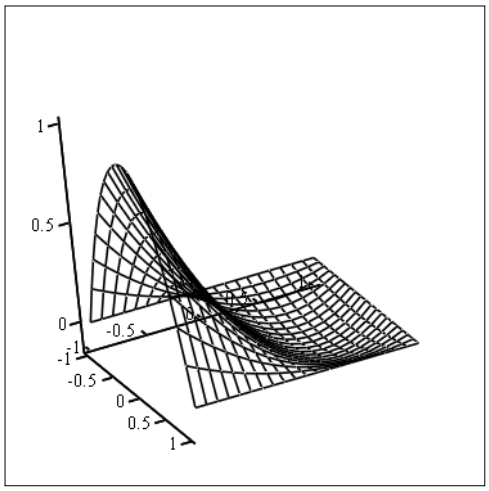
N4

FUNÇÃO DE FORMA N5:

$N5(-1,1) = 0 \quad N5(0,1) = 0 \quad N5(1,1) = 0$

$N1(-1,0) = 0 \quad N5(0,0) = 0 \quad N5(1,0) = 0$

$N5(-1,-1) = 0 \quad N5(0,-1) = 1 \quad N5(1,-1) = 0$



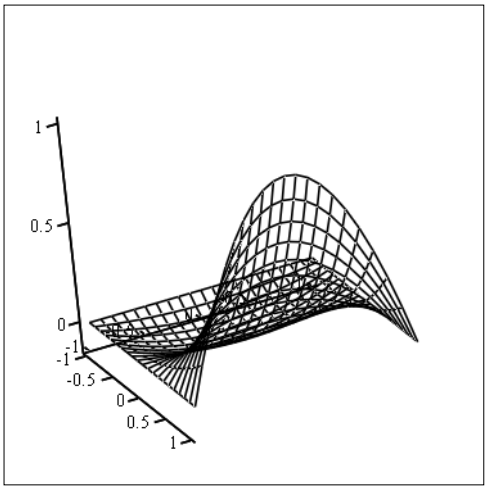
N5

FUNÇÃO DE FORMA N6:

$N6(-1,1) = 0 \quad N6(0,1) = 0 \quad N6(1,1) = 0$

$N6(-1,0) = 0 \quad N6(0,0) = 0 \quad N6(1,0) = 1$

$N6(-1,-1) = 0 \quad N6(0,-1) = 0 \quad N6(1,-1) = 0$



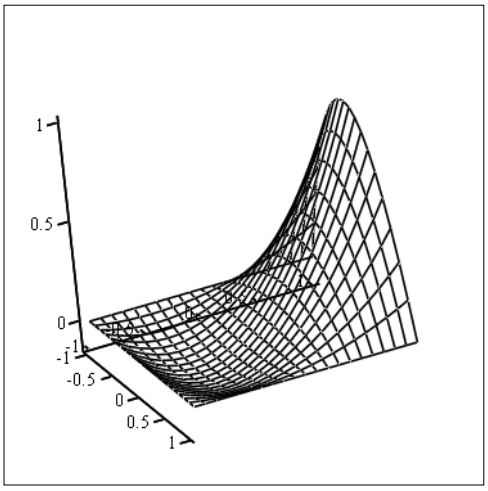
N6

FUNÇÃO DE FORMA N7:

$N7(-1,1) = 0 \quad N7(0,1) = 1 \quad N7(1,1) = 0$

$N7(-1,0) = 0 \quad N7(0,0) = 0 \quad N7(1,0) = 0$

$N7(-1,-1) = 0 \quad N7(0,-1) = 0 \quad N7(1,-1) = 0$



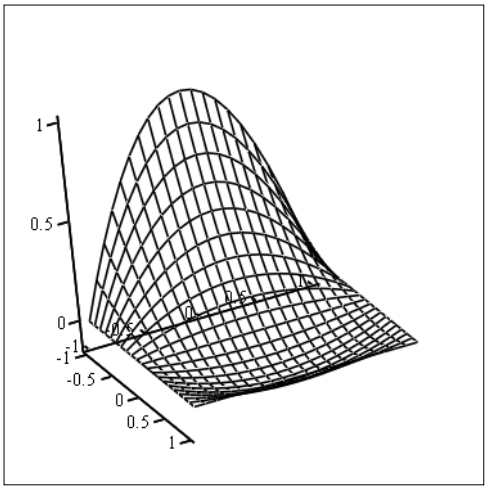
N7

FUNÇÃO DE FORMA N8:

$N8(-1,1) = 0 \quad N8(0,1) = 0 \quad N8(1,1) = 0$

$N8(-1,0) = 1 \quad N8(0,0) = 0 \quad N8(1,0) = 0$

$N8(-1,-1) = 0 \quad N8(0,-1) = 0 \quad N8(1,-1) = 0$



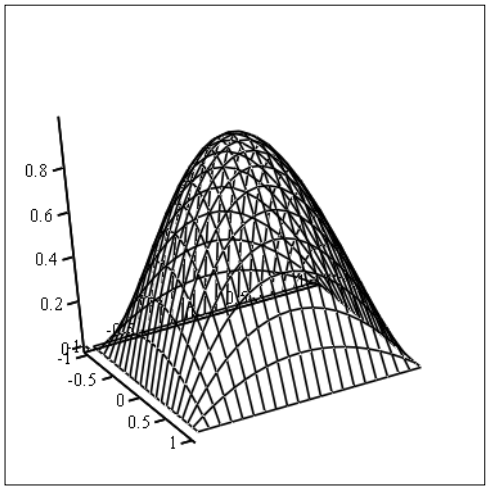
N8

FUNÇÃO DE FORMA N9:

$N9(-1,1) = 0 \quad N9(0,1) = 0 \quad N9(1,1) = 0$

$N9(-1,0) = 0 \quad N9(0,0) = 1 \quad N9(1,0) = 0$

$N9(-1,-1) = 0 \quad N9(0,-1) = 0 \quad N9(1,-1) = 0$



N9

Obtenção das derivadas das funções de forma:

$$dNd\xi(\xi,\eta) := \frac{d}{d\xi}N(\xi,\eta) \qquad dNd\eta(\xi,\eta) := \frac{d}{d\eta}N(\xi,\eta)$$

$$dNd\xi(\xi,\eta)\ \text{simplify} \rightarrow \begin{bmatrix} \frac{\eta\cdot(\eta-1)\cdot(2\cdot\xi-1)}{4} \\ \frac{\eta\cdot(\eta-1)\cdot(2\cdot\xi+1)}{4} \\ \frac{\eta\cdot(\eta+1)\cdot(2\cdot\xi+1)}{4} \\ \frac{\eta\cdot(\eta+1)\cdot(2\cdot\xi-1)}{4} \\ -\xi\cdot\eta\cdot(\eta-1) \\ -\frac{(2\cdot\xi+1)\cdot\left(\eta^2-1\right)}{2} \\ -\xi\cdot\eta\cdot(\eta+1) \\ -\frac{(2\cdot\xi-1)\cdot\left(\eta^2-1\right)}{2} \\ 2\cdot\xi\cdot\left(\eta^2-1\right) \end{bmatrix} \qquad dNd\eta(\xi,\eta)\ \text{simplify} \rightarrow \begin{bmatrix} \frac{\xi\cdot(\xi-1)\cdot(2\cdot\eta-1)}{4} \\ \frac{\xi\cdot(\xi+1)\cdot(2\cdot\eta-1)}{4} \\ \frac{\xi\cdot(\xi+1)\cdot(2\cdot\eta+1)}{4} \\ \frac{\xi\cdot(\xi-1)\cdot(2\cdot\eta+1)}{4} \\ -\frac{(2\cdot\eta-1)\cdot\left(\xi^2-1\right)}{2} \\ -\xi\cdot\eta\cdot(\xi+1) \\ -\frac{(2\cdot\eta+1)\cdot\left(\xi^2-1\right)}{2} \\ -\xi\cdot\eta\cdot(\xi-1) \\ 2\cdot\eta\cdot\left(\xi^2-1\right) \end{bmatrix}$$

ANEXO C

ANÁLISE MODAL DE ELEMENTO DE MOLA E VIGA

Esta planilha avalia os autovalores e autovetores da matriz de rigidez de um elemento de mola, de um elemento de viga e de um elemento de pórtico plano.

Análise Modal da Matriz de Rigidez de Elemento de Mola:

$$k_{mola} := \begin{pmatrix} \frac{E \cdot A}{L} & -\frac{E \cdot A}{L} \\ -\frac{E \cdot A}{L} & \frac{E \cdot A}{L} \end{pmatrix}$$

Matriz de rigidez de um elemento de mola linear.

Cálculo dos autovalores da matriz:

$$\text{eigenvals}(k_{mola}) \rightarrow \begin{pmatrix} \frac{2 \cdot A \cdot E}{L} \\ 0 \end{pmatrix}$$

=> O Autovalor se associa à rigidez para um determinado modo de deformação.

=> O Autovalor nulo está associado à um deslocamento de corpo rígido.

Cálculo dos autovetores da matriz:

$$\text{eigenvec}\left(k_{mola}, \frac{2 \cdot A \cdot E}{L}\right) \rightarrow \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

=> Autovetor associado ao primeiro autovalor (não nulo)

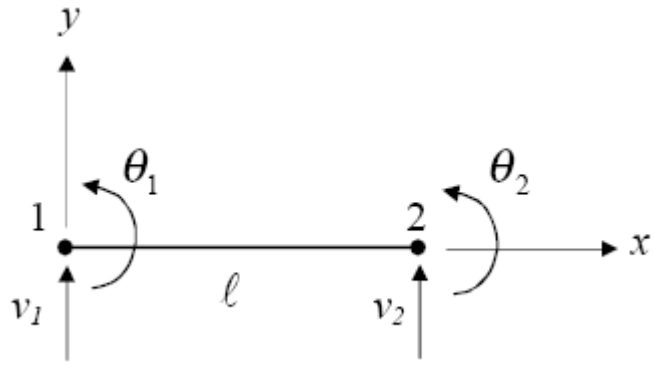
$$\text{eigenvec}(k_{mola}, 0) \rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

=> Autovetor associado ao segundo autovalor (nulo), que corresponde à um deslocamento de corpo rígido.

