Vibração de placas metálicas finas: um estudo inspirado no problema de Chladni

Daniela Passos Maia Moura

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: **Matemática Aplicada**

Orientador: **Profº. Dr. Eduardo Colli**

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro do CNPq

São Paulo, Janeiro de 2017

Vibração de placas metálicas finas:
um estudo inspirado no problema de Chladni

Esta é a versão original da dissertação elaborada pela
candidata Daniela Passos Maia Moura, tal como
submetida à Comissão Julgadora.

Agradecimentos

Agradeço primeiramente a Deus, por ter me dado a oportunidade de estudar e aprender matemática e por ter me capacitado para isso.

Agradeço a minha família que sempre me apoiou nos estudos em todos os momentos, aos meus pais Haroldo e Conceição que sempre valorizaram a educação, incentivaram e apoiaram minha formação, e aos meus irmãos Lígia, Kaio Alexandre e Cínthia, pelo apoio. Agradeço em especial ao meu esposo Daniel de Andrade Moura que sempre esteve ao meu lado, acreditando e sendo paciente ao longo dessa jornada. Seu apoio e incentivo foram fundamentais.

Agradeço ao meu orientador, professor Dr. Eduardo Colli, pelos ensinamentos, apoio, incentivo, paciência e bom ânimo. Agradeço a professora Dra. Sônia Regina Leite Garcia, que me orientou no início do mestrado, sua sabedoria e conselhos foram muito valiosos nesta etapa.

Agradeço aos amigos que tive a oportunidade de conhecer e com os quais convivi ao longo desse período. O companheirismo foi de suma importância e sem dúvida, ajudou a tornar os desafios do mestrado mais leves: Janaina Baldan Santos, Verônica Leão Neves, Inocêncio Jaime Ferreira Zimba, Marcela Guerrini Alves, Carla Reis Evangelista, Rosângela Teixeira Guedes. Agradeço também ao suporte dado pelo Laboratório de Matemática Aplicada durante a etapa de programação. Agradeço aos colegas de laboratório Nelson Leonardo Vidaurre Navarrete, Nils Urmersbach e Evandro (ex-administrador do LabMap), a amizade e a ajuda desprendida foram muito valiosas. Agradeço também aos professores Drs. Saulo Rabello Maciel de Barros e Pedro da Silva Peixoto, pela disposição, ajuda e compartilhamento de seu conhecimento de análise numérica.

Agradeço ao Instituto de Matemática e Estatística (IME-USP) e ao departamento de Matemática Aplicada pelo acolhimento e oportunidade e ao CNPQ pelo auxílio financeiro.

Por fim, deixo um agradecimento especial à Matemateca do IME-USP, e toda a sua equipe, pelo acolhimento desde o período da minha graduação, pela motivação pelo estudo da matemática e pela inspiração ao tema de estudo neste trabalho.

Resumo

MOURA, D. P. M. **Vibração de placas metálicas finas: um estudo inspirado no problema de Chladni**. 2017. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2017.

Os estudos descritos neste trabalho foram inspirados no experimento de vibração de placas metálicas finas realizado/investigado por Ernst Chladni no século XVIII, no qual era possível, através da vibração das placas, a observação das linhas nodais decorrentes dessa vibração. Nesse experimento, a placa é presa apenas pelo seu centro e sua borda permanece livre. Estudamos o problema de vibração de placas finas desde a sua modelagem física, que leva à equação diferencial parcial que governa o problema, até a implementação numérica. Na implementação numérica trabalhamos com uma condição de contorno que equivale a considerar a placa presa nas bordas. Apesar de essa condição de contorno ser diferente daquela do problema de Chladni, este trabalho apresenta discussões e apontamentos para trabalhos futuros com a condição de contorno de bordo livre. Caracterizamos a solução da equação diferencial, que consiste de uma parte temporal e uma parte espacial. A parte espacial é solução de um problema de autovalor, que nos propusemos a resolver numericamente, utilizando o Método de Elementos Finitos (MEF). Apresentamos (três) diferentes formulações dentro do mesmo método (MEF): uma aproximação polinomial quadrática não conforme e duas aproximações polinomiais de quinto grau, ambas conformes; e fizemos comparações entre elas. As conformes serão úteis para trabalhos futuros com a condição de contorno de bordo livre. A não conforme foi usada para a obtenção dos principais resultados: as imagens das linhas nodais que caracterizam os modos de vibração da placa e os autovalores correspondentes. Estudamos como os autovalores associados aos modos de vibração de uma placa retangular evoluem em função da proporção entre seus lados. Também apresentamos imagens de linhas nodais para placas de diferentes formatos poligonais.

Palavras-chave: Placas de Chladni, figuras de Chladni, método de elementos finitos, vibração

de placas finas, linhas nodais, bilaplaciano, método conforme, método não conforme.

Abstract

MOURA, D. P. M. **Vibration of thin metal plates: a study inspired by Chladni's problem**. 2017. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2017.

The studies described in this work were inspired by the experiment of vibration of thin metal plates, carried out/investigated by Ernst Chladni in the eighteenth century, in which the observation of the nodal lines resulting from this vibration was possible through the vibration of the plates. In this experiment, the plate is clamped only by its center and its edge remains free. We studied the vibration problem of thin plates from their physical modeling, which leads to the partial differential equation that governs the problem, until the numerical implementation. In the numerical implementation, we worked with a boundary condition that is equivalent to considering the plate clamped by their edges. Although this boundary condition is different from that of the Chladni's problem, this work presents discussions and notes that can be used for future works with the condition of free boundary. We characterized the solution of the differential equation, which is composed of a temporal part and a spatial part. The spatial part is the solution of an eigenvalue problem, which we proposed to solve numerically, using the Finite Element Method (FEM). We present (three) different formulations within the same method (FEM): a nonconforming quadratic polynomial approximation and two fifth degree polynomial approximations, both conforming; and we made comparisons between them. The conforming ones will be useful for future works with the condition of free boundary. The nonconforming one was used to obtain the main results: the nodal line images that characterize the plate vibration modes and the corresponding eigenvalues. We study how the eigenvalues associated with the vibration modes of a rectangular plate evolve as a function of the proportion between its sides. Also we present images of nodal lines for plates of different polygonal formats.

Key-words: Chladni plates, Chladni figures, finite element method, thin vibration plates, nodal

lines, bilaplacian, conforming method, nonconforming method.

Sumário

Introdução

Há cerca de 200 anos, Ernst Friedrich Chladni, da Saxônia, publicou trabalhos nos quais ele descrevia seu método bem conhecido de uso de areia aspergida sobre placas para mostrar as linhas nodais. Nesse método, a placa fina era presa apenas pelo centro, e a vibração era induzida passando um arco de violino sobre a aresta (livre) da placa. E. Chladni publicou *Entdeckungen über die Theorie des Klanges* em 1787 e *Die Akustic*, em 1802. As palestras de Chladni em cortes europeias atraíram personalidades famosas. Napoleão Bonaparte ficou tão encantado com suas demonstrações que financiou a tradução de *Die Akustik* para a língua francesa, e também forneceu ao *Institute of France* um prêmio de 3.000 francos a ser concedido para uma teoria matemática satisfatória de vibrações de placas. Este prêmio foi concedido, em 1815, a Marie-Shopie Germain, que forneceu uma modelagem correta para vibrações de placas com uma equação diferencial de quarta ordem, embora sua escolha para as condições de contorno estivesse incorreta [13]. Lagrange e Poisson também deram contribuições na teoria da vibração de placas finas [4].

Em 1850, G.R. Kirchhoff publicou dois artigos sobre a teoria de vibrações de placas finas, onde deu um tratamento mais preciso às condições de contorno [13]. Muitos outros cientistas estiveram envolvidos com o estudo dos padrões de Chladni gerados pela vibração de placas. Segundo T. Rossing [13], um dos estudos mais exaustivos foi feito por Mary Desirée Waller, que escreveu vários artigos a respeito, onde apresentou resultados e imagens de vibrações de placas quadradas (com o bordo livre).

O objetivo deste trabalho foi estudar a equação diferencial parcial que modela o problema de vibração de placas finas, a partir da modelagem física que determina essa equação, e estudar um método para resolução numérica do problema para placas de diferentes formatos. Embora inspirado nos experimentos de E. Chladni, o estudo e a implementação numérica apresentado nesse trabalho foram feitos para uma condição de contorno mais simples: a de *bordo preso*. As equações que descrevem as equações de contorno de *bordo livre* são mais complicadas e ficarão para estudos futuros. De fato, o problema de vibração de uma placa retangular com bordo livre, foi tido como de

dificuldade considerável por Rayleigh [12]. Nossa abordagem, no entanto, em vários aspectos deixa preparado o caminho para o tratamento do problema de bordo livre e centro preso, proposto por Chladni.

Embora a condição de *bordo preso* não represente o estudo feito por E. Chladni, mesmo assim as imagens das linhas nodais obtidas são bastante interessantes e, ao menos para o caso retangular, são parecidas com as da membrana vibrante.

No primeiro Capítulo, apresentamos a formulação da equação diferencial que governa o fenômeno, baseado na teoria de Kirchhoff, juntamente com as condições de contorno e indicamos nossa escolha pela condição de contorno de bordo preso. Desenvolvemos a equação diferencial parcial (EDP) até obter um problema de autovalor e caracterizamos a solução (autovalores e autofunções) procurada. No Capítulo 2, apresentamos a implementação numérica baseada no Método de Elementos Finitos (com duas formulações) para a solução do problema e geração de imagens das linhas nodais da vibração. A primeira formulação, usando polinômios quadráticos, apresentada é mais simples. A segunda, que usa polinômios de quinto grau, é mais complexa, mas permite o tratamento do problema com condição de contorno *bordo livre*. Os melhores resultados, no entanto, ocorreram com a primeira. A segunda formulação trouxe algumas dificuldades e as alternativas que implementamos para contorná-las estão descritas no Capítulo 3. Finalmente, no Capítulo 4, apresentamos os resultados obtidos numericamente e algumas análises desses.

Vale a pena observar que, durante o desenvolvimento deste trabalho, não foi encontrada nenhuma referência que tratasse o problema de forma completa, considerando todas as questões teóricas – como o tratamento correto a ser dados às condições de contorno – e de implementação numérica usando o Método de Elementos Finitos. Nem tampouco uma referência atualizada com resultados teóricos ou numéricos para diferentes formatos de placas e condições de contorno. Para ter parâmetros comparativos, utilizamos um livro de 1969 publicado pela NASA [8], que compila diversos resultados esparsos da literatura, obtidos seja por meio de experimentos seja por solução aproximada usando argumentos de expansão em séries. Este trabalho poderia ser, portanto, a semente para a criação de uma obra de referência semelhante que usasse a tecnologia atual.