Cálculo da energia de deformação para cada modo:

$$\begin{pmatrix} -1 \\ 1 \end{pmatrix}^T \cdot k_{mola} \cdot \begin{pmatrix} -1 \\ 1 \end{pmatrix} \rightarrow \frac{4 \cdot A \cdot E}{L} \qquad \begin{pmatrix} 1 \\ 1 \end{pmatrix}^T \cdot k_{mola} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} \rightarrow 0$$

Análise Modal da Matriz de Rigidez de Elemento de Viga:



$$k_{viga} := \begin{pmatrix} \frac{12\cdot E\cdot I}{L^3} & \frac{6\cdot E\cdot I}{L^2} & -\frac{12\cdot E\cdot I}{L^3} & \frac{6\cdot E\cdot I}{L^2} \\ \frac{6\cdot E\cdot I}{L^2} & \frac{4\cdot E\cdot I}{L} & -\frac{6\cdot E\cdot I}{L^2} & \frac{2\cdot E\cdot I}{L} \\ -\frac{12\cdot E\cdot I}{L^3} & -\frac{6\cdot E\cdot I}{L^2} & \frac{12\cdot E\cdot I}{L^3} & -\frac{6\cdot E\cdot I}{L^2} \\ \frac{6\cdot E\cdot I}{L^2} & \frac{2\cdot E\cdot I}{L} & -\frac{6\cdot E\cdot I}{L^2} & \frac{4\cdot E\cdot I}{L} \end{pmatrix}$$

Matriz de Rigidez de um Elemento de Viga.

Cálculo dos autovalores da matriz:

$$\text{eigenvals}(k_{viga}) \rightarrow \begin{pmatrix} \frac{2\cdot E\cdot I}{L} \\ 0 \\ 0 \\ \frac{6\cdot E\cdot I\cdot L^2 + 24\cdot E\cdot I}{L^3} \end{pmatrix}$$

=> Dois Autovalores nulo associados à deslocamentos de corpo rígido.

Cálculo dos autovetores da matriz:

$$\text{eigenvec}\left(k_{viga}, \frac{2\cdot E\cdot I}{L}\right) \rightarrow \begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \end{pmatrix}$$

=> Autovetor associado ao primeiro autovalor (não nulo)

$$\text{eigenvec}(k_{viga}, 0) \rightarrow \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

=> Autovetor associado ao segundo autovalor (nulo), que corresponde à um deslocamento de corpo rígido.

$$\text{eigenvec}\left(k_{viga}, \frac{6\cdot E\cdot I\cdot L^2 + 24\cdot E\cdot I}{L^3}\right) \rightarrow \begin{pmatrix} \frac{2}{L} \\ 1 \\ -\frac{2}{L} \\ 1 \end{pmatrix}$$

=> Autovetor associado ao segundo autovalor (nulo), que corresponde à um deslocamento de corpo rígido.

$$\text{eigenvecs}\left(k_{viga}\right) \rightarrow \begin{pmatrix} 0 & 1 & -L & \frac{2}{L} \\ -1 & 0 & 1 & 1 \\ 0 & 1 & 0 & -\frac{2}{L} \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

=> Usando a função para pedir todos os autovalores é possível retirar aquele que faltava.

Cálculo da energia de deformação para cada modo:

$$\begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \end{pmatrix}^T \cdot k_{viga} \cdot \begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \end{pmatrix} \rightarrow \frac{4\cdot E\cdot I}{L} \qquad \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}^T \cdot k_{viga} \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \rightarrow 0$$

Energia de deformação nula (desl. de corpo rígido)

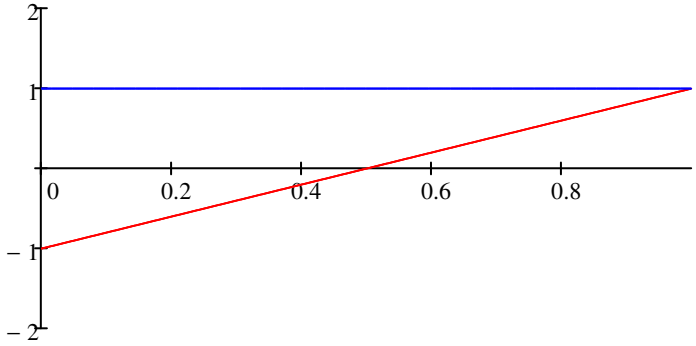
$$\begin{pmatrix} \frac{2}{L} \\ 1 \\ -\frac{2}{L} \\ 1 \end{pmatrix}^T \cdot k_{viga} \cdot \begin{pmatrix} \frac{2}{L} \\ 1 \\ -\frac{2}{L} \\ 1 \end{pmatrix} \text{ simplify } \rightarrow \frac{12\cdot E\cdot I\cdot\left(L^2+4\right)^2}{L^5} \qquad \begin{pmatrix} -L \\ 1 \\ 0 \\ 1 \end{pmatrix}^T \cdot k_{viga} \cdot \begin{pmatrix} -L \\ 1 \\ 0 \\ 1 \end{pmatrix} \rightarrow 0$$

Análise Gráfica dos Modos de Deformação:

Elemento de mola:

$L := 1 \qquad N_1(x) := \frac{L-x}{L} \qquad N_2(x) := \frac{x}{L}$ (funções de forma)

$\text{modo1}_{mola}(x) := -1\cdot N_1(x) + 1\cdot N_2(x) \qquad \text{modo2}_{mola}(x) := 1\cdot N_1(x) + 1\cdot N_2(x)$



=> O segundo modo corresponde a um deslocamento de corpo rígido.

Primeiro Modo
Segundo Modo

Elemento de viga:

$$N1(x) := \frac{2\cdot x^3}{L^3} - \frac{3\cdot x^2}{L^2} + 1 \qquad N2(x) := x - \frac{2\cdot x^2}{L} + \frac{x^3}{L^2} \qquad N3(x) := \frac{3\cdot x^2}{L^2} - \frac{2\cdot x^3}{L^3} \qquad N4(x) := \frac{x^3}{L^2} - \frac{x^2}{L}$$

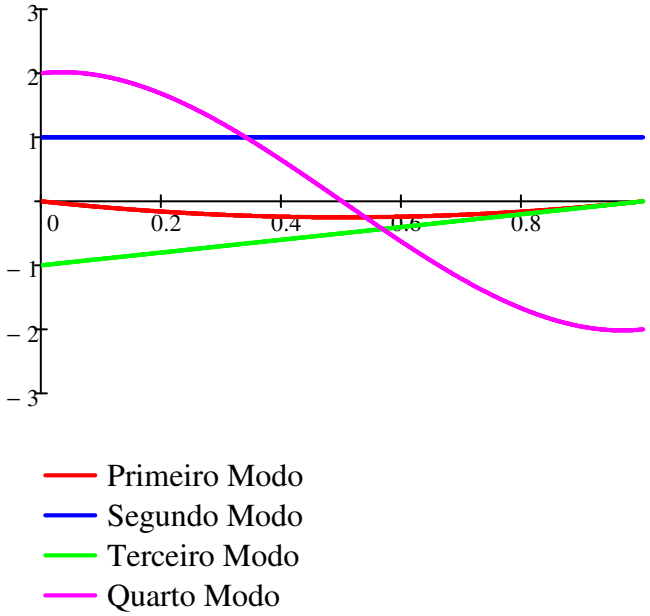
(funções de forma de viga)

$$modo1_{viga}(x) := 0\cdot N1(x) + -1\cdot N2(x) + 0\cdot N3(x) + 1\cdot N4(x)$$

$$modo2_{viga}(x) := 1\cdot N1(x) + 0\cdot N2(x) + 1\cdot N3(x) + 0\cdot N4(x)$$

$$modo3_{viga}(x) := -L\cdot N1(x) + 1\cdot N2(x) + 0\cdot N3(x) + 1\cdot N4(x)$$

$$modo4_{viga}(x) := \frac{2}{L}\cdot N1(x) + 1\cdot N2(x) + -\frac{2}{L}\cdot N3(x) + 1\cdot N4(x)$$



=> O segundo e o terceiro modo correspondem a deslocamentos de corpo rígido.