A título de ilustração, as imagens a seguir mostram as linhas nodais da vibração de uma placa pentagonal. A primeira, Figura 1, foi obtida experimentalmente e a segunda, Figura 2, apesar de não representar exatamente o experimento da primeira – devido às diferentes condições de contorno – é um dos resultados deste trabalho.

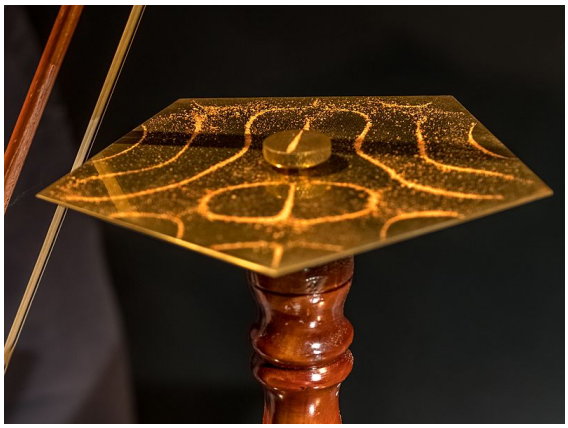


Figura 1: *Linhas nodais para uma placa pentagonal geradas experimentalmente. Fonte: Matemateca (IME/USP)/Rodrigo Tetsuo Argenton.*

Figura 2: *Linhas nodais para uma placa pentagonal, com o bordo preso, geradas numericamente. Formulação quadrática.*

Capítulo 1

Formulação

O objetivo deste capítulo é apresentar, a partir de uma modelagem física (ou de um modelo físico), a formulação do problema de vibração de uma placa fina de metal e obter a equação que governa tal fenômeno.

Além disso, preparar o terreno teórico para as técnicas de solução numérica, quer serão apresentadas nos Capítulos 2 e 3.

1.1 Modelagem dinâmica de uma placa fina de metal

Definimos uma placa como sendo um corpo tridimensional (sólido e elástico) de espessura constante e de material isotrópico (i.e. cujas propriedades não dependem da orientação do sistema) e homogêneo (que possui as mesmas características em qualquer ponto). A equação diferencial parcial associada à vibração de uma placa fina, de espessura constante h e de material sólido e isotrópico é

$$D\Delta\Delta w + \rho w_{tt} = 0 \ , \tag{1.1}$$

em que $w = w(x, y, t)$ é o deslocamento perpendicular ao plano da placa no ponto (x, y) e no instante de tempo t, ρ é a densidade de área e a constante $D = \dfrac{Eh^3}{12(1-\nu^2)}$ é chamada de rigidez da placa. Ela depende do coeficiente de Poisson ν (constante adimensional), do módulo de elasticidade de Yong E e da espessura da placa h.

Essa equação pode ser obtida através da teoria do Cálculo Variacional, que consiste na formulação e minimização do funcional Ação J associado à vibração de uma placa fina. Determinar um ponto crítico do funcional J equivale a determinar uma função que satisfaz a condição $\delta J = 0$, isto é, a primeira variação de J deve ser zero.

Para escrever o funcional Ação é preciso primeiro obter a energia potencial através da teoria da Elasticidade Linear, de onde também surgem as constantes físicas presentes na equação diferencial (1.1).

1.1.1 Problemas elasto-mecânicos estáticos e dinâmicos

Para o tratamento de problemas elasto-mecânicos estáticos e/ou dinâmicos existem princípios variacionais que surgem naturalmente da mecânica e que são bastante úteis. Como estamos trabalhando com o problema de vibração de uma placa de metal, utilizamos o princípio de minimização

do funcional Ação (que envolve a energia potencial e cinética do sistema). A definição de uma placa e suas características descritas na seção acima são importantes para a formulação da energia potencial de deformação da placa, onde utilizamos conceitos e relações provenientes da teoria de Elasticidade Linear. Tal teoria trata de entender como um corpo elástico se comporta (se deforma) quando está sob a ação de forças externas que geram tensões e deformações neste corpo [6]. Ela ainda estabelece relações entre as tensões e deformações sofridas. É dela que surgem as constantes elásticas, como o módulo de elasticidade de Yong e o coeficiente de Poisson, que aparecem na equação diferencial. Com base nas relações entre tensão e deformação, podemos escrever a energia potencial (elástica) em função dos vetores de deslocamento da placa. O efeito de uma vibração na placa pode ser modelado segundo a Lei Generalizada de Hooke, que relaciona, de maneira linear, a deformação e a tensão de uma estrutura plana ou espacial, de material elástico, isotrópico e cujas deformações sofridas são pequenas. Apresentaremos aqui as relações básicas, mas para um estudo mais profundo, indicamos o Capítulo 3 de [6] e [4].

1.1.2 A energia potencial de um corpo elástico

Considere um corpo elástico tridimensional que não sofre a ação de forças externas. A energia potencial desse corpo é dada por:

$$U = \frac{1}{2} \iiint_V \boldsymbol{\sigma}^T \boldsymbol{\varepsilon} \, dV \tag{1.2}$$

em que $\boldsymbol{\sigma}$ é o vetor de tensão, $\boldsymbol{\varepsilon}$ é o vetor deformação e V é o volume do corpo. Tal integral de volume corresponde ao trabalho ("força×deslocamento") realizado por tensões internas. Vejamos o que isso significa.

Seja $\boldsymbol{f}$ o vetor deslocamento, dependente da posição num ponto $P = (x, y, z)$ do corpo. Ele é composto por três componentes w_1,w_2 e w_3, nas direções espaciais x, y e z.

$$\boldsymbol{f} = (w_1, w_2, w_3)^T,$$

em que w_1,w_2, e w_3 são funções de cada uma das variáveis espaciais x,y e z.

O vetor deformação $\boldsymbol{\varepsilon}$ é adimensional e dado por

$$\boldsymbol{\varepsilon} = (\varepsilon^x, \varepsilon^y, \varepsilon^z, \gamma^{xy}, \gamma^{yz}, \gamma^{zx})^T \text{[1]},$$

em que $\varepsilon^x, \varepsilon^y$ e ε^z são deformações nas direções x, y e z: elas caracterizam as mudanças nos comprimentos preservando ângulos; $\gamma^{xy}, \gamma^{yz}, \gamma^{zx}$ são chamadas de deformações de cisalhamento: elas correspondem à mudanças nos ângulos originalmente retos dos planos xy, yz e xz [6].

Na teoria da elasticidade linear, essas deformações são definidas, em termos das componentes do vetor deslocamento por

$$\varepsilon^x = \frac{\partial w_1}{\partial x}, \qquad \varepsilon^y = \frac{\partial w_2}{\partial y}, \qquad \varepsilon^z = \frac{\partial w_3}{\partial z}, \tag{1.3}$$

[1] Em geral a notação, encontrada na literatura, para o vetor deformação é feita com índices na parte inferior. Optamos por utilizar índices na parte superior, pois índices inferiores serão usados para denotar derivações.

e

$$\gamma^{xy} = \frac{\partial w_2}{\partial x} + \frac{\partial w_1}{\partial y}, \qquad \gamma^{yz} = \frac{\partial w_2}{\partial z} + \frac{\partial w_3}{\partial y}, \qquad \gamma^{xz} = \frac{\partial w_3}{\partial x} + \frac{\partial w_1}{\partial z} \ . \tag{1.4}$$

Elas também podem ser escritas matricialmente como $\boldsymbol{\varepsilon} = \boldsymbol{B}\boldsymbol{f}$, em que $\boldsymbol{B}$ é o operador diferencial:

$$B = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} \ . \tag{1.5}$$

A tensão $\boldsymbol{\sigma}$ num ponto genérico $P = (x, y, z)$ dentro de um corpo deformado corresponde a força por unidade de área agindo sobre os planos normais aos eixos coordenados x, y e z. O vetor tensão pode ser decomposto em componentes normais (σ) à secção transversal do corpo e em componentes tangenciais (τ) à tal secção transversal. As componentes tangenciais são chamadas de tensões de cisalhamento. Considerando condições de equilíbrio interno, a tensão $\boldsymbol{\sigma}$ pode ser descrita por seis componentes:

$$\sigma^x, \quad \sigma^y, \quad \sigma^z, \quad \tau^{xy}, \quad \tau^{yz}, \quad \tau^{xz} \ ,$$

em que σ^x, σ^y, σ^z são as tensões normais nas direções coordenadas e τ^{xy}, τ^{yz}, τ^{xz}, são as três tensão de cisalhamento. Temos então o vetor tensão :

$$\boldsymbol{\sigma} = (\sigma^x, \sigma^y, \sigma^z, \tau^{xy}, \tau^{yz}, \tau^{zx})^T$$

Existem leis físicas que relacionam a tensão e a deformação e descrevem o comportamento elástico-mecânico do material. No contexto da teoria da elasticidade linear, essa relação é dada pela forma geral da Lei de Hooke, que relaciona a tensão e a deformação de forma linear. Como estamos considerando um corpo de material isotrópico e homogêneo, tal lei é aplicável e obtêm-se (conforme Capítulo 3 de [6]) as relações:

$$\begin{aligned} \varepsilon^x &= \frac{1}{E}(\sigma^x - \nu\sigma^y - \nu\sigma^z), \quad \gamma^{xy} = \frac{1}{G}\tau^{xy} \ , \\ \varepsilon^y &= \frac{1}{E}(-\nu\sigma^x + \sigma^y - \nu\sigma^z), \quad \gamma^{yz} = \frac{1}{G}\tau^{yz} \ , \\ \varepsilon^z &= \frac{1}{E}(\nu\sigma^x - \nu\sigma^y - \sigma^z), \quad \gamma^{xz} = \frac{1}{G}\tau^{xz} \ , \end{aligned} \tag{1.6}$$

em que E é o módulo de elasticidade de Yong, ν é o coeficiente de Poisson (adimensional) e $G = \frac{2(1+\nu)}{E}$ é o módulo de cisalhamento. Estas são constantes que descrevem o comportamento elástico do material. A constante E é usualmente expressa em MPa ou GPa ($1GPa = 10^3 MPa = 10^6 N/m^2$). O coeficiente de Poisson ν é adimensional e representa a razão entre deformações horizontais e verticais (Capítulo 3 de [6]). Se o material considerado não sofre tais deformações, a constante ν deve ser nula (Capítulo 10 de [12]). Segundo Rayleigh [12], $0 \leq \nu < 1$, e, para placas de metal, $\nu \approx 0.3$ [6, 12].

Resolvendo o sistema (1.6) para o vetor tensão $\boldsymbol{\sigma}$, obtemos:

$$\boldsymbol{\sigma} = D\boldsymbol{\varepsilon} \ ,$$

em que D é matriz simétrica e definida positiva, dada por:

$$D = \frac{E}{(1-2\nu)(1+\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} \end{bmatrix} .$$

Assim, a expressão para a energia potencial (devido apenas à forças internas) fica assim:

$$U = \frac{1}{2} \iiint_V \boldsymbol{\sigma}^T \boldsymbol{\varepsilon} \ dV = \frac{1}{2} \iiint_V \boldsymbol{\varepsilon}^T D\boldsymbol{\varepsilon} \ dV \ . \tag{1.7}$$

Usando a relação entre os vetores de deformação e deslocamento dada por (1.5), também obtemos uma expressão para a energia potencial em termos do vetor deslocamento $\boldsymbol{f}$:

$$U = \frac{1}{2} \iiint_V \boldsymbol{\sigma}^T \boldsymbol{\varepsilon} \ dV = \frac{1}{2} \iiint_V \boldsymbol{\varepsilon}^T D\boldsymbol{\varepsilon} \ dV = \frac{1}{2} \iiint_V \boldsymbol{f}^T \boldsymbol{B}^T \boldsymbol{D}\boldsymbol{B}\boldsymbol{f} \ dV \ . \tag{1.8}$$

1.1.3 Lagrangiano de um corpo elástico

Olhemos agora para as energias potencial e cinética do sistema. Consideramos o funcional Lagrangiano:

$$L = T - U$$

em que T é a energia cinética do sistema em consideração e U é a energia potencial de deformação. Para um corpo com densidade $\tilde{\rho}(x, y, z)$ a energia cinética pode ser escrita em função do vetor de deslocamento $\boldsymbol{f}$:

$$T = \frac{1}{2} \iiint_V \tilde{\rho} \dot{\boldsymbol{f}}^T \dot{\boldsymbol{f}} \ dV \tag{1.9}$$

em que $\dot{\boldsymbol{f}}$ representa a derivada do vetor deslocamento $\boldsymbol{f}$ com respeito ao tempo, i.e. representa o vetor velocidade.

Assim, da equação (1.2), temos que o funcional Lagrangiano é dado por:

$$L = \frac{1}{2} \iiint_V \tilde{\rho} \dot{\boldsymbol{f}}^T \dot{\boldsymbol{f}} \ dV - \frac{1}{2} \iiint_V \boldsymbol{\sigma}^T \boldsymbol{\varepsilon} \ dV \ . \tag{1.10}$$

Segundo o princípio de Hamilton, entre dois instantes t_0 e t_1, o movimento procede de tal modo que a integral

$$J = \int_{t_0}^{t_1} L \ dt \tag{1.11}$$

é estacionária em comparação à funções vizinhas cujos valores em t_0 e em t_1 coincidem com os

valores de L nesses mesmos instantes (Capítulo 4, P243 de [2]).

1.1.4 Caso específico: vibração de uma placa fina

Nesta etapa adequamos as equações obtidas acima, de maneira geral, para as condições do nosso caso particular: a vibração de uma placa fina de metal. Estamos considerando que uma placa é um corpo sólido, tridimensional, feito de material homogêneo e isotrópico e que a característica "fina" significa que sua espessura é pequena quando comparada com as duas outras dimensões. Essas características são importantes para que se possa considerar a Teoria de Kirchhoff para placas finas de metal. Kirchhoff foi o responsável pela formulação correta da formulação matemática da vibração de uma placa fina de metal, primeiramente apresentada por Sophie Germain, em 1815 [13].

Consideramos uma placa fina, de espessura constante h, cuja superfície média (ou central) coincide com o plano xy. Consideramos ainda as seguintes hipóteses (de Kirchhoff):

(i) o deslocamento vertical w_3 da superfície média (central) e suas derivadas parciais são pequenos; assim, a superfície média é considerada como uma camada neutra que não sofre distorções (Seção 1.2 de [15]); isso implica que, no plano $z = 0, \varepsilon^x = \varepsilon^y = \gamma^{xy} = 0$;

(ii) direções normais ao plano médio da placa não deformada permanecem normais sob deformações;

(iii) a espessura da placa não muda durante a deformação;

(iv) a tensão σ^z na direção normal ao plano xy é irrelevante quando comparada com as tensões σ^x e σ^y.

Sob essas hipóteses, a teoria de Kirchhoff estabelece uma relação entre as componentes do vetor deslocamento. Essa relação nos diz que os deslocamentos w_1 e w_2 variam com a altura z do ponto e dependem do deslocamento vertical w de um ponto $(x, y, 0)$ da superfície média [15].

Observe a Figura (1.1), onde foi considerada uma secção no plano $y = 0$, de modo que olharemos apenas para as deformações que ocorrem no plano xz da placa. Consideramos a placa (bem como a sua superfície média) em sua posição original, antes da deformação, e depois da deformação. Seja $w(x, y)$ a função que descreve a posição vertical da superfície média após a deformação.

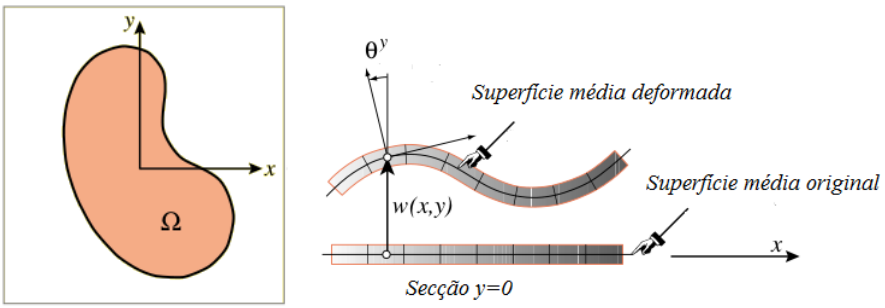


Figura 1.1: *À esquerda: Domínio da placa visto de cima. À direita: deslocamento e variação angular no plano xz da placa. Figura grandemente exagerada, para melhor visibilidade. Na prática, w é muito menor que a espessura h, para que o modelo de Kirchhoff possa ser considerado. Figura baseada na referência [4].*

Considere um ponto (x, y, z) fora da superfície média da placa e, portanto, a uma altura z desta. Olhando só para o plano da figura, no corte $y = 0$, temos que o ponto de altura z sofreu

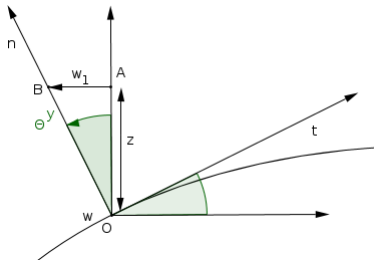


Figura 1.2: *Região do ângulo θ^y aumentada.*

um deslocamento horizontal $w_1(x, y, z)$ no sentido oposto à orientação do eixo horizontal x. A expressão desse deslocamento pode ser obtida com base no triângulo OAB da Figura 1.2, e é tal que $\tan\theta^y \cong \dfrac{w_1(x,y,z)}{z}$. Por outro lado, essa é a inclinação da reta tangente $w(x, y)$ na direção de x. Ou seja,

$$\frac{-w_1(x,y,z)}{z} \cong \tan\theta^y = \frac{\partial w}{\partial x} \ .$$

Logo, $w_1(x, y, z) = -z\dfrac{\partial w}{\partial x}$, em primeira ordem. Olhando para a secção no plano $x = 0$, obtemos de forma análoga, a relação entre o deslocamento vertical da superfície média $w(x, y)$ e o deslocamento w_2 na direção do eixo y.

Assim, temos que

$$w_1(x,y,z) = -z\frac{\partial w}{\partial x}(x,y) \ , \quad w_2(x,y,z) = -z\frac{\partial w}{\partial y}(x,y) \ \ e \ \ w_3(x,y,z) = w(x,y) \ . \tag{1.12}$$

Com base nas equações (1.3), (1.4) e (1.12), as componentes do vetor de deformação são:

$$\varepsilon^x = -z\frac{\partial^2 w}{\partial x^2}, \qquad \varepsilon^y = -z\frac{\partial^2 w}{\partial y^2}, \qquad \varepsilon^z = 0,$$

$$\gamma^{xy} = -2z\frac{\partial^2 w}{\partial x \partial y}, \qquad \gamma^{yz} = 0, \qquad \gamma^{zx} = 0. \tag{1.13}$$

Quanto às forças de tensão, baseando-nos ainda nas hipóteses de Kirchhoff, consideramos que as tensões normais à superfície média são desconsideradas, o que nos leva a assumir que $\sigma^z = \tau^{xz} = \tau^{yz} = 0$.

Assim, aplicando a Lei geral de Hooke (1.6), obtemos as expressões

$$\sigma^x = \frac{E}{1-\nu^2}(\varepsilon^x + \nu\varepsilon^y) \ ,$$

$$\sigma^y = \frac{E}{1-\nu^2}(\nu\varepsilon^x + \varepsilon^y) \ ,$$

$$\tau^{xy} = \frac{E}{2(1+\nu)}\gamma^{xy} \ ,$$

que se resumem à equação matricial $\boldsymbol{\sigma} = D\boldsymbol{\varepsilon}$, com

$$D = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{(1-\nu)}{2} \end{bmatrix}.$$

Usando as equações em (1.13), obtemos as expressões das componentes de tensão em função da componente vertical $w(x,y)$ do deslocamento:

$$\begin{aligned} \sigma^x &= \frac{-Ez}{1-\nu^2}\left(\frac{\partial^2 w}{\partial x^2} + \nu\frac{\partial^2 w}{\partial y^2}\right) , \\ \sigma^y &= \frac{-Ez}{1-\nu^2}\left(\nu\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}\right) , \\ \tau^{xy} &= \frac{-Ez}{1+\nu}\frac{\partial^2 w}{\partial x \partial y} . \end{aligned} \tag{1.14}$$

No nosso caso, estamos considerando que não há forças externas envolvidas na vibração da placa, e a força gravitacional também está sendo desconsiderada. Desse modo, para obter a energia potencial, precisamos apenas calcular

$$\boldsymbol{\sigma}^T\boldsymbol{\varepsilon} = (D\boldsymbol{\varepsilon})^T\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^T D\boldsymbol{\varepsilon}.$$

Como a matriz D é simétrica e definida positiva, podemos afirmar que a energia potencial é a integral de uma forma quadrática, definida positiva:

$$\iiint_V \boldsymbol{\sigma}^T\boldsymbol{\varepsilon}\ dx\ dy\ dz = \iiint_V \boldsymbol{\varepsilon}^T D\boldsymbol{\varepsilon}\ dxdydz\ .$$

A partir daí, integramos na direção do eixo z, sobre a espessura da placa, (lembrando que o deslocamento w representa o deslocamento de pontos pertencentes à superfície média da placa e, portanto, independe da coordenada z).