ANEXO D

ANÁLISE MODAL DE ELEMENTO BI-LINEAR DE PLACA ESPESSA

ORIGIN := 1

Dados de entrada:

Espessura, mod. elastcidade e poisson: $t := 0.2$ $E := 2 \times 10^7$ $\nu := 0.3$

nnodes := 4 nelem := 1

gdl := 3·nnodes = 12

i := 1 .. gdl j := 1 .. gdl $dG_i := 0$

i := 1 .. nnodes

Coordenadas Nodais:

$$x := \begin{pmatrix} -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \end{pmatrix}$$

Matriz de incidência:

$$inc := \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

$$C_b := \frac{E}{1-\nu^2}\begin{pmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{pmatrix} \qquad C_s := \frac{E}{2\cdot(1+\nu)}\cdot\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Funções de Forma (em coordenadas paramétricas ξ e η):

$$N1(\xi,\eta) := \frac{1}{4}\cdot(1-\xi)\cdot(1-\eta) \qquad N3(\xi,\eta) := \frac{1}{4}\cdot(1+\xi)\cdot(1+\eta)$$

$$N2(\xi,\eta) := \frac{1}{4}\cdot(1+\xi)\cdot(1-\eta) \qquad N4(\xi,\eta) := \frac{1}{4}\cdot(1-\xi)\cdot(1+\eta)$$

Derivadas das funções de forma em relação às coordenadas paramétricas:

$$N1\xi(\xi,\eta) := \frac{d}{d\xi}N1(\xi,\eta) \qquad N1\eta(\xi,\eta) := \frac{d}{d\eta}N1(\xi,\eta)$$

$$N2\xi(\xi,\eta) := \frac{d}{d\xi}N2(\xi,\eta) \qquad N2\eta(\xi,\eta) := \frac{d}{d\eta}N2(\xi,\eta)$$

$$N3\xi(\xi,\eta) := \frac{d}{d\xi}N3(\xi,\eta) \qquad N3\eta(\xi,\eta) := \frac{d}{d\eta}N3(\xi,\eta)$$

$$N4\xi(\xi,\eta) := \frac{d}{d\xi}N4(\xi,\eta) \qquad N4\eta(\xi,\eta) := \frac{d}{d\eta}N4(\xi,\eta)$$

Cálculo numérico da área do elemento finito com uma Quadratura de Gauss 3x3:

Coordenadas paramétricas dos pontos de Gauss:

2 pontos

$$\xi g := \begin{pmatrix} \frac{-1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{pmatrix} \quad \eta g := \begin{pmatrix} \frac{-1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{pmatrix}$$

1 ponto

$$cood := 0$$

Pesos associados aos pontos de Gauss:

2 pontos

$$W\xi := \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad W\eta := \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

1 ponto

$$peso := 2$$

Matriz de rigidez:

$$Xe := x^T$$

$$DN(\xi,\eta) := \begin{pmatrix} N1\xi(\xi,\eta) & N2\xi(\xi,\eta) & N3\xi(\xi,\eta) & N4\xi(\xi,\eta) \\ N1\eta(\xi,\eta) & N2\eta(\xi,\eta) & N3\eta(\xi,\eta) & N4\eta(\xi,\eta) \end{pmatrix}$$

Matriz do Jacobiano:

$$J(\xi,\eta) := DN(\xi,\eta)\cdot Xe$$

$$J(\xi,\eta) \rightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$ => Identidade

Determinante:

$$detJ(\xi,\eta) := |J(\xi,\eta)|$$

$$invJ(\xi,\eta) := \frac{1}{detJ(\xi,\eta)} \cdot \begin{pmatrix} J(\xi,\eta)_{2,2} & -J(\xi,\eta)_{1,2} \\ -J(\xi,\eta)_{2,1} & J(\xi,\eta)_{1,1} \end{pmatrix}$$

$$N1dxdy(\xi,\eta) := invJ(\xi,\eta)\cdot \begin{pmatrix} N1\xi(\xi,\eta) \\ N1\eta(\xi,\eta) \end{pmatrix}$$

$$N1x(\xi,\eta) := N1dxdy(\xi,\eta)_1 \qquad N1y(\xi,\eta) := N1dxdy(\xi,\eta)_2$$

$$N2dxdy(\xi,\eta) := invJ(\xi,\eta)\cdot \begin{pmatrix} N2\xi(\xi,\eta) \\ N2\eta(\xi,\eta) \end{pmatrix}$$

$$N2x(\xi,\eta) := N2dxdy(\xi,\eta)_1 \qquad N2y(\xi,\eta) := N2dxdy(\xi,\eta)_2$$

$$N3dxdy(\xi,\eta) := invJ(\xi,\eta)\cdot \begin{pmatrix} N3\xi(\xi,\eta) \\ N3\eta(\xi,\eta) \end{pmatrix}$$

$$N3x(\xi,\eta) := N3dxdy(\xi,\eta)_1 \qquad N3y(\xi,\eta) := N3dxdy(\xi,\eta)_2$$

$$N4dxdy(\xi,\eta) := invJ(\xi,\eta)\cdot \begin{pmatrix} N4\xi(\xi,\eta) \\ N4\eta(\xi,\eta) \end{pmatrix}$$

$$N4x(\xi,\eta) := N4dxdy(\xi,\eta)_1 \qquad N4y(\xi,\eta) := N4dxdy(\xi,\eta)_2$$

Matriz "B" de deformação (compatibilidade cinemática):

$$B_b(\xi,\eta) := \begin{pmatrix} 0 & 0 & -N1x(\xi,\eta) & 0 & 0 & -N2x(\xi,\eta) & 0 & 0 & -N3x(\xi,\eta) & 0 & 0 & -N4x(\xi,\eta) \\ 0 & -N1y(\xi,\eta) & 0 & 0 & -N2y(\xi,\eta) & 0 & 0 & -N3y(\xi,\eta) & 0 & 0 & -N4y(\xi,\eta) & 0 \\ 0 & -N1x(\xi,\eta) & -N1y(\xi,\eta) & 0 & -N2x(\xi,\eta) & -N2y(\xi,\eta) & 0 & -N3x(\xi,\eta) & -N3y(\xi,\eta) & 0 & -N4x(\xi,\eta) & -N4y(\xi,\eta) \end{pmatrix}$$

$$B_s(\xi,\eta) := \begin{pmatrix} N1x(\xi,\eta) & 0 & -N1(\xi,\eta) & N2x(\xi,\eta) & 0 & -N2(\xi,\eta) & N3x(\xi,\eta) & 0 & -N3(\xi,\eta) & N4x(\xi,\eta) & 0 \\ N1y(\xi,\eta) & -N1(\xi,\eta) & 0 & N2y(\xi,\eta) & -N2(\xi,\eta) & 0 & N3y(\xi,\eta) & -N3(\xi,\eta) & 0 & N4y(\xi,\eta) & -N4(\xi,\eta) \end{pmatrix}$$

$k_s := \frac{5}{6}$ (fator de correção)

$$D_b := \frac{t^3}{12} \cdot C_b = \begin{pmatrix} 14652.015 & 4395.604 & 0 \\ 4395.604 & 14652.015 & 0 \\ 0 & 0 & 5128.205 \end{pmatrix} \qquad D_s := k_s \cdot t \cdot C_s = \begin{pmatrix} 1282051.282 & 0 \\ 0 & 1282051.282 \end{pmatrix}$$

Verificação da Área do Elemento:

$$A := \sum_{i=1}^{2} \sum_{j=1}^{2} \left(\det J\left(\xi g_i, \eta g_j\right) \cdot W\xi_i \cdot W\eta_j \right) \qquad A2 := \det J(cood, cood) \cdot peso \cdot peso$$

$A = 4$ $\qquad$ $A2 = 4$

Avaliação das matrizes de rigidez numericamente:

$$[k] = \int_A [B]^T \cdot [D] \cdot [B] dA = \int_{-1}^{1} \int_{-1}^{1} [B]^T \cdot [D] \cdot [B] \cdot |J| \, d\xi \, d\eta$$

$KB_{red}(m) := B_b(cood, cood)^T \cdot D_b \cdot B_b(cood, cood) \cdot \det J(cood, cood) \cdot peso \cdot peso$ — Matriz de Flexão com integração reduzida

$$KB_{comp}(m) := \sum_{i=1}^{2} \sum_{j=1}^{2} \left(B_b\left(\xi g_i, \eta g_j\right)^T \cdot D_b \cdot B_b\left(\xi g_i, \eta g_j\right) \cdot \det J\left(\xi g_i, \eta g_j\right) \cdot W\xi_i \cdot W\eta_j \right)$$ — Matriz de Flexão com integração completa

$KS_{red}(m) := B_s(cood, cood)^T \cdot D_s \cdot B_s(cood, cood) \cdot \det J(cood, cood) \cdot peso \cdot peso$ — Matriz de Cisalhamento com integração reduzida

$$KS_{comp}(m) := \sum_{i=1}^{2} \sum_{j=1}^{2} \left(B_s\left(\xi g_i, \eta g_j\right)^T \cdot D_s \cdot B_s\left(\xi g_i, \eta g_j\right) \cdot \det J\left(\xi g_i, \eta g_j\right) \cdot W\xi_i \cdot W\eta_j \right)$$ — Matriz de Cisalhamento com integração completa

Reduzida	Seletiva	Completa
$K_{red}(m) := KB_{red}(m) + KS_{red}(m)$	$K_{sel}(m) := KB_{comp}(m) + KS_{red}(m)$	$K_{comp}(m) := KB_{comp}(m) + KS_{comp}(m)$

Matriz de Rigidez com Integração Reduzida:

$K_{red}(1) =$

	1	2	3	4	5	6	7
1	641025.641	320512.821	320512.821	0	320512.821	320512.821	-641025.641
2	320512.821	325457.875	2380.952	320512.821	322893.773	183.15	-320512.821
3	320512.821	2380.952	325457.875	-320512.821	-183.15	318131.868	-320512.821
4	0	320512.821	-320512.821	641025.641	320512.821	-320512.821	0
5	320512.821	322893.773	-183.15	320512.821	325457.875	-2380.952	-320512.821
6	320512.821	183.15	318131.868	-320512.821	-2380.952	325457.875	-320512.821
7	-641025.641	-320512.821	-320512.821	0	-320512.821	-320512.821	641025.641
8	320512.821	315567.766	-2380.952	320512.821	318131.868	-183.15	-320512.821
9	320512.821	-2380.952	315567.766	-320512.821	183.15	322893.773	-320512.821
10	0	-320512.821	320512.821	-641025.641	-320512.821	320512.821	0
11	320512.821	318131.868	183.15	320512.821	315567.766	2380.952	-320512.821
12	320512.821	-183.15	322893.773	-320512.821	2380.952	315567.766	...