A partir deste ponto, utilizaremos a notação de índice inferior para derivadas, isto é,

$$\frac{\partial^2 w}{\partial x^2} = w_{xx}, \quad \frac{\partial^2 w}{\partial xy} = w_{xy} \ \ e \ \ \frac{\partial^2 w}{\partial y^2} = w_{yy}\ .$$

Portanto,

$$\begin{aligned} U_{placa} &= \frac{1}{2}\iiint_V \boldsymbol{\varepsilon}^T D\boldsymbol{\varepsilon}\ dxdydz = \\ &= \frac{1}{2}\iiint_V \frac{Ez^2}{1-\nu^2}\left[{w_{xx}}^2 + \nu w_{xx}w_{yy} + {w_{yy}}^2 + \nu w_{xx}w_{yy}\right] + \frac{Ez^2}{2(1+\nu)}4z^2{w_{xy}}^2\ dxdydz = \\ &= \frac{1}{2}\int_{-h/2}^{h/2} z^2\ dz \iint_\Omega \frac{E}{1-\nu^2}\left[{w_{xx}}^2 + 2\nu w_{xx}w_{yy} + w_{yy}^2 + 2(1-\nu){w_{xy}}^2\right]\ dxdy = \\ &= \frac{1}{2}\frac{h^3}{12}\frac{E}{(1-\nu^2)}\iint_\Omega \left[{w_{xx}}^2 + 2\nu w_{xx}w_{yy} + {w_{yy}}^2 + 2(1-\nu){w_{xy}}^2\right]\ dxdy\ . \end{aligned} \tag{1.15}$$

A expressão dentro da integral dupla pode ser ajustada de modo que a expressão da energia potencial fique na forma:

$$U_{placa} = \frac{1}{2}\frac{Eh^3}{12(1-\nu^2)} \iint_\Omega (\Delta w)^2 - 2(1-\nu)\left(w_{xx}w_{xy} - w_{xy}^2\right) dxdy \ . \tag{1.16}$$

Considerando que a placa em questão pode ser considerada como uma superfície parametrizada como um gráfico de função, i.e. que a componente vertical do deslocamento pode ser escrita em função das coordenadas x e y do plano, $w(x,y)$, podemos identificar as expressões das curvaturas média e gaussiana dessa superfície:

$$\begin{aligned} \frac{1}{\rho_1} + \frac{1}{\rho_2} = \Delta w : &\quad \text{dobro da curvatura média} \\ \frac{1}{\rho_1\rho_2} = w_{xx}w_{xy} - w_{xy}^2 : &\quad \text{curvatura gaussiana} \end{aligned} \tag{1.17}$$

em que ρ_1 e ρ_2 são os raios de curvatura (da placa).

Assim, temos que

$$U_{placa} = \frac{1}{2}\frac{Eh^3}{12(1-\nu^2)} \iint_\Omega \left[\left(\frac{1}{\rho_1} + \frac{1}{\rho_2}\right)^2 - 2(1-\nu)\frac{1}{\rho_1\rho_2}\right] dxdy \ .$$

1.1.5 O funcional Ação

Para uma placa de espessura h e densidade de área ρ ($\tilde{\rho} = \frac{\rho}{h}$), com base na equação (1.9), temos

$$T = \frac{1}{2}\iiint_V \tilde{\rho}\dot{\boldsymbol{f}}^T\dot{\boldsymbol{f}} \ dV = \frac{1}{2}\iint_\Omega \rho w_t^2 \ dxdy \ . \tag{1.18}$$

Com a expressão para a energia potencial obtida acima, para aplicar o Princípio de Hamilton, basta minimizar o funcional

$$J = \int_{t_0}^{t_1} (T-U)\ dt = \int_{t_0}^{t_1} \iint_\Omega \frac{1}{2}\rho {w_t}^2 - \frac{1}{2}\frac{Eh^3}{12(1-\nu^2)}\left[(\Delta w)^2 - 2(1-\nu)\left(w_{xx}w_{xy} - w_{xy}^2\right)\right] dxdy\ dt \ . \tag{1.19}$$

Segundo Courant e Hilbert (P.243) [2], o Princípio de Hamilton afirma que entre dois instantes de tempo t_0 e t_1, o movimento procede de tal modo que a integral do Lagrangiano (1.10), chamada de Ação, é estacionária em comparação à funções vizinhas $\bar{w}$ tais que $\bar{w}(t_0) = w(t_0)$ e $\bar{w}(t_1) = w(t_1)$.

Em outras palavras: a função que representa o movimento real faz a integral J estacionária, com relação às outras funções do movimento, numa vizinhança, que vão da posição inicial à final no mesmo intervalo de tempo.

Assim, esse princípio nos leva ao problema de encontrar uma função admissível para a qual a integral (1.19) seja estacionária, quando comparada com todas as funções vizinhas contínuas $\bar{w}(t,x,y)$ e com derivadas contínuas (até quarta ordem, nesse caso [2]).

Para isso calculamos a variação δJ e impomos que ela seja nula. Essa é uma condição necessária para determinar a função crítica, ou mínimo do funcional Ação.

1.1.6 Minimização do funcional Ação

Considere o funcional J de (1.19), em que w é função contínua com derivadas contínuas até quarta ordem. Sejam $\bar{w} = w(x,y,t) + \eta(x,y,t)\varepsilon$ (com $\varepsilon > 0$ e pequeno) e

$$\phi(\varepsilon) = J[w(x,y,t) + \eta(x,y,t)\varepsilon] = J[\bar{w}] \ .$$

Estamos buscando um ponto crítico w para J, entre as funções $\bar{w} = w + \eta\varepsilon$, em que a função η é tal que $\eta(x,y,t_0) = \eta(x,y,t_1) = 0$. Ou seja, todas as funções $\bar{w}$ coincidem com w em t_0 e em t_1. Estamos considerando que η, η_t, η_x, η_y, η_{xx}, η_{xy} e η_{yy} são funções contínuas e (a princípio, como não definimos valores para w em $\partial\Omega$) podem assumir qualquer valor em $\partial\Omega \times (t_0, t_1)$, em que $\partial\Omega$, fronteira do domínio Ω, é uma curva C^1 por partes.

Se w é ponto crítico de J, então,

$$\phi'(0) = \frac{d}{d\varepsilon}\phi(\varepsilon)\Big|_{\varepsilon=0} = 0 \ .$$

Lembrando que $D = \dfrac{Eh^3}{12(1-\nu^2)}$, explicitamos $\phi'(0)$:

$$\begin{aligned}
\phi'(\varepsilon) &= \frac{d}{d\varepsilon}\left[\int_{t_0}^{t_1}\iint_{\Omega} \frac{1}{2}\rho\bar{w}_t^2 - \frac{1}{2}D\left\{(\Delta\bar{w})^2 - 2(1-\nu)\left[\bar{w}_{xx}\bar{w}_{yy} - \bar{w}_{xy}^2\right]\right\} dxdy\ dt\right]\Bigg|_{\varepsilon=0} = \\
&= \int_{t_0}^{t_1}\iint_{\Omega} \rho w_t \cdot \eta_t - D\left[\Delta w \Delta\eta - (1-\nu)(w_{xx}\eta_{yy} + w_{yy}\eta_{xx} - 2w_{xy}\eta_{xy})\right] dxdy\ dt = \\
&= \underbrace{\int_{t_0}^{t_1}\iint_{\Omega} \rho w_t\eta_t - D\Delta w\Delta\eta\ dxdy\ dt}_{1} + \underbrace{D(1-\nu)\int_{t_0}^{t_1}\iint_{\Omega} \left[w_{xx}\eta_{yy} + w_{yy}\eta_{xx} - 2w_{xy}\eta_{xy}\right]\ dxdy\ dt}_{2} \ .
\end{aligned} \tag{1.20}$$

Parte 1:

$$\begin{aligned}
&\int_{t_0}^{t_1}\iint_{\Omega} \rho w_t\eta_t - D\Delta w\Delta\eta\ dxdy\ dt = \\
&\iint_{\Omega}\left(\int_{t_0}^{t_1} \rho w_t\eta_t dt\right) dxdy - \int_{t_0}^{t_1}\iint_{\Omega} D\Delta w\Delta\eta\ dxdy\ dt \ .
\end{aligned} \tag{1.21}$$

Fazendo integração por partes em t ($u = w_t$ e $dv = \eta_t$) na primeira parcela, temos

$$\iint_{\Omega}\left(\int_{t_0}^{t_1} \rho w_t\eta_t dt\right) dxdy = \iint_{\Omega} \rho\left[\underbrace{w_t\eta\Big|_{t_0}^{t_1}}_{=0} - \int_{t_0}^{t_1} w_{tt}\eta\ dt\right] dxdy = \int_{t_0}^{t_1}\iint_{\Omega} -\rho w_{tt}\eta\ dxdy\ dt \tag{1.22}$$

e, utilizando a Segunda Identidade de Green [1] para $\varphi = \Delta w$ e $\psi = \eta$ na segunda parcela, temos:

$$\int_{t_0}^{t_1} \iint_{\Omega} D\Delta w\Delta\eta \; dxdy \; dt = \int_{t_0}^{t_1} D\left[\iint_{\Omega} \Delta\Delta w\eta \; dxdy - \int_{\partial\Omega} \eta\nabla(\Delta w)\cdot\vec{n} \; ds + \int_{\partial\Omega} \Delta w\nabla\eta\cdot\vec{n} \; ds\right] dt =$$

$$= \int_{t_0}^{t_1} \iint_{\Omega} D\Delta\Delta w\eta \; dxdy \; dt + D\int_{t_0}^{t_1}\left[\int_{\partial\Omega} \eta\nabla(\Delta w)\cdot\vec{n} \; ds - \int_{\partial\Omega} \Delta w\nabla\eta\cdot\vec{n} \; ds\right] dt =$$

$$= \int_{t_0}^{t_1} \iint_{\Omega} D\Delta\Delta w\eta + D\int_{t_0}^{t_1}\left[\int_{\partial\Omega} \eta\frac{\partial}{\partial\vec{n}}(\Delta w) \; ds - \int_{\partial\Omega} \Delta w\frac{\partial}{\partial\vec{n}}\eta \; ds\right] dt \; . \tag{1.23}$$

Juntando as parcelas (1.22) e (1.23):

$$\int_{t_0}^{t_1} \iint_{\Omega} (-\rho w_{tt} - D\Delta\Delta w)\eta \; dxdy \; dt + D\int_{t_0}^{t_1}\left[\int_{\partial\Omega} \eta\frac{\partial}{\partial\vec{n}}(\Delta w) \; ds - \int_{\partial\Omega} \Delta w\frac{\partial}{\partial\vec{n}}\eta \; ds\right] dt \; . \tag{1.24}$$

Estamos consideramos que cada parte de $\partial\Omega$ está parametrizada pelo comprimento de arco, para considerar o vetor tangente unitário. As componentes do vetor tangente $\vec{t}$ à curva $\partial\Omega$ são $\left(\frac{dx}{ds}, \frac{dy}{ds}\right) = (-\sin\theta, \cos\theta)$ e as componentes do vetor normal $\vec{n}$ a curva $\partial\Omega$ são $\left(\frac{dy}{ds}, -\frac{dx}{ds}\right) = (\cos\theta, \sin\theta)$, em que θ é o ângulo entre a normal unitária $\vec{n}$ à fronteira $\partial\Omega$ e o eixo horizontal.