Matriz de Rigidez com Integração Seletiva:

$K_{sel}(1) =$

	1	2	3	4	5	6	7
1	641025.641	320512.821	320512.821	0	320512.821	320512.821	-641025.641
2	320512.821	327106.227	2380.952	320512.821	321245.421	183.15	-320512.821
3	320512.821	2380.952	327106.227	-320512.821	-183.15	316483.516	-320512.821
4	0	320512.821	-320512.821	641025.641	320512.821	-320512.821	0
5	320512.821	321245.421	-183.15	320512.821	327106.227	-2380.952	-320512.821
6	320512.821	183.15	316483.516	-320512.821	-2380.952	327106.227	-320512.821
7	-641025.641	-320512.821	-320512.821	0	-320512.821	-320512.821	641025.641
8	320512.821	317216.117	-2380.952	320512.821	316483.516	-183.15	-320512.821
9	320512.821	-2380.952	317216.117	-320512.821	183.15	321245.421	-320512.821
10	0	-320512.821	320512.821	-641025.641	-320512.821	320512.821	0
11	320512.821	316483.516	183.15	320512.821	317216.117	2380.952	-320512.821
12	320512.821	-183.15	321245.421	-320512.821	2380.952	317216.117	...

Matriz de Rigidez com Integração Completa:

$K_{comp}(1) =$

	1	2	3	4	5	6	7
1	854700.855	427350.427	427350.427	-213675.214	213675.214	427350.427	-427350.427
2	427350.427	576393.976	2380.952	213675.214	285632.886	183.15	-213675.214
3	427350.427	2380.952	576393.976	-427350.427	-183.15	280870.981	-213675.214
4	-213675.214	213675.214	-427350.427	854700.855	427350.427	-427350.427	-213675.214
5	213675.214	285632.886	-183.15	427350.427	576393.976	-2380.952	-427350.427
6	427350.427	183.15	280870.981	-427350.427	-2380.952	576393.976	-213675.214
7	-427350.427	-213675.214	-213675.214	-213675.214	-427350.427	-213675.214	854700.855
8	213675.214	139153.439	-2380.952	427350.427	280870.981	-183.15	-427350.427
9	213675.214	-2380.952	139153.439	-213675.214	183.15	285632.886	-427350.427
10	-213675.214	-427350.427	213675.214	-427350.427	-213675.214	213675.214	-213675.214
11	427350.427	280870.981	183.15	213675.214	139153.439	2380.952	-213675.214
12	213675.214	-183.15	285632.886	-213675.214	2380.952	139153.439	...

ANÁLISE DOS AUTOVALORES DAS MATRIZES DE RIGIDEZ:

$\text{eigenvals}(K_{red}(1)) =$

	1
1	-0
2	2564102.564
3	2564102.564
4	19047.619
5	10256.41
6	10256.41
7	0
8	-0
9	0
10	-0+0i
11	-0-0i
12	-0

7 auto-valores nulos
3 associados a deslocamentos de corpo rígido
4 modos espúrios

$\text{eigenvals}(K_{sel}(1)) =$

	1
1	-0
2	2564102.564
3	2564102.564
4	19047.619
5	10256.41
6	10256.41
7	6593.407
8	6593.407
9	-0
10	-0
11	-0
12	-0

5 auto-valores nulos
3 associados a deslocamentos de corpo rígido
2 modos espúrios

$\text{eigenvals}(K_{comp}(1)) =$

	1
1	1285488.368
2	2564102.564
3	2564102.564
4	6819.325
5	427350.427
6	446398.046
7	437606.838
8	149043.549
9	149043.549
10	-0+0i
11	-0-0i
12	0