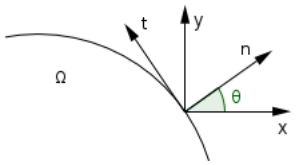


Figura 1.3: *Relação entre vetores normal e tangente à fronteira do domínio*

Parte 2:

$$D(1-\nu)\int_{t_0}^{t_1} \iint_{\Omega} w_{xx}\eta_{yy} + w_{yy}\eta_{xx} - 2w_{xy}\eta_{xy} \; dxdy \; dt =$$

$$D(1-\nu)\int_{t_0}^{t_1} \iint_{\Omega} \frac{\partial}{\partial x}[\eta_x w_{yy} - \eta_y w_{xy}] - \frac{\partial}{\partial y}[-(\eta_x w_{xy} - \eta_y w_{xx})] \; dxdy \; dt \; . \tag{1.25}$$

[1] A segunda identidade de Green afirma que, se $\Omega \in \mathbb{R}^2$ é um domínio limitado com fronteira $\partial\Omega$ de classe C^1, orientada de modo que a normal unitária $\vec{n}$ à curva $\partial\Omega$ aponta para fora de Ω, e se $\varphi, \psi \in C^2(\bar{\Omega})$, então vale a fórmula: $\iint_{\Omega} \varphi\Delta\psi - \psi\Delta\varphi \; dxdy = \int_{\partial\Omega} \varphi\nabla\psi\cdot\vec{n} - \psi\nabla\varphi\cdot\vec{n} \; ds = \int_{\partial\Omega} \varphi\frac{\partial\psi}{\partial\vec{n}} - \psi\frac{\partial\varphi}{\partial\vec{n}} \; ds$.

Aplicando o Teorema de Green[2] na integral dupla, (1.25) se torna

$$D(1-\nu)\int_{t_0}^{t_1} \iint_{\Omega} w_{xx}\eta_{yy} + w_{yy}\eta_{xx} - 2w_{xy}\eta_{xy}\; dxdy\; dt =$$
$$D(1-\nu)\int_{t_0}^{t_1} \int_{\partial\Omega} \left[-(\eta_y w_{xx} - \eta_x w_{xy})\frac{dx}{ds} + (\eta_x w_{yy} - \eta_y w_{xy})\frac{dy}{ds}\right] ds\; dt \tag{1.26}$$

e, como $\frac{dx}{ds}$ e $\frac{dy}{ds}$ são as componentes do vetor tangente (unitário) à curva $\partial\Omega$, obtemos

$$D(1-\nu)\int_{t_0}^{t_1} \int_{\partial\Omega} -(\eta_y w_{xx} - \eta_x w_{xy})(-\sin\theta)\; ds + \int_{\partial\Omega} (\eta_x w_{yy} - \eta_y w_{xy})\cos\theta\; ds\; dt = (*)\;. \tag{1.27}$$

Escrevendo η_x e η_y como combinação linear de $\frac{\partial\eta}{\partial\vec{n}}$ e $\frac{\partial\eta}{\partial s}$,

$$\begin{cases} \eta_x &= \cos\theta\dfrac{\partial\eta}{\partial\vec{n}} - \sin\theta\dfrac{\partial\eta}{\partial s} \\ \\ \eta_y &= \sin\theta\dfrac{\partial\eta}{\partial\vec{n}} + \cos\theta\dfrac{\partial\eta}{\partial s} \end{cases}$$

de (1.27) segue que:

$$(*) = D(1-\nu)\int_{t_0}^{t_1} \left[\int_{\partial\Omega} \frac{\partial\eta}{\partial\vec{n}}(w_{xx}\sin^2\theta - 2w_{xy}\cos\theta\sin\theta + w_{yy}\cos^2\theta)\; ds + \right.$$
$$\left. + \int_{\partial\Omega} \frac{\partial\eta}{\partial s}(w_{xx}\cos\theta\sin\theta + w_{xy}(\sin^2\theta - \cos^2\theta) - w_{yy}\sin\theta\cos\theta)\; ds \right] dt\;. \tag{1.28}$$

[2]O teorema de Green estabelece uma relação entre a integral de linha sobre uma curva fechada C e a integral dupla sobre a região D limitada por ela:

$$\int_C Pdx + Qdy = \iint_D \frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\; dxdy\;,$$

em que C é curva simples, fechada, contínua por partes e orientada positivamente; D é a região limitada pela curva C e, P e Q são funções de x e y, continuamente diferenciáveis e definidas num domínio aberto contendo D. No nosso caso, a região D é o domínio Ω e a curva C, sua fronteira $\partial\Omega$.

Integrando por partes em s, na segunda parcela temos:

$$
\begin{aligned}
(*) = D(1-\nu)\int_{t_0}^{t_1} \Bigg\{ & \int_{\partial\Omega} \frac{\partial\eta}{\partial\vec{n}}(w_{xx}\sin^2\theta - 2w_{xy}\cos\theta\sin\theta + w_{yy}\cos^2\theta)\; ds + \\
& + \underbrace{\eta\,[w_{xx}\cos\theta\sin\theta + w_{xy}(\sin^2\theta - \cos^2\theta) - w_{yy}\sin\theta\cos\theta]\Big|_{\partial\partial\Omega}}_{=0,\text{ pois } \partial(\partial\Omega)=\emptyset} \\
& - \int_{\partial\Omega} \eta\frac{d}{ds}[w_{xx}\cos\theta\sin\theta + w_{xy}(\sin^2\theta - \cos^2\theta) - w_{yy}\sin\theta\cos\theta] \Bigg\}\; dt \\
& = D(1-\nu)\int_{t_0}^{t_1} \Bigg\{ \int_{\partial\Omega} \frac{\partial\eta}{\partial\vec{n}}(w_{xx}\sin^2\theta - 2w_{xy}\cos\theta\sin\theta + w_{yy}\cos^2\theta)\; ds + \\
& - \int_{\partial\Omega} \eta\frac{d}{ds}[w_{xx} - w_{yy}]\cos\theta\sin\theta + w_{xy}(\sin^2\theta - \cos^2\theta) \Bigg\}\; dt\ . \qquad (1.29)
\end{aligned}
$$

Juntando as parcelas (1) e (2), temos então que :

$$
\begin{aligned}
0 = \phi'(0) = \int_{t_0}^{t_1} \Bigg\{ & \iint_{\Omega} -(\rho w_{tt} + D\Delta\Delta w)\;\eta\; dxdy + \\
& + D\int_{\partial\Omega} \eta\left\{\frac{\partial}{\partial\vec{n}}(\Delta w) - (1-\nu)\frac{d}{ds}\left[(w_{xx} - w_{yy})\cos\theta\sin\theta + w_{xy}(\sin^2\theta - \cos^2\theta)\right]\right\} ds \\
& + D\int_{\partial\Omega} \frac{\partial\eta}{\partial\vec{n}}\left[-\Delta w + (1-\nu)(w_{xx}\sin^2\theta - 2w_{xy}\cos\theta\sin\theta + w_{yy}\cos^2\theta)\right] ds \Bigg\}\; dt \qquad (1.30)
\end{aligned}
$$

isto é, $\phi'(0)$ deve ser zero para toda função η. Em particular, quando olhamos para a classe de funções η tais que $\eta\Big|_{\partial\Omega} = \dfrac{\partial\eta}{\partial\vec{n}}\Big|_{\partial\Omega} = 0$, as duas últimas parcelas de (1.30) desaparecem e a condição necessária para termos $\phi'(0) = 0$ é que a integral da primeira parcela de (1.30) seja zero. Com isso temos que

$$
\iint_{\Omega} -(\rho w_{tt} + D\Delta\Delta w)\;\eta\; dxdy = 0 \text{ se e somente se } (\rho w_{tt} + D\Delta\Delta w)\;\eta = 0
$$

para toda função η nessa classe particular. Se essa condição deve ser satisfeita para essa classe particular, então essa é uma condição que ainda deverá ser satisfeita quando consideramos que η é arbitrária. Assim, w, o ponto crítico de J (que minimiza J) deve satisfazer a equação diferencial parcial

$$
\rho w_{tt} + D\Delta\Delta w = 0\ . \qquad (1.31)
$$

Seguindo o mesmo raciocínio anterior (de considerar classes específicas para a função η), é possível entender por que, para que J seja estacionário, cada uma das integrais de bordo de (1.30) deve ser zero. Olhando para a classe de funções η tais que $\eta = 0$ e $\dfrac{\partial\eta}{\partial\vec{n}}$ é arbitrária em $\partial\Omega$, temos que as duas primeiras parcelas de (1.30) desaparecem. A primeira integral deve ser zero pelo que

foi exposto acima, e a condição $\eta\Big|_{\partial\Omega} = 0$ faz a segunda integral se anular. Assim, temos que

$$D\int\limits_{\partial\Omega} \frac{\partial\eta}{\partial\vec{n}}\left[-\Delta w + (1-\nu)(w_{xx}\sin^2\theta - 2w_{xy}\cos\theta\sin\theta + w_{yy}\cos^2\theta)\right]\ ds = 0$$

se torna condição necessária para que $\phi'(0)$ seja zero. Essa condição implica que a função entre colchetes

$$M(w) = -\Delta w + (1-\nu)(w_{xx}\sin^2\theta - 2w_{xy}\cos\theta\sin\theta + w_{yy}\cos^2\theta) \tag{1.32}$$

deve ser zero.