3 auto-valores nulos associados a deslocamentos de corpo rígido

---

A N E X O E

- Listagem do código computacional.

```
!======================================================================
!
! Programa: PLACA_elem_quadratico
!
! Descrição: Esta programa realiza análise elástica linear de placa
!            através do método dos elementos finitos c/ elemento de placa
!            espessa (mindlin) quadrático de 9 nós (lagrange).
!
! Autor:     Ricardo Caldeira de Oliveira
!
! Comentários:
!         Programa realizado no projeto final de curso do aluno Ricardo
!         Caldeira de Oliveira sob orientação do professor José Antônio
!         Fontes Santiago.
!
!======================================================================
Program PLACA_elem_quadratico

!CNO   -> COORDENADA DOS NOS
!INCID ->NUMERAÇÃO NODAL DE CADA ELEMENTO E SUAS PROPRIEDADES
!NNO   -> NÚMERO DE NÓS
!NELM  -> NÚMERO DE ELEMENTOS
!PROP  -> PROPRIEDADES DA SEÇÃO
!NPROP -> NUMERO DE PRORPIEDADES
!F     -> VETOR FORÇA GLOBAL
!SG    -> MATRIZ DE ELEMENTO NO SISTEMA GLOBAL
!a     -> VETOR DE TRABALHO QUE ARMAZENA A MATRIZ DE RIGIDEZ
!p     -> VETOR APONTADOR AUXILIAR
!NCONT -> NÚMERO DE NÓS COM CONDIÇÕES DE CONTORNO
!CCONT -> MATRIZ QUE ARMAZENA AS CONDIÇÕES DE CONTORNO

integer::i,NNO,NELM,NPROP,NNOCG,NDIST,NCONT,saida,saida2,entrada,job(6)
integer,allocatable:: p(:),INCID(:,:),CCONT(:,:),pnt(:),ia(:),ja(:)
real*8 , allocatable::
CNO(:,:),PROP(:,:),a(:),F(:),CDIST(:,:),DESL(:),acsr(:)
real*8 :: NG,tempo,tempototal
CHARACTER*24 FILEIN,FILEOUT,temp
CHARACTER*1 FILEIO(20)
logical lparal
!======================================================================
==================
!LEITURA E ABERTURA DOS ARQUIVOS DE ENTRADA E SAÍDA
!======================================================================
==================

call TITULO
WRITE(*,'(a,\)') " NOME DO ARQUIVO DE ENTRADA(.inp, sem a extensao): "
READ(*,'(20A1)') (FILEIO(I),I=1,20)
! FILEIO(1)='6'
! FILEIO(2)='4'
! FILEIO(3)='e'
! FILEIO(4)='l'
! FILEIO(5)='_'
! FILEIO(6)='r'
! FILEIO(7)='2'
! FILEIO(8)='0'
! FILEIO(9)='0'
! do i =10,20;  FILEIO(i)=' '; enddo;
tempototal= secnds(0.0)
 Do i = 1, 20
   If (FILEIO(i).EQ.' ') Exit
 End Do
```

```
FILEIO(i)  ='.'
FILEIO(i+1)='i'
FILEIO(i+2)='n'
FILEIO(i+3)='p'

Write(FILEIN,'(24A1)') (FILEIO(j),j=1,i+3)
temp= "temp.tmp"
!chamada de subrotina para criacao de arquivo temporário sem comentários
call gera_temp (FILEIN,temp)
entrada=10
saida=20
saida2=30

open(unit=entrada,file=temp,status='old',action='read',iostat=ierror)
!
FILEIO(i)='.'
FILEIO(i+1)='o'
FILEIO(i+2)='u'
FILEIO(i+3)='t'
Write(FILEOUT,'(24A1)') (FILEIO(j),j=1,i+5)
open(unit=saida,file=FILEOUT,status='replace',action='write',iostat=ierror)
Write(*,'(/," Arquivos:"/,4x,"Entrada: ",a,/,4x,"Saida  : ",a,/)')
FILEIN,FILEOUT

FILEIO(i)='.'
FILEIO(i+1)='r'
FILEIO(i+2)='i'
FILEIO(i+3)='g'
Write(FILEOUT,'(24A1)') (FILEIO(j),j=1,i+5)
open(unit=saida2,file=FILEOUT,status='replace',action='write',iostat=
ierror)

!=============================================================================
==================
!LEITURA DOS DADOS
!=============================================================================
==================

!LEITURA DOS NÓS  E SUAS COORDENADAS

READ(entrada,*)NNO
allocate (CNO(NNO,2))
        Do i=1,NNO
                READ(entrada,*)j,CNO(j,1),CNO(j,2)
        end do

allocate (p(3*nno)) !VETOR APONTADOR = 3*NUMERO DE NOS
p=0
!LEITURA DA PROPRIEDADE DAS SEÇÕES ---- E,POISSON,thickness

READ(entrada,*)NPROP
allocate (PROP(NPROP,3))
        Do i=1,NPROP
                READ(entrada,*)j,(PROP(j,k),k=1,3)
        end do

!LEITURA DOS ELEMENTOS E SEUS NÓS ----  Nº , NO 1 , NO 2 , NO 3 , NO 4 ,
PROPRIEDADE

READ(entrada,*)NELM
allocate (INCID(NELM,10))
        Do i=1,NELM
                READ(entrada,*)
                j,INCID(j,1),INCID(j,2),INCID(j,3),INCID(j,4),INCID(j,5),IN
                CID(j,6)&
,INCID(j,7),INCID(j,8),INCID(j,9),INCID(j,10)
        end do

!LEITURA DAS CONDIÇÕES DE CONTORNO

READ(entrada,*)NCONT
allocate (CCONT(NCONT,4))
        DO i=1,NCONT
```

```
                READ(entrada,*) (CCONT(i,j),j=1,4)
        end do

!LEITURA DAS CARGAS NOS NÓS ----  N° , NO INICIAL , NO FINAL , PROPRIEDADE
allocate (F(NNO*3),DESL(NNO*3))

        F=0.d0

READ(entrada,*)NNOCG
if (NNOCG.NE.0) then
        Do i=1,NNOCG
                READ(entrada,*)j,F((j)*3-2),F((j)*3-1),F((j)*3)
        end do
end if

!LEITURA DAS CARGAS DISTRIBUÍDAS NOS ELEMENTOS

READ(entrada,*)NDIST
allocate (CDIST(NDIST,2))
        Do i=1,NDIST
                READ(entrada,*)CDIST(i,1),CDIST(i,2)
        end do


call APONTADOR(p,INCID,NNO,NELM)

allocate (a(p(3*NNO)))
write(*,*) " ...Montando a Matriz de Rigidez e Vetor de Forcas Nodais"
call MONTAMATRIZ(p,a,CNO,INCID,PROP,CDIST,F,NNO,NELM,NPROP,NDIST)
call DADOS
(CCONT,CNO,INCID,PROP,CDIST,F,saida,NNO,NELM,NPROP,NNOCG,NDIST,NCONT)
call CONTORNO(a,p,CCONT,NG,NNO,NCONT)
!call SAIDAMATRIZ (a,p,saida2,NNO)
write(*,*) " ...Resolvendo o sistema linear"


tempo= secnds(0.0)
lparal=.true.
if (lparal) then
    DESL=0.d0
    allocate (ia(3*nno+1),pnt(3*nno+1),acsr(p(3*NNO)),ja(p(3*NNO)))
    !acsr=0.d0
    call pnt_creator(p,pnt,nno)
    ngdl=3*NNO
    job(1) = 1; job(2) = 1; job(3) = 1; job(4) = 1;
  !  call mkl_dskysv('t',ngdl,1.d0,'tuu',a,pnt,F,DESL)
    call mkl_dcsrsky(job,ngdl,acsr,ja,ia,a,pnt,info);
    !do iteste=1,p(3*NNO); write(*,*) acsr(iteste); enddo;
    call
    solve_pardiso(ngdl,p(3*NNO),ia,ja,acsr,F,DESL,.true.,.true.,.false.,2)
    call DESLOCAMENTOS (DESL,saida,NNO)
    call REACOES(DESL,CCONT,saida,NG,NNO,NCONT)
else
    call CHOLESKY(a,p,F,NNO)
    call DESLOCAMENTOS (F,saida,NNO)
    call REACOES(F,CCONT,saida,NG,NNO,NCONT)
end if
tempo= secnds(tempo)
write ( *, * ) 'Elapsed time solving system = ', tempo

write(*,*) " ...fim"
CLOSE(ENTRADA,STATUS="delete")
CLOSE(SAIDA)
tempototal= secnds(tempototal)
write ( *, * ) 'Elapsed time running all = ', tempototal
PAUSE
stop
END !FIM DO PROGRAMA PRINCIPAL



!=====================================================================
!
```

```
! Subrotina: Titulo
!
! Descrição: Esta subrotina imprime o título na tela
!
! Autor:     Ricardo Caldeira de Oliveira
!
! Comentários:
!
!=====================================================================
Subroutine TITULO
write(*,*)'*********************************************************************
**************'
write(*,*)
write(*,*)'      P R O G R A M A  D E  A N A L I S E  E L A S T I C A  L I
N E A R'
write(*,*)
write(*,*)'                    D E      P L A C A      E S P E S S A'
write(*,*)
write(*,*)
write(*,*)'      V I A  M E F ( T E O R I A  D E  R E I S S N E R - M I N D
L I N)'
write(*,*)
write(*,*)'*********************************************************************
**************'
write(*,'(//)')
RETURN
end

!=====================================================================
!
! Subrotina: Geratemp
!
! Descrição: Esta subrotina limpa o arquivo de entrada e gera o arquivo
!            de trabalho sem comentários "temp.tmp"
!
! Autor:     Ricardo Caldeira de Oliveira
!
! Comentários:
!
!=====================================================================
SUBROUTINE gera_temp (arquivo1,arquivo2)
CHARACTER*24 arquivo2
CHARACTER*24 arquivo1
CHARACTER*256 linha

OPEN (UNIT=100, FILE=TRIM(arquivo1), STATUS='OLD')
OPEN (UNIT=101, FILE=TRIM(arquivo2))

DO WHILE (.not. eof(100))
        READ (100,'(A256)') linha
        IF (linha(1:1) .eq. "#") cycle
        WRITE(101,'(A256)') linha
END DO
CLOSE(100)
CLOSE(101)
RETURN
END

!=====================================================================
!
! Subrotina: Dados
!
! Descrição: Esta subrotina é responsável apenas pela impressão dos dados
!            de entrada no arquivo de saída ".sai"
!
! Autor:     Ricardo Caldeira de Oliveira
!
! Comentários:
!
!=====================================================================
subroutine DADOS
(CCONT,CNO,INCID,PROP,CDIST,F,saida,NNO,NELM,NPROP,NNOCG,NDIST,NCONT)
integer::i,j,saida,CCONT(NCONT,4),INCID(NELM,5)
```

```
real*8 ::CNO(NNO,2),PROP(NPROP,3),CDIST(NDIST,2),F(3*NNO)


write(saida,*)'UNIVERSIDADE FEDERAL DO RIO DE JANEIRO'
write(saida,*)'ESCOLA POLITÉCNICA'
write(saida,*)'ENGENHARIA CIVIL'
write(saida,*)
write(saida,*)
write(saida,*)'************************************************************
******************'
write(saida,*)'                                D A D O S'