Por outro lado, quando olhamos para a classe de funções η tais que $\dfrac{\partial\eta}{\partial\vec{n}}\Big|_{\partial\Omega} = 0$, a última parcela de (1.30) desaparece e, como ja vimos que o anulamento da primeira parcela é condição necessária, obtemos, para essa classe de funções η, que o anulamento da segunda parcela de (1.30) também é condição necessária para que $\phi'(0)$ seja zero, i.e., obtemos que:

$$D\int\limits_{\partial\Omega} \eta\left\{\frac{\partial}{\partial\vec{n}}(\Delta w) - (1-\nu)\frac{d}{ds}\left[(w_{xx}-w_{yy})\cos\theta\sin\theta + w_{xy}(\sin^2\theta - \cos^2\theta)\right]\right\}\ ds = 0\ .$$

o que implica no anulamento da função entre colchetes

$$P(w) = \frac{\partial}{\partial\vec{n}}(\Delta w) + (1-\nu)\frac{d}{ds}[(w_{yy}-w_{xx})\cos\theta\sin\theta + w_{xy}(\cos^2\theta - \sin^2\theta)]\ . \tag{1.33}$$

Assim, mesmo quando voltamos a olhar η e $\dfrac{\partial\eta}{\partial\vec{n}}$ como funções arbitrárias em $\partial\Omega$, as duas condições obtidas acima continuam a ser necessárias para que J seja estacionária.

Concluímos então que para que w seja um ponto crítico de J, as três integrais de (1.30) devem se anular. O anulamento da primeira parcela nos fornece a EDP que governa o fenômeno e o anulamento das integrais de fronteira nos fornece as equações (1.32) e (1.33) que a função w deve satisfazer na fronteira $\partial\Omega$. Essas são as condições de contorno (naturais) associadas à EDP.

1.2 Condições de contorno

Considere novamente a equação (1.30), que deve valer para que J seja estacionário, e em que η é uma função arbitrária, contínua e com derivadas até segunda ordem contínuas em $\bar{\Omega}$.

Se a função $w(x,y,t)$ e sua derivada normal estiverem prescritas na fronteira, isto é, se

$$w(x,y,t)\Big|_{\partial\Omega} = w_0(x,y,t)\ \text{ e }\ \frac{\partial w}{\partial\vec{n}}\Big|_{\partial\Omega} = f_0(x,y,t)\ ,$$

então w está sendo procurada dentre as funções $\bar{w}$ tais que $\bar{w}(x,y,t)\Big|_{\partial\Omega} = w_0(x,y,t)$ e $\dfrac{\partial\bar{w}}{\partial\vec{n}}\Big|_{\partial\Omega} = f_0(x,y,t)$. Desse modo,

$$\bar{w}(x,y,t) = w(x,y,t) + \eta(x,y,t) = w_0(x,y,t) \Longleftrightarrow \eta = 0 \text{ em } \partial\Omega$$

e

$$\frac{\partial\bar{w}}{\partial\vec{n}}(x,y,t) = \frac{\partial w}{\partial\vec{n}}(x,y,t) + \frac{\partial\eta}{\partial\vec{n}}(x,y,t) = f_0 \Longleftrightarrow \frac{\partial\eta}{\partial\vec{n}} = 0 \text{ em } \partial\Omega\ .$$

Ou seja, quando a condição de contorno está prescrita dessa forma, a função η deve ser tal que $\eta = \dfrac{\partial \eta}{\partial \vec{n}} = 0$ na fronteira $\partial\Omega$. Isso garante que as integrais de fronteira de (1.30) se anulem e que w torne J estacionário. Este caso caracteriza a condição de contorno *bordo grampeado*. Neste trabalho, usaremos essa expressão para o caso particular em que $w = \dfrac{\partial w}{\partial \vec{n}} = 0$ em $\partial\Omega$. Reservaremos, a partir daqui, a expressão *bordo preso* para o caso em que somente $w = 0$ em $\partial\Omega$.

Se a função w não estiver prescrita na fronteira $\partial\Omega$, então η e $\frac{\partial \eta}{\partial \vec{n}}$ não serão necessariamente nulas em $\partial\Omega$. Para que as integrais de fronteira de (1.30) se anulem é necessário que as expressões $P(w)$ e $M(w)$ se anulem. Este é o caso de vibração de uma placa com a borda (fisicamente) livre (não presa e não apoiada) e será chamado de condição de *bordo livre*. Esse caso onde não são impostas, previamente, condições para w na fronteira de Ω, é conhecido como problema de *fronteira livre* [2], pois a fronteira do domínio está livre de condições preestabelecidas. As condições $P(w) = 0$ e $M(w) = 0$ que surgem naturalmente da minimização do funcional J são chamadas de *condições de contorno naturais*.

Se o valor de w estiver prescrito na fronteira e $\dfrac{\partial \eta}{\partial \vec{n}}$ for arbitrária, então, para que a integral da última parcela de (1.30) se anule, é necessário que $M(w) = 0$. O caso particular em que $w|_{\partial\Omega} = 0$ e $M(w) = 0$ é a condição de contorno chamada de *bordo simplesmente suportado*.

Nas próximas seções, onde trabalharemos com a solução do problema, optamos por trabalhar com a condição de contorno de *bordo grampeado*. Essa condição de contorno não expressa o problema original sobre as vibrações de placas finas segundo os experimentos de E. Chladni. Nesse caso, deveríamos estar considerando a condição de *bordo livre*, com o centro da placa preso. Optamos por essa condição, pelo menos para um estudo inicial, pelo fato de esta apresentar equações mais simples a serem satisfeitas na fronteira. Porém o tratamento desse caso será feito de modo que seja útil para o trabalho no caso de bordo livre.

Observação: As condições naturais de contorno podem ser obtidas apenas do funcional energia potencial. Em grande parte da literatura estudada, observamos que, apesar de termos um problema temporal e espacial, na busca das condições de contorno naturais, utiliza-se apenas o funcional energia potencial (ao invés de considerar juntamente a energia cinética e, portanto, a energia total). Entretanto as condições de contorno naturais dizem respeito a toda a equação diferencial parcial (com sua parte temporal) e, portanto, à energia total do sistema. O motivo de podermos olhar apenas para a parte espacial do funcional energia, e considerar, portanto, somente a energia potencial na busca por condições de contorno naturais é que, ao fazermos a variação do Lagrangiano (como foi feito na subseção acima), é possível integrar a parcela da energia cinética separadamente e por partes na variável temporal t. Essa integral resultará num termo de fronteira que envolve $\eta(x,t)\Big|_{t_0}^{t_1}$ (que por definição do princípio de Hamilton dever ser zero), e numa parte onde aparece o termo temporal da EDP (1.24). Deixando a integral temporal para fora (trocando-a com a integral da parte espacial) é possível tratar a parte espacial de maneira independente do tempo, e assim, obter a parte espacial da EDP e as condições de contorno relativas à essa parte, como é possível ver em (1.24) e (1.27). A conclusão é que mesmo trabalhando com a minimização do funcional Ação relativa à energia total, em algum momento, trabalhamos apenas com a parte espacial da energia potencial (dentro da integral temporal, mas independente dela) do problema, de onde surgem as condições de contorno da parte espacial. Daí percebemos que poderíamos ter buscado as condições de contorno somente com o funcional energia potencial.

1.3 Solução

Nesta seção, resolveremos, até certo ponto, a equação diferencial (1.1), considerando a condição inicial $w(x,y,t)=0$ em $t=0$ e a condição de contorno *bordo grampeado*: $w=0$ e $\dfrac{\partial w}{\partial \vec{n}}=0$.

Usamos a técnica de separação de variáveis, supondo que a função $w(x,y,t)$ procurada é da forma

$$w(x,y,t)=u(x,y)\cdot v(t) \tag{1.34}$$

e substituindo essa equação em (1.1):

$$\begin{aligned} D\Delta\Delta w+\rho w_{tt} &= D\Delta\Delta[u(x,y)\cdot v(t)]+\rho\frac{\partial^2[u(x,y)\cdot v(t)]}{\partial t^2}= \\ &= D\Delta\Delta u(x,y).v(t)+\rho v''(t)\cdot u(x,y)\ . \end{aligned}$$

Impondo que

$$D\Delta\Delta w+\rho w_{tt}=0$$

obtemos

$$D\Delta\Delta u(x,y)\cdot v(t)=-\rho v''(t)\cdot u(x,y)\ .$$

Como estamos buscando uma solução não trivial, consideramos as funções $u(x,y)$ e $v(t)$ não nulas e podemos dividir a expressão acima por $u(x,y)\cdot v(t)$, obtendo:

$$D\frac{\Delta\Delta u(x,y)}{u(x,y)}=-\rho\frac{v''(t)}{v(t)}\ ,$$

que nos leva à condição

$$\frac{\Delta\Delta u(x,y)}{u(x,y)}=-\frac{\rho}{D}\frac{v''(t)}{v(t)}=cte=\lambda\ ,$$

uma vez que os lados direito e esquerdo da igualdade dependem de variáveis distintas.

Então temos que :

$$\frac{\Delta\Delta u(x,y)}{u(x,y)}=\lambda \tag{1.35}$$

e

$$-\frac{\rho}{D}\frac{v''(t)}{v(t)}=\lambda\ . \tag{1.36}$$

Da equação (1.35), $u(x,y)$ deverá satisfazer o seguinte problema de autovalor nas variáveis x e y :

$$\Delta\Delta u(x,y)=\lambda u(x,y)\ ,$$

com condição de contorno

$$u(x,y)=0\ \text{ e }\ \frac{\partial u(x,y)}{\partial \vec{n}}=0\ \text{ para }\ (x,y)\in\partial\Omega.$$

A 2ª identidade de Green é útil neste momento, para entendermos algumas restrições para λ . Da expressão da 2ª Identidade de Green para $\varphi=u$ e $\psi=\Delta u$ temos que

$$\iint_\Omega u\Delta(\Delta u)-\Delta u\Delta u\ dxdy=\int_{\partial\Omega}u\frac{\partial}{\partial\vec{n}}\Delta u-\Delta u\frac{\partial u}{\partial\vec{n}}\ ds\ .$$

Aplicando a definição do problema de autovalor juntamente com as condições de contorno, temos que

$$\lambda \iint_\Omega u^2 = \iint_\Omega (\Delta u)^2 \; dxdy \; .$$

Como $u \neq 0$ (pois não consideramos a solução nula como autofunção), concluímos que $\lambda > 0$.

A partir daqui não é possível fazer a separação das variáveis espaciais x e y, de modo semelhante ao que foi feito no início da seção para separar variáveis espaciais e temporais, ou como é possível fazer no caso da equação da onda no retângulo (membrana vibrante). Assim, optamos por buscar uma solução numérica para esse problema de autovalor, utilizando o Método de Elementos Finitos, num domínio Ω que seja uma região simples qualquer de bordo poligonal.