write(saida,*)'************************************************************
******************'
write(saida,*)
write(saida,*)'      c o o r d e n a d a s   n o d a i s'
write(saida,*)'      nó   sistema     coordenada     coordenada'
write(saida,*)'                            x              y'
do i=1,NNO
        write(saida,'(6x,i3,4x,a6,7x,f5.2,9x,f5.2)')
        i,'global',CNO(i,1),CNO(i,2)
end do
write(saida,'(/,6x,a,i3,2/)')'                                        número de
nós ......',NNO
write(saida,*)'      r e s t r i ç õ e s   n o d a i s'
write(saida,*)'       nó    código'
do i=1,NCONT
        write(saida,'(6x,i3,6x,i1,i1,i1)')
        CCONT(i,1),CCONT(i,2),CCONT(i,3),CCONT(i,4)
end do
write(saida,'(/,6x,a,i3,2/)')'                    número de nós com
restrição ......',NCONT
write(saida,*)'      p r o p r i e d a d e s   d o s   e l e m e n t o s'
write(saida,*)'      elem  propriedades     nó 1       nó 2       nó 3
nó 4  '
do i=1,NELM
        write(saida,'(6x,i3,10x,i3,8x,i3,8x,i3,8x,i3,8x,i3)')
        i,INCID(i,5),INCID(i,1),INCID(i,2),INCID(i,3),INCID(i,4)
end do
write(saida,'(/,6x,a,i3,2/)')'                                 número de
elementos ......',NELM
write(saida,*)'      p r o p r i e d a d e s   d a s   s e ç õ e s'
write(saida,*)'      material        E     POISSON    ESPESSURA'
do i=1,NPROP

        write(saida,'(9x,i3,7x,e8.2,8x,e8.2,3x,e8.2,3x,e8.2,9x,f5.2,6x,f4.2
        )')i,PROP(i,1),PROP(i,2),PROP(i,3)
end do
write(saida,'(/,6x,a,i3,2/)')'                                    número de tipos
......',NPROP

write(saida,*)

if (NDIST.NE.0) then
        write(saida,*)'      c a r g a s  d i s t r i b u i d a s  n o s  e
        l e m e n t o s'
        write(saida,*)'      elem         p'
        do i=1,NDIST
                write(saida,'(8x,i4,5x,f7.2,5x,f7.2)')
                int(CDIST(i,1)),CDIST(i,2)
        end do
        write(saida,*)
end if

if (NNOCG.NE.0) then
        write(saida,*)'      c a r g a s  n o d a i s'
        write(saida,*)'      nó     força      momento      momento'
        write(saida,*)'               z           xx           yy'
        do i=1,NNOCG
                do j=1,3*NNO,3
                        if(F(j).NE.0) then

                                write(saida,'(6x,i3,4x,f6.2,6x,f6.2,6x,f6.2
                                )')(j-1)/3+1,F(j),F(j+1),F(j+2)
```

```
                        end if
                end do
        end do
        write(saida,*)
end if

RETURN
end


!=====================================================================
!
! Subrotina: Apontador
!
! Descrição: Esta subrotina monta o vetor apontador dos elementos
!            da diagonal principal da matriz de rigidez, que para este
!            programa está armazenada em perfil (tipo: Skyline)
!
! Autor:     Ricardo Caldeira de Oliveira
!
! Comentários:
!
!=====================================================================
subroutine APONTADOR(p,INCID,NNO,NELM)
integer i,j,k,noi,d,l,m,INCID(NELM,10),p(3*NNO)

do i=1,3*NNO
        p(i)=1
end do

! Calculo das aluras de colunas
do i=1,NELM
        noi=1
        iaux=INCID(i,1)
        do j=2,9
        if (INCID(i,j)<iaux) then
        noi = j
        iaux= INCID(i,j)
        endif
        enddo
        l=3*(INCID(i,noi)-1)+1
        do j=1,9
                do k=1,3
                        m=3*(INCID(i,j)-1)+k
                        d=m-l+1
                        if (p(m).lt.d) then
                        p(m)=d
                        endif
                end do
        end do
end do

! Calculo das posiçoes do elemento da diagonal principal
do i=2,3*NNO
        p(i)=p(i-1)+p(i)
end do

RETURN
END


!=====================================================================
!
! Subrotina: Montamatriz
!
! Descrição: Esta subrotina chama a geração das matrizes de rigidez dos
!            elementos e as espalha no vetor de trabalho global A
!            (tipo skyline)
!
! Autor:     Ricardo Caldeira de Oliveira
!
! Comentários:
!
```

```
!=====================================================================
subroutine MONTAMATRIZ(p,a,CNO,INCID,PROP,CDIST,F,NNO,NELM,NPROP,NDIST)
use omp_lib
INTEGER i,j,k,z,m,l,p(3*NNO),INCID(NELM,10),q(27)
real*8
SG(27,27),a(p(3*NNO)),CNO(NNO,2),PROP(NPROP,3),CDIST(NDIST,2),F(NNO*3)
real*8 tempo

do i=1,p(3*NNO)
        a(i)=0.d0
end do
q=0

tempo= secnds(0.0)

nthreads = omp_get_num_procs()
write (*,*) 'There are', nthreads, 'threads available'

call OMP_SET_NUM_THREADS(2)

nthreads = 2!OMP_GET_MAX_THREADS()
write (*,*) 'O Loop sera paralelizado com ', nthreads, ' threads'


 SG= 0.d0

!$OMP PARALLEL DO DEFAULT(SHARED) PRIVATE(id,I,J,K,Z,M,L,q)
FIRSTPRIVATE(SG) SCHEDULE(STATIC) IF(nthreads.GT.1)
do i=1,NELM
 !!  id = omp_get_thread_num()
 !!  write(*,*) 'Thread', id,' iteration',i
 !!  write(*,*) i
        CALL MAT_ELM (i,INCID,SG,CNO,PROP,CDIST,F,NNO,NELM,NPROP,NDIST)
        z=0
        do j=1,9
                do k=1,3
                        z=z+1
                        m=3*INCID(i,j)-3+k
                        q(z)=m
                end do
        end do
        do j=1,27
                do k=1,27
                        if (q(k)>=q(j)) then
                                l=p(q(k))+q(j)-q(k)
!$OMP ATOMIC
                a(l)=a(l)+SG(j,k)
                        end if
                end do
        end do
end do
!$OMP END PARALLEL DO
tempo= secnds(tempo)
write ( *, * ) 'Elapsed time storing element matrices = ', tempo

RETURN
END


!=====================================================================
!
! Subrotina: Matelm
!
! Descrição: Esta subrotina é responsável pela montagem da matriz de
!            rigidez de cada elemento de placa quadrático (lagrange)
!
! Autor:     Ricardo Caldeira de Oliveira
!
! Comentários:
!
!=====================================================================
SUBROUTINE MAT_ELM (elm,INCID,SG,CNO,PROP,CDIST,F,NNO,NELM,NPROP,NDIST)
integer::elm,i,j,INCID(NELM,10)
```

```
real*8
AREA,SG(27,27),CNO(NNO,2),PROP(NPROP,3),SLS(27,27),SLB(27,27),SL(27,27),XEL
EM(9,2),CDIST(NDIST,2),F(NNO*3)
real*8
JACOBI(2,2),INVJ(2,2),PTO(2),PESO(2),DN(2,9),Bb(3,27),BbT(27,3),xi,eta,aux(
27,3),aux2(27,27)
real*8
dN1xi,dN1eta,dN2xi,dN2eta,dN3xi,dN3eta,dN4xi,dN4eta,Bs(2,27),BsT(27,2),aux3
(27,2),aux4(27,27)
real*8
dN5xi,dN5eta,dN6xi,dN6eta,dN7xi,dN7eta,dN8xi,dN8eta,dN9xi,dN9eta,Db(3,3),Ds
(2,2)
real*8
dN1x,dN1y,dN2x,dN2y,dN3x,dN3y,dN4x,dN4y,N1,N2,N3,N4,intn1,intn2,intn3,intn4
real*8
dN5x,dN5y,dN6x,dN6y,dN7x,dN7y,dN8x,dN8y,dN9x,dN9y,N5,N6,N7,N8,N9,intn5,intn
6,intn7,intn8,intn9


   SL    = 0.d0
   SLS    = 0.d0
   SLB    = 0.d0
   SG    = 0.d0
   XELEM = 0.d0
   JACOBI= 0.d0
   DN    = 0.d0
   Bb    = 0.d0
   BbT   = 0.d0
   Bs    = 0.d0
   BsT   = 0.d0
   Db=0.d0 ; Ds=0.d0
   AREA  = 0.d0
   intn1= 0.d0; intn2= 0.d0; intn3=0.d0; intn4=0.d0; intn5=0.d0; intn6=
   0.d0; intn7=0.d0; intn8=0.d0; intn9=0.d0

!CALCULO DAS MATRIZES CONSTITUTIVAS

  Db(1,1)= 1                        ;Db(1,2)= PROP(INCID(elm,10),2)
  Db(2,1)= PROP(INCID(elm,10),2)  ;Db(2,2)= 1
  Db(3,3)= (1-PROP(INCID(elm,10),2))/2
  Db     = Db * PROP(INCID(elm,10),1) / (1 - PROP(INCID(elm,10),2)**2)
  *(PROP(INCID(elm,10),3)**3)/12


  Ds(1,1)= 1 ; Ds(2,2)=1
  Ds     = Ds * PROP(INCID(elm,10),1) / (2*(1+PROP(INCID(elm,10),2))) *
  PROP(INCID(elm,10),3) * 5 / 6


Do i=1,2
    do j=1,9
    XELEM(j,i) = CNO(INCID(elm,j),i)
    enddo
Enddo

   PTO(1)  = -1/SQRT(3.) ; PTO(2) =  1/SQRT(3.)
   PESO(1)  =  1.         ;PESO(2) =  1.

!Avaliação da Matriz de Rigidez de Flexão
DO i=1,2
  DO j=1,2
    xi = PTO(i)
        eta= PTO(j)

    dN1xi =  eta/4 - 0.5*xi*(eta-1) - 0.25*eta*eta + 0.5*xi*(eta*eta-1)
    dN1eta =  xi/4 + 0.5*eta*(xi*xi-1) - 0.25*xi*xi- 0.5*eta*(xi-1)
    dN2xi = -eta/4 - 0.5*xi*(eta-1) + 0.5*xi*(eta*eta-1) + 0.25*eta*eta
    dN2eta = -xi/4 + 0.5*eta*(xi+1) + 0.5*eta*(xi*xi-1) -0.25*xi*xi
    dN3xi =  eta/4 + 0.5*xi*(eta+1) + 0.25*eta*eta + 0.5*xi*(eta*eta-1)
    dN3eta =  xi/4 + 0.25*xi*xi + 0.5*eta*(xi+1) +0.5*eta*(xi*xi-1)
    dN4xi = -eta/4 + 0.5*xi*(eta+1) - 0.25*eta*eta +0.5*xi*(eta*eta-1)
    dN4eta = -xi/4 + 0.5*eta*(xi*xi-1) + 0.25 *xi*xi - 0.5*eta*(xi-1)
        dN5xi = xi*(eta-1) - xi*(eta*eta-1)
```

```
      dN5eta = 0.5*xi*xi - eta*(xi*xi-1) - 0.5
      dN6xi = 0.5 - xi*(eta*eta-1) - 0.5*eta*eta
      dN6eta =-eta*(xi+1) - eta*(xi*xi-1)
      dN7xi = -xi*(eta+1) - xi*(eta*eta-1)
      dN7eta = 0.5 - eta*(xi*xi-1) - 0.5*xi*xi