De (1.36), obtemos o problema de autovalor na variável t:

$$-v''(t) = \frac{\lambda D}{\rho} v(t) \; ,$$

que resulta na equação diferencial ordinária

$$v''(t) + \omega^2 v(t) = 0$$

com

$$\omega^2 \stackrel{def}{=} \frac{\lambda D}{\rho} \; ,$$

uma vez que $\lambda > 0$ e, cuja solução $v(t)$, é uma combinação linear de funções da forma

$$e^{\omega i t} \quad \text{e} \quad e^{-\omega i t} \; .$$

Assim $v(t)$ deverá ser uma função do tipo

$$v(t) = A\cos(\omega t + \phi) \; , \tag{1.37}$$

em que ϕ é fase e ω representa a frequência angular do movimento. Além disso, λ depende dos parâmetros da EDP original: $\lambda = \dfrac{\omega^2 \rho}{D}$.

1.3.1 Análise e forma de apresentação dos resultados

O objetivo aqui é falar da apresentação dos resultados para o problema de autovalor nas variáveis espaciais:

$$\Delta\Delta u = \lambda u \; .$$

É bom ter em conta como os autovalores e, portanto, as frequências de vibração, são influenciados por variações no tamanho (homotetias) da placa. De forma geral, quanto maior a placa, menor é a frequência de vibração. Dessa forma, escolher uma maneira padronizada de se apresentar os resultados obtidos é importante, para que esses possam ser comparados com resultados provenientes de outros tamanhos de placas. Falaremos então sobre como apresentar os autovalores de uma forma padronizada (normalizada), de modo que os valores não dependam do tamanho da placa, mas apenas de seu formato.

Ao resolver o problema de autovalor $\Delta\Delta u = \lambda u$ obteremos autovalores λ e, para cada um deles,

uma autofunção associada $u(x, y)$, que nos fornece um modo de vibração. Vimos na seção acima uma expressão para o autovalor, que depende das constantes da EDP e da frequência de vibração:

$$\lambda = \frac{\omega^2 \rho}{D} \ . \tag{1.38}$$

Para observar como os autovalores e, portanto, as frequências de vibração varias sob homotetias, consideramos uma placa que chamaremos de "placa zero" de dimensão a. Aqui chamaremos de dimensão da placa alguma medida (cuja escolha depende do formato da placa) escolhida para caracterizar o tamanho da placa (por exemplo, se a placa for quadrada a dimensão pode ser o lado, se a placa for retangular, usamos o lado menor, no caso de um triângulo isósceles, usamos o tamanho de um dos lados congruentes).

Seja P uma placa de dimensão a e P' a placa α vezes maior, imagem da homotetia:

$$\begin{aligned} H: \ \mathbb{R}^2 &\longrightarrow \mathbb{R}^2 \\ (x, y) &\longmapsto (\alpha x, \alpha y) \end{aligned} \tag{1.39}$$

Seja $w_0(x, y, t) = u_0(x, y) \cdot v_0(t)$ a solução da EDP na placa P. Defina

$$w(x, y, t) = w_0 \left(\frac{x}{\alpha}, \frac{y}{\alpha}, ct \right), \tag{1.40}$$

em que c é um fator que vai indicar a influência da homotetia sobre a parte temporal da solução, e portanto, sobre a frequência de vibração ω. Essa constante depende do parâmetro α da homotetia e será determinada a seguir.

Na placa P, $u_0(x, y)$ e $v_0(t)$ satisfazem

$$\Delta\Delta u_0 = \lambda_0 u_0 \ \text{ e } \ v_0''(t) = \frac{-\lambda_0 D}{\rho} v_0(t)$$

e, portanto, $v_0(t) = A\cos(\omega_0 t)$ com $\omega_0 = \sqrt{\frac{\lambda_0 D}{\rho}}$ (estabelecemos a fase $\phi = 0$, sem perda de generalidade).

Calculamos as derivadas de w:

$$\begin{aligned} \frac{\partial^2}{\partial t^2} w(x, y, t) &= c^2 \frac{\partial^2}{\partial t^2} w_0 \left(\frac{x}{\alpha}, \frac{y}{\alpha}, ct \right) \ , \\ \Delta w(x, y, t) &= \frac{1}{\alpha^2} \Delta w_0 \left(\frac{x}{\alpha}, \frac{y}{\alpha}, ct \right) \ , \end{aligned}$$

e

$$\Delta\Delta w = \frac{1}{\alpha^4} \Delta\Delta w_0 \ . \tag{1.41}$$

Na placa P', w é solução de $D\Delta\Delta w + \rho \frac{\partial^2}{\partial t^2} w = 0$ se e somente se w_0 é solução de $D\frac{1}{\alpha^4}\Delta\Delta w_0 + \rho c^2 \frac{\partial^2}{\partial t^2} w_0 = 0$.

Então, w é solução se e somente se

$$c^2\alpha^4 = 1 \Leftrightarrow c = \frac{1}{\alpha^2} \ .$$

Assim,

$$w(x,y,t) = u(x,y)\cdot v(t),\ u(x,y) = u_0\left(\frac{x}{\alpha},\ \frac{y}{\alpha}\right) \quad \text{e} \quad v(t) = v_0\left(\frac{t}{\alpha^2}\right) \ .$$

Da última expressão, temos que

$$\cos(\omega t) = \cos\left(\frac{\omega_0}{\alpha^2}t\right) \ ,$$

logo

$$\omega = \frac{\omega_0}{\alpha^2} \tag{1.42}$$

Da relação $\omega_0 = \sqrt{\frac{\lambda_0 D}{\rho}}$, temos que $\lambda_0 = \omega_0^2\frac{\rho}{D}$ e então,

$$\lambda = \omega^2\frac{\rho}{D} = \frac{\omega_0^2}{\alpha^4}\frac{\rho}{D} = \frac{\lambda_0}{\alpha^4} \ . \tag{1.43}$$

Assim, concluímos que a homotetia de fator α no tamanho da placa faz a frequência ω ficar dividida por α^2 e o autovalor λ por α^4. Se a placa P tem tamanho a, a placa P' terá tamanho αa. Se ω_0 e λ_0 são a frequência e autovalor encontrado em P, a frequência ω e autovalor λ encontrados em P' são : $\omega = \frac{\omega_0}{\alpha^2}$ e $\lambda = \frac{\lambda_0}{\alpha^4}$.

Ou seja com o aumento no tamanho da placa, temos a diminuição do autovalor e da frequência, e portanto essas duas quantidades que caracterizam a vibração da placa dependem do tamanho da placa.

Uma maneira de apresentar os resultados obtidos para os autovalores é, ao invés de apresentar os valores em si, apresentar uma quantidade normalizada que represente esse parâmetro e o modo de vibração, que independa do tamanho da placa, mas apenas de seu formato. A quantidade escolhida para apresentar os resultados obtidos de autovalor será, neste trabalho, chamada de *coeficiente de frequência* e será representada pela letra κ:

$$\kappa = \omega\sqrt{\frac{\rho}{D}}a^2 \ . \tag{1.44}$$

Note que κ é uma constante adimensional e ela independe do tamanho da placa. Se κ_0 e κ são os coeficientes de frequência das placas P e P', respectivamente, temos, para a placa P,

$$\kappa_0 = \omega_0\sqrt{\frac{\rho}{D}}a^2$$

e, para a placa P',

$$\kappa = \omega\sqrt{\frac{\rho}{D}}(\alpha a)^2 = \frac{\omega_0}{\alpha^2}\sqrt{\frac{\rho}{D}}\alpha^2 a^2 = \omega_0\sqrt{\frac{\rho}{D}}a^2 = \kappa_0 \ .$$

Cada autovalor está associado a um modo de vibração. Para placas de mesmo formato, os autovalores se alteram, mas os modos de vibração não. Apresentar então o coeficiente de frequência κ nos dá uma única constante associada a cada modo de vibração.

Essa constante κ será utilizada na apresentação dos resultados obtidos.

1.4 Formulação variacional ou fraca do problema de autovalor

Estamos interessados em resolver, numericamente, o problema de autovalor

$$\Delta\Delta u = \lambda u \quad \text{em} \quad \Omega \subset \mathbb{R}^2 \ , \tag{1.45}$$

com condição de contorno

$$u = 0 \ \text{ e } \ \frac{\partial u}{\partial \vec{n}} = 0 \ \text{ em } \Omega. \tag{1.46}$$

Para resolver esse problema pelo método de Elementos Finitos, precisamos escrevê-lo numa forma diferente, chamada de *forma fraca* ou *variacional*. É nessa forma que o problema será discretizado, e a partir dela buscaremos, numericamente, uma solução.

Uma solução clássica de (1.45) com condição de contorno (1.46) é uma função quatro vezes continuamente diferenciável em Ω e pelo menos continuamente diferenciável em $\partial\Omega$. Considere uma função $v \in C(\Omega)$. Multiplicando a equação (1.45) por v, podemos integrá-la e obtemos

$$\iint_\Omega (\Delta\Delta u)v \ dxdy = \iint_\Omega \lambda uv \ dxdy \ . \tag{1.47}$$

Por outro lado, se $u \in C^4(\Omega)$ e $v \in C(\Omega)$, e vale (1.47), então u é solução clássica de (1.45). Se v for $C^2(\bar{\Omega})$, então é possível usar a 2ª Identidade de Green e obter

$$\iint_\Omega \Delta u \Delta v \ dxdy = \iint_\Omega \lambda uv \ dxdy \ ,$$

uma equação que faz sentido se $u, v \in C^2(\Omega)$, o que é uma exigência menor (para u) do que a que tínhamos antes em Ω. A ideia de apresentar o problema clássico (1.45) numa forma integral e relaxar condições sobre diferenciabilidade da solução é o que está por trás da *forma fraca*.