      dN8xi = 0.5*eta*eta - xi*(eta*eta-1) -0.5
      dN8eta = eta*(xi-1) - eta*(xi*xi-1)
      dN9xi = 2*xi*(eta*eta-1)
      dN9eta = 2*eta*(xi*xi -1)

      N9   = (1-xi*xi) * (1-eta*eta)
      N5   = 0.5*(1-xi *xi )*(1-eta    ) - 0.5*N9
      N6   = 0.5*(1+xi     )*(1-eta*eta) - 0.5*N9
      N7   = 0.5*(1-xi *xi )*(1+eta    ) - 0.5*N9
      N8   = 0.5*(1-xi     )*(1-eta*eta) - 0.5*N9
      N1   = (1 - xi )*(1 - eta)/4 - 0.5*N5 - 0.5*N8 - 0.25*N9
      N2   = (1 + xi )*(1 - eta)/4 - 0.5*N5 - 0.5*N6 - 0.25*N9
      N3   = (1 + xi )*(1 + eta)/4 - 0.5*N6 - 0.5*N7 - 0.25*N9
      N4   = (1 - xi )*(1 + eta)/4 - 0.5*N7 - 0.5*N8 - 0.25*N9
    DN(1,1)= dN1xi ; DN(1,2)= dN2xi ; DN(1,3)= dN3xi ; DN(1,4)= dN4xi ;
    DN(1,5)= dN5xi ; DN(1,6)= dN6xi ; DN(1,7)= dN7xi ; DN(1,8)= dN8xi ;
    DN(1,9)= dN9xi ;
    DN(2,1)= dN1eta; DN(2,2)= dN2eta; DN(2,3)= dN3eta; DN(2,4)= dN4eta;
    DN(2,5)= dN5eta; DN(2,6)= dN6eta; DN(2,7)= dN7eta; DN(2,8)= dN8eta;
    DN(2,9)= dN9eta;

    CALL MULTIPLICA(DN,XELEM,JACOBI,2,9,2)
      DETJ = JACOBI(1,1)*JACOBI(2,2) - JACOBI(1,2)*JACOBI(2,1)
      INVJ(1,1) = JACOBI(2,2) ; INVJ(2,2) = JACOBI(1,1)
      INVJ(1,2) =-JACOBI(1,2) ; INVJ(2,1) =-JACOBI(2,1)
      INVJ = INVJ / DETJ
    dN1x = INVJ(1,1)*dN1xi + INVJ(1,2)*dN1eta
    dN1y = INVJ(2,1)*dN1xi + INVJ(2,2)*dN1eta
    dN2x = INVJ(1,1)*dN2xi + INVJ(1,2)*dN2eta
    dN2y = INVJ(2,1)*dN2xi + INVJ(2,2)*dN2eta
    dN3x = INVJ(1,1)*dN3xi + INVJ(1,2)*dN3eta
    dN3y = INVJ(2,1)*dN3xi + INVJ(2,2)*dN3eta
    dN4x = INVJ(1,1)*dN4xi + INVJ(1,2)*dN4eta
    dN4y = INVJ(2,1)*dN4xi + INVJ(2,2)*dN4eta
      dN5x = INVJ(1,1)*dN5xi + INVJ(1,2)*dN5eta
    dN5y = INVJ(2,1)*dN5xi + INVJ(2,2)*dN5eta
    dN6x = INVJ(1,1)*dN6xi + INVJ(1,2)*dN6eta
    dN6y = INVJ(2,1)*dN6xi + INVJ(2,2)*dN6eta
    dN7x = INVJ(1,1)*dN7xi + INVJ(1,2)*dN7eta
    dN7y = INVJ(2,1)*dN7xi + INVJ(2,2)*dN7eta
    dN8x = INVJ(1,1)*dN8xi + INVJ(1,2)*dN8eta
    dN8y = INVJ(2,1)*dN8xi + INVJ(2,2)*dN8eta
      dN9x = INVJ(1,1)*dN9xi + INVJ(1,2)*dN9eta
    dN9y = INVJ(2,1)*dN9xi + INVJ(2,2)*dN9eta

      Bb(1,3)=  dN1x ;Bb(1,6)= dN2x ;Bb(1,9)= dN3x ;Bb(1,12)= dN4x
      ;Bb(1,15)=  dN5x ;Bb(1,18)= dN6x ;Bb(1,21)= dN7x ;Bb(1,24)= dN8x
      ;Bb(1,27)= dN9x
      Bb(2,2)=  dN1y ;Bb(2,5)= dN2y ;Bb(2,8)= dN3y ;Bb(2,11)= dN4y
      ;Bb(2,14)=  dN5y ;Bb(2,17)= dN6y ;Bb(2,20)= dN7y ;Bb(2,23)= dN8y
      ;Bb(2,26)= dN9y
      Bb(3,2)=  dN1x ;Bb(3,5)= dN2x ;Bb(3,8)= dN3x ;Bb(3,11)= dN4x
      ;Bb(3,14)=  dN5x ;Bb(3,17)= dN6x ;Bb(3,20)= dN7x ;Bb(3,23)= dN8x
      ;Bb(3,26)= dN9x
      Bb(3,3)=  dN1y ;Bb(3,6)= dN2y ;Bb(3,9)= dN3y ;Bb(3,12)= dN4y
      ;Bb(3,15)=  dN5y ;Bb(3,18)= dN6y ;Bb(3,21)= dN7y ;Bb(3,24)= dN8y
      ;Bb(3,27)= dN9y
      CALL TRANSPOSTA(Bb,BbT,3,27)
      CALL MULTIPLICA(BbT,Db,aux,27,3,3)
      CALL MULTIPLICA(aux,Bb,aux2,27,3,27)
      SLB = SLB +aux2*DETJ*PESO(i)*PESO(j)
      INTN1 = INTN1 + N1*DETJ*PESO(i)*PESO(j)
      INTN2 = INTN2 + N2*DETJ*PESO(i)*PESO(j)
      INTN3 = INTN3 + N3*DETJ*PESO(i)*PESO(j)
      INTN4 = INTN4 + N4*DETJ*PESO(i)*PESO(j)
      INTN5 = INTN5 + N5*DETJ*PESO(i)*PESO(j)
      INTN6 = INTN6 + N6*DETJ*PESO(i)*PESO(j)
      INTN7 = INTN7 + N7*DETJ*PESO(i)*PESO(j)
```

```
        INTN8 = INTN8 + N8*DETJ*PESO(i)*PESO(j)
        INTN9 = INTN9 + N9*DETJ*PESO(i)*PESO(j)

        AREA = AREA +DETJ*PESO(i)*PESO(j)
        Bs(1,1)= dN1x ; Bs(1,3)=-N1 ; Bs(1,4)= dN2x ; Bs(1,6)=-N2 ;
        Bs(1,7)= dN3x ; Bs(1,9)=-N3 ; Bs(1,10)= dN4x ; Bs(1,12)=-N4 ;
        Bs(1,13)= dN5x ; Bs(1,15)=-N5 ; Bs(1,16)= dN6x ; Bs(1,18)=-N6 ;
        Bs(1,19)= dN7x ; Bs(1,21)=-N7 ; Bs(1,22)= dN8x ; Bs(1,24)=-N8 ;
        Bs(1,25)= dN9x ; Bs(1,27)=-N9
    Bs(2,1)= dN1y ; Bs(2,2)=-N1 ; Bs(2,4)= dN2y ; Bs(2,5)=-N2 ; Bs(2,7)=
    dN3y ; Bs(2,8)=-N3 ; Bs(2,10)= dN4y ; Bs(2,11)=-N4 ; Bs(2,13)= dN5y ;
    Bs(2,14)=-N5 ; Bs(2,16)= dN6y ; Bs(2,17)=-N6 ; Bs(2,19)= dN7y ;
    Bs(2,20)=-N7 ; Bs(2,22)= dN8y ; Bs(2,23)=-N8 ; Bs(2,25)= dN9y ;
    Bs(2,26)=-N9
        CALL TRANSPOSTA(Bs,BsT,2,27)
        CALL MULTIPLICA(BsT,Ds,aux3,27,2,2)
        CALL MULTIPLICA(aux3,Bs,aux4,27,2,27)
        SLS = SLS + aux4*DETJ*PESO(i)*PESO(j)
  END DO
END DO


if (NDIST.NE.0) then
  f(3*INCID(elm,1)-2) = CDIST(elm,2)*INTN1 + f(3*INCID(elm,1)-2)
  f(3*INCID(elm,2)-2) = CDIST(elm,2)*INTN2 + f(3*INCID(elm,2)-2)
  f(3*INCID(elm,3)-2) = CDIST(elm,2)*INTN3 + f(3*INCID(elm,3)-2)
  f(3*INCID(elm,4)-2) = CDIST(elm,2)*INTN4 + f(3*INCID(elm,4)-2)
  f(3*INCID(elm,5)-2) = CDIST(elm,2)*INTN5 + f(3*INCID(elm,5)-2)
  f(3*INCID(elm,6)-2) = CDIST(elm,2)*INTN6 + f(3*INCID(elm,6)-2)
  f(3*INCID(elm,7)-2) = CDIST(elm,2)*INTN7 + f(3*INCID(elm,7)-2)
  f(3*INCID(elm,8)-2) = CDIST(elm,2)*INTN8 + f(3*INCID(elm,8)-2)
  f(3*INCID(elm,9)-2) = CDIST(elm,2)*INTN9 + f(3*INCID(elm,9)-2)
end if

SG=SLS+SLB

RETURN
END


!=====================================================================
!
! Subrotina: Contorno
!
! Descrição: Esta subrotina impõe as condições de contorno pelo método
!            do número grande
!
! Autor:     Ricardo Caldeira de Oliveira
!
! Comentários:
!
!=====================================================================
subroutine CONTORNO(a,p,CCONT,NG,NNO,NCONT)
integer i,j,p(3*NNO),CCONT(NCONT,4)
real*8 NG,a(p(3*NNO))

NG=0

NG= 10e+20

do i=1,NCONT
        do j=1,3
                a(p(3*(CCONT(i,1)-1)+j))=a(p(3*(CCONT(i,1)-1)+j))
                +CCONT(i,j+1)*NG
        end do
end do

RETURN
END


!=====================================================================
!
```

```
! Subrotina: SAIDAMATRIZ
!
! Descrição: Esta subrotina imprime o vetor de trabalho A e vetor
!            apontador P no arquivo ".rig" para controle
!
!
! Autor:     Ricardo Caldeira de Oliveira
!
! Comentários:
!
!=====================================================================
subroutine SAIDAMATRIZ (a,p,saida2,NNO)
integer::p(3*NNO),saida2
real*8 ::a(p(3*NNO))

write(saida2,*)'************************************************************
******************'
write(saida2,*)'                                V E T O R E S'
write(saida2,*)'************************************************************
******************'
write(saida2,*)
write(saida2,*)'      Vetor Apontador'
write(saida2,*)
do j=1,NNO
        write(saida2,'(6x,a3,i6,a2,i6,2x,a6,i6,a2,i6,2xa3,i6,a2,i6)') '
        p(',(j-1)*3+1,')=',p((j-1)*3+1),' p(',(j-1)*3+2,')=',p((j-1)*3+2),'
        p(',(j-1)*3+3,')=',p((j-1)*3+3)
end do

write(saida2,*)
write(saida2,*)'      Matriz de Rigidez - Vetor de Trabalho'
write(saida2,*)
do j=1,p(3*NNO)/3

        write(saida2,'(6x,a3,i6,a2,e12.6,2x,a3,i6,a2,e12.6,2x,a3,i6,a2,e12.
        6)') ' a(',(j-1)*3+1,')=',a((j-1)*3+1),' a(',(j-1)*3+2,')=',a((j-1)
        *3+2),' a(',(j-1)*3+3,')=',a((j-1)*3+3)
end do
write(saida2,*)
RETURN
end

!=====================================================================
!
! Subrotina: Cholesky
!
! Descrição: Esta subrotina resolve o sistema linear A*U=F pelo método
!            de fatoração de Cholesky (A = LT*L), onde a matriz A é
simétrica
!            armazenada na forma de perfil (tipo skyline) e "p" corres-
!            ponde ao vetor apontador da mesma.
!
! Autor:     Cristiano Santos Aguiar
!            Thiago Lacerda
!
! Comentários:
!
!=====================================================================
subroutine CHOLESKY(a,p,F,NNO)
integer p(3*NNO),POS
real*8 a(p(3*NNO)),aux,F(3*NNO),U

!Monta a matriz Uij
do i=1,3*NNO
        do j=i,3*NNO
                if (i.NE.j) then
                        if (i.gt.(j-(p(j)-p(j-1)))) then
                                aux=0
                                do k=1,(i-1)
                                        aux=aux+U(k,i,a,p)*U(k,j,a,p)
                                end do
                                a(POS(i,j,p))=(U(i,j,a,p)-aux)/U(i,i,a,p)
                        end if
```

```
                else
                        aux=0
                        do k=1,(i-1)
                                aux=aux+U(k,i,a,p)**2
                        end do
                        a(POS(i,j,p))=sqrt(U(i,j,a,p)-aux)
                end if
        end do
end do

!Monta o Vetor Y --> Ut*Y=f

F(1)=F(1)/a(1)

do i=2,3*NNO
        aux=0
        do j=1,(i-1)
                aux=aux+U(j,i,a,p)*F(j)
        end do
        F(i)=(F(i)-aux)/U(i,i,a,p)
end do

!Monta o vetor d --> U*d=Y

F(3*NNO)=F(3*NNO)/U(3*NNO,3*NNO,a,p)

do i=(3*NNO-1),1,-1
        aux=0
        do j=(3*NNO),(i+1),-1
                aux=aux+U(i,j,a,p)*F(j)
        end do
        F(i)=(F(i)-aux)/U(i,i,a,p)
end do

return
end


!=====================================================================
!
! Subrotina: Deslocamentos
!
! Descrição: Esta subrotina imprime os resultados de deslocamentos no
!            arquivo de saída ".sai"
!
! Autor:     Ricardo Caldeira de Oliveira
!
! Comentários:
!
!=====================================================================
subroutine DESLOCAMENTOS (F,saida,NNO)
integer::i,saida
real*8 ::F(3*NNO)

write(saida,*)'*****************************************************************
******************'
write(saida,*)'                                  R E S U L T A D O S'
write(saida,*)'*****************************************************************
******************'
write(saida,*)
write(saida,*)'      d e s l o c a m e n t o s    n o d a i s'
write(saida,*)'      no  sistema   deslocamento        rotacao
rotacao'
write(saida,*)'                              z                xx
yy'
do i=1,NNO
        write(saida,'(6x,i3,3x,a6,4x,e12.4,4x,e12.4,3x,e12.4)')
        i,'global',F((i-1)*3+1),F((i-1)*3+2),F((i-1)*3+3)
end do
write(saida,*)

RETURN
end
```

```
!=====================================================================
!
! Subrotina: Reacoes
!
! Descrição: Esta subrotina calcula as reações de apoio nodais
!
! Autor:     Ricardo Caldeira de Oliveira
!
! Comentários:
!
!=====================================================================
subroutine REACOES(F,CCONT,saida,NG,NNO,NCONT)
integer:: saida,CCONT(NCONT,4)
real*8 :: F(3*NNO),NG

write(saida,*)'      r e a ç õ e s    n o s    a p o i o s'
write(saida,*)'      nó  sistema          força           momento
momento'
write(saida,*)'                              z               xx
yy'
do i=1,NCONT
        write(saida,'(6x,i3,3x,a6,\)')CCONT(i,1),'global'
        do j=1,3
                if (CCONT(i,j+1).eq.1) then
                        write(saida,'(6x,f10.2,\)')-NG*F((CCONT(i,1)-1)
                        *3+j)
                else
                        write(saida,'(6x,f10.2,\)')0
                end if
        end do
        write(saida,*)
end do
write(saida,*)

return
end


!=====================================================================
!
! Subrotina: Pnt_creator
!
! Descrição: Esta subrotina cria vetor apontador no esquema de
!            armazenamento presente no pacote MKL (Intel)
!
! Autor:     Ricardo Caldeira de Oliveira
!
! Comentários:
!
!=====================================================================
SUBROUTINE pnt_creator(p,pnt,nno)
INTEGER i,nno
INTEGER:: p(nno*3),pnt(nno*3+1),alt(nno*3)


pnt   =0
pnt(1)=1
alt(1)=1

        Do i=2, nno*3
          alt(i)=p(i)-p(i-1)
        End Do

        Do i=1, nno*3
          pnt(i+1)= alt(i) + pnt(i)
        End Do

Return
END SUBROUTINE
```

```
!=====================================================================
!
! Subrotina: solve_pardiso
!
! Descrição: Esta subrotina é responsável pela chamadas das rotinas de
!            resolução do sistema linear do pacote Pardiso do conjunto
!            de rotinas do MKL.
!
! Autor:     Cristiano Santos Aguiar
!
! Comentários:
!
!=====================================================================
subroutine
solve_pardiso(neq,nmatglob,iacrs,jacrs,mglob,fdes,resp,l_fat,l_ret,l_fim,np
roc)

implicit none
save
! neq => número de equações
! nmatglob => tamanho do vetor ja e acsr(mglob)
integer neq,nmatglob,nproc
logical impr_status
integer,intent(in)   :: iacrs(neq+1),jacrs(nmatglob)
real*8,intent(inout) :: mglob(nmatglob),fdes(neq),resp(neq)
logical,intent(in)   :: l_fat,l_ret,l_fim

integer    :: mtype,iparm(64),phase,msglvl,maxfct,mnum,nrhs,error,idum
real*8     :: ddum
integer*8  :: pt(64)

! Setup Pardiso control parameters and initialize the solvers
! internal adress pointers. This is only necessary for the FIRST
! call of the PARDISO solver.
nrhs = 1
maxfct = 1
mnum = 1
iparm(6) = 1
impr_status = .true.

if (l_fat) then ! Inicializaçao e fatorização

!   write(*,*) 'entrou l_fat'
!   pause
   mtype     = 2  ! matrix symmetric, definite
   call pardisoinit(pt, mtype, iparm)

!--Numbers of Processors ( value of OMP_NUM_THREADS )
   iparm(3) = nproc
!   iparm(60) = 2  !
!--Reordering and Symbolic Factorization, This step also allocates
!  all memory that is necessary for the factorization

   phase     = 11     ! only reordering and symbolic factorization
   msglvl    = 0       ! without statistical information
   call pardiso
   (pt,maxfct,mnum,mtype,phase,neq,mglob,iacrs,jacrs,idum,nrhs,iparm,msglvl
   ,ddum,ddum,error)

   if (impr_status) write(*,*) 'PARDISO - Reordering completed ... '

   if (error .ne. 0) then
      write(*,*) 'The following ERROR was detected: ', error
      pause
      stop
   end if

!   write(*,*) 'Number of nonzeros in factors   = ',iparm(18)
!   write(*,*) 'Number of factorization MFLOPS  = ',iparm(19)

!--Factorization.
   phase     = 22  ! only factorization
   call pardiso
```

```
   (pt,maxfct,mnum,mtype,phase,neq,mglob,iacrs,jacrs,idum,nrhs,iparm,msglvl
   ,ddum,ddum,error)

   if (impr_status) write(*,*) 'PARDISO - Factorization completed ... '
   if (error .ne. 0) then
       write(*,*) 'The following ERROR was detected: ', error
       pause
       stop
   endif
endif

if (l_ret) then !retrosubstituiçao
!   write(*,*) 'entrou l_ret'
!   pause

   resp(:) = 0.0d0
!--Back substitution and iterative refinement
   iparm(8)  = 1   ! max numbers of iterative refinement steps
   phase     = 33  ! only factorization

   call pardiso
   (pt,maxfct,mnum,mtype,phase,neq,mglob,iacrs,jacrs,idum,nrhs,iparm,msglvl
   ,fdes,resp,error)

!   write(*,*)sum(resp)

   if (impr_status) write(*,*) 'PARDISO - Solve completed ...  '
endif

if (l_fim) then
!   write(*,*) 'entrou l_fim'
!   pause

!--Termination and release of memory
   phase     = -1           ! release internal memory
   call pardiso
   (pt,maxfct,mnum,mtype,phase,neq,ddum,idum,idum,idum,nrhs,iparm,msglvl,dd
   um,ddum,error)
endif

return
end


!=====================================================================
!
! Subrotina: Multiplica
!
! Descrição: Esta subrotina multiplica matriz (C=A*B)
!
! Autor:     Ricardo Caldeira de Oliveira
!
! Comentários:
!
!=====================================================================
subroutine multiplica(a,b,c,m,n,q)
integer m,n,q,i,j,k
real*8 a(m,n),b(n,q),C(m,q)

  do i=1,m
    do j=1,q
       C(i,j)=0
       do k=1,n
         C(i,j)=C(i,j)+A(i,k)*B(k,j)
       end do
    end do
  end do
return
end


!=====================================================================
!
```

```
! Subrotina: Transposta
!
! Descrição: Esta subrotina avalia a transposta de A (B=AT)
!
! Autor:     Ricardo Caldeira de Oliveira
!
! Comentários:
!
!=====================================================================
subroutine transposta(a,b,m,n)
integer m,n
real*8 a(m,n),b(n,m)

  do i=1,m
    do j=1,n
      b(j,i)=a(i,j)
    end do
 end do
return
end


!=====================================================================
!
! Função:    POS
!
! Descrição: Esta função indica a posição no auxílio da resolução do
!            sistema linear (cholesky)
!
! Autor:     Cristiano Santos Aguiar
!            Thiago Lacerda
!
! Comentários:
!
!=====================================================================
recursive function POS(i,j,p) result(R)
common NNO
integer:: i,j,R,p(3*NNO)

if ((i.eq.1).and.(j.eq.1)) then
        R=1
else if (p(j)-p(j-1).eq.j) then
                R=p(j-1)+i
        else
                R=p(j-1)+i-(j-(p(j)-p(j-1)))
end if

RETURN
end function


!=====================================================================
!
! Função:    U
!
! Descrição: Esta função calcula posições para subrotina cholesky
!
! Autor:     Cristiano Santos Aguiar
!            Thiago Lacerda
!
! Comentários:
!
!=====================================================================
recursive function U(i,j,a,p) result (R)
common NNO
integer:: i,j,p(3*NNO)
real*8 ::R,a(p(3*NNO))


if (i.eq.j) then
        if(i.eq.1) then
                R=a(1)
        else
```

```
                R=a(p(i))
        end if
else if (p(j)-p(j-1).eq.j) then
                R=a(p(j-1)+i)
        else if (i.le.(j-(p(j)-p(j-1)))) then
                        R=0
                 else
                        R=a(p(j-1)+i-(j-(p(j)-p(j-1))))
end if
RETURN
end function
```