1.4.1 Conceito básicos

Nesta subseção, estabeleceremos conceitos e definições importantes para trabalharmos com a forma fraca de uma EDP. Consideraremos $\Omega \subset \mathbb{R}^n$ um subconjunto aberto e limitado e $x = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$. As definições e considerações a seguir foram baseadas em [7] e [1]. No tratamento clássico de EDP's, as soluções devem ser contínuas e por isso podemos trabalhar com espaços $C^k(\bar{\Omega})$, que contêm derivadas contínuas até ordem k, num certo domínio Ω. Quando a forma forte de uma EDP é passada para a forma fraca ou variacional, então, ao invés de diferenciabilidade ponto a ponto ser exigida, é preciso apenas garantir a existência de algumas integrais que contêm a função desconhecida e certas derivadas dela. Assim, faz sentido usar espaços de funções que são especialmente adequados à essa situação. Esses espaços, são os *Espaços de Sobolev*, que consistem numa generalização dos espaços L_p. Neste trabalho consideraremos a classe particular de espaços de Sobolev que consiste numa generalização do espaço de Hilbert

$$L_2(\Omega) = \left\{ v : \Omega \subset \mathbb{R}^n \longrightarrow \mathbb{R} : \int_\Omega |v(x)|^2 \ dx < +\infty \right\} \ ,$$

cuja norma é definida por

$$||v||_{L_2(\Omega)} = \left(\int_\Omega |v(x)|^2 \ dx \right)^{1/2} \ ,$$

proveniente do produto interno

$$\langle u, v\rangle = \int_\Omega u(x)v(x)\ dx\ , \qquad \forall\ u, v \in L_2(\Omega)\ .$$

Vale ressaltar que $L_2(\Omega)$ é, na verdade, o conjunto das classes de equivalência das funções que são iguais a menos de um conjunto de medida nula. Apesar disso, neste trabalho, trataremos os elementos de $L_2(\Omega)$ como funções, exceto nos casos que se fizerem necessários.

Precisaremos também do conceito de *derivada fraca* ou *generalizada*, definição importante para a definição de Espaços de Sobolev. Para isso, considere as definições a seguir.

Denotaremos por $cl_V A$ o fecho de um conjunto $A \subset V$ com respeito a topologia do espaço V. Para $v \in C(\overline{\Omega})$, o *suporte* de v é definido como

$$supp\ v = cl_{\mathbb{R}^n}\{x \in \Omega : v(x) \neq 0\}\ .$$

Definimos também

$$C_0^\infty(\Omega) = \{v \in C^\infty(\Omega) : supp\ v \subset \Omega\}\ .$$

Em particular, se $v \in C_0^\infty$ então v se anula em $\partial\Omega$.

Também é importante o teorema de integração por partes em várias variáveis. Para funções arbitrárias $u \in C^1(\overline{\Omega})$ e $v \in C_0^\infty(\Omega)$ temos que

$$\int_\Omega u\frac{\partial v}{\partial x_i}\ dx = \int_{\partial\Omega} uvn_i\ ds - \int_\Omega v\frac{\partial u}{\partial x_i}\ dx\ , \tag{1.48}$$

em que n_i é a i-ésima componente do vetor normal $\vec{n}$ à fronteira $\partial\Omega$, apontando para fora. Como $v \in C_0^\infty(\Omega)$, então a integral de fronteira se anula, e obtemos a expressão

$$\int_\Omega u\frac{\partial v}{\partial x_i}\ dx = -\int_\Omega v\frac{\partial u}{\partial x_i}\ dx\ , \tag{1.49}$$

que é o ponto de partida para a definição de *derivada fraca* ou *generalizada*.

A seguinte notação, chamada de *multi-índice* será usada para denotar derivadas parciais de uma função $f : \mathbb{R}^n \to \mathbb{R}$. Um *múlti-índice* α é um vetor de $\mathbb{N}^n$, $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$, de tamanho $|\alpha| = \alpha_1 + \alpha_2 + ... + \alpha_n$. A derivada parcial de f, de ordem $|\alpha|$, com respeito ao multi-índice α é denotada por

$$D^\alpha f = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1}\partial x_1^{\alpha_2}...\partial x_n^{\alpha_n}} f\ .$$

A expressão (1.49), motiva a seguinte definição.

Definição 1.4.1. *Dizemos que uma função integrável u é diferenciável no sentido fraco ou generalizado, com relação ao multi-índice α, se existir uma função integrável w tal que*

$$\int_\Omega wv\ dx = (-1)^{|\alpha|}\int_\Omega D^\alpha vu\ dx\ , \qquad \forall v \in C_0^\infty(\Omega)\ . \tag{1.50}$$

A função $w = D^\alpha u$ é chamada de derivada fraca (ou generalizada) *da função u com respeito ao multi-índice α.*

Agora estamos prontos para definir o espaço de Sobolev que vamos utilizar.

Definição 1.4.2. *Seja $l \in \mathbb{N}$ $(l \geq 0)$. Considere o espaço de todas as funções de $L_2(\Omega)$ cujas derivadas fracas até ordem l existem e pertencem a $L_2(\Omega)$. Esse subespaço é chamado de* espaço de Sobolev $H^l(\Omega)$. *Em outras palavras:*

$$H^l(\Omega) = \left\{ v \in L_2(\Omega) : \int_\Omega |D^\alpha v|^2 \, dx < +\infty, \qquad \forall \ \alpha \ tal \ que \ |\alpha| \leq l \right\} .$$

$H^l(\Omega)$ é um espaço de Hilbert, com produto interno dado por

$$\langle u, v \rangle = \int_\Omega \Big(\sum_{|\alpha| \leq l} D^\alpha u D^\alpha v \Big) \, dx \ ,$$

e norma dada por

$$||u||_{H^l(\Omega)} = \langle u, u \rangle^{1/2} = \left(\int_\Omega \Big(\sum_{|\alpha| \leq l} D^\alpha u D^\alpha v \Big) \, dx \right)^{1/2} .$$

Também é frequente o uso da semi-norma

$$|u|_{H^l(\Omega)} = \left(\int_\Omega \Big(\sum_{|\alpha| = l} D^\alpha u D^\alpha v \Big) \, dx \right)^{1/2} .$$

É possível mostrar que $C^\infty(\Omega)$ é denso em $H^l(\Omega)$. Isso quer dizer que funções em espaços de Sobolev podem ser aproximadas por funções diferenciáveis no sentido clássico. Definimos $H_0^l(\Omega)$ como o fecho de $C_0^\infty(\Omega)$ em $H^l(\Omega)$ e, para $l = 1$ e $l = 2$, seguem as seguintes caracterizações [1]:

$$H_0^1(\Omega) = \big\{ u \in H^1(\Omega); \ u = 0 \text{ sobre } \partial\Omega \big\} \ ,$$

e

$$H_0^2(\Omega) = \big\{ u \in H^2(\Omega); \ u = \partial_n u = 0 \text{ sobre } \partial\Omega \big\} \ .$$

No tratamento de problemas elípticos de ordem 4, é adequado o uso de $H^2(\Omega)$ ou $H_0^2(\Omega)$, dependendo das condições de contorno consideradas.

1.4.2 Forma fraca

Agora que já sabemos o espaços de funções onde buscaremos nossa solução, vamos formular a forma fraca para o problema (1.45). Por enquanto, diferentemente do exemplo na introdução desta seção, não vamos prescrever condições de contorno para os valores de u e de $\frac{\partial u}{\partial \vec{n}}$, consideraremos esse problema de autovalor "livre de condição de fronteira" i.e., sem predeterminar a condição de fronteira. Desse modo, como vimos na Seção (1.2), chegaremos as expressões $M(u)$ e $P(u)$ das condições de contorno naturais para u.

[1] Essas caracterizações decorrem do *Lema do Traço*, um resultado que permite caracterizar/atribuir valores de u e de sua derivada ao longo da fronteira $\partial\Omega$ do domínio aberto Ω. Para mais detalhes a respeito desteLema, as referências [3] e [1] podem ser consultadas.

A forma fraca pode ser obtida multiplicando-se a equação (1.45) por uma função $v \in C^2(\bar{\Omega})$ e integrando a expressão resultante em Ω.

$$\iint_\Omega (\Delta\Delta u)v \; dxdy = \lambda \iint_\Omega uv \; dxdy \; . \tag{1.51}$$

Aplicando a 2ª Identidade de Green para $\varphi = \Delta u$ e $\psi = v$, obtemos:

$$\iint_\Omega \Delta u \Delta v \; dxdy - \int_{\partial\Omega} \Delta u \frac{\partial v}{\partial \vec{n}} \; ds + \int_{\partial\Omega} v \frac{\partial}{\partial \vec{n}}(\Delta u) \; ds = \lambda \iint_\Omega uv \; dxdy \; . \tag{1.52}$$

Antes de prosseguirmos, é bom nos determos um pouco sobre essa equação. Da mesma maneira que analisamos (1.30) à luz das condições de contorno, o mesmo podemos tentar fazer agora.

Se nosso objetivo é procurar funções u tais que $u = \dfrac{\partial u}{\partial \vec{n}} = 0$ em Ω, então faz sentido exigir o mesmo para v: $v = \dfrac{\partial v}{\partial \vec{n}} = 0$. Neste caso as duas integrais de fronteira seriam nulas, restando-nos a equação

$$\iint_\Omega \Delta u \Delta v \; dxdy = \iint_\Omega \lambda uv \; dxdy \tag{1.53}$$

como forma fraca de (1.45).

Caso contrário, $v = 0$ ou $\dfrac{\partial v}{\partial \vec{n}} = 0$ não serão exigidos, o que não permitirá concluir que pelo menos uma das integrais é nula. Uma alternativa para manter (1.53) como forma fraca quando nem v nem $\dfrac{\partial v}{\partial \vec{n}}$ podem ser consideradas nulas é impor as condições de contorno

$$\begin{aligned} \Delta u &= 0 \; , \\ \frac{\partial}{\partial \vec{n}}(\Delta u) &= 0 \; . \end{aligned}$$

que seriam de 2ª e 3ª ordens.

O problema dessa abordagem é que não nos livraríamos das condições naturais discutidas na Seção (1.2), e ficaríamos com quatro condições no total, o que poderia nos levar a um problema sem solução.

Para compatibilizar tudo isso olhamos para (1.32) e (1.33) e, agora sem a dependência temporal, definimos

$$\begin{aligned} M(u) &= -\Delta u + (1-\nu)(u_{xx}\sin^2\theta - 2u_{xy}\cos\theta\sin\theta + u_{yy}\cos^2\theta) \\ P(u) &= \frac{\partial}{\partial \vec{n}}(\Delta u) + (1-\nu)\frac{d}{ds}[(u_{yy}-u_{xx})\cos\theta\sin\theta + u_{xy}(\cos^2\theta - \sin^2\theta)] \; . \end{aligned} \tag{1.54}$$

Colocando Δu em função de $M(u)$ e $\dfrac{\partial}{\partial \vec{n}}(\Delta u)$ em função de $P(u)$, o lado esquerdo de (1.52) se

torna

$$\begin{aligned}\iint_{\Omega} \Delta u\Delta v\ dxdy - \int_{\partial\Omega} \frac{\partial v}{\partial \vec{n}}\ [-M(u) + (1-\nu)(u_{xx}\sin^2\theta - 2u_{xy}\cos\theta\sin\theta + u_{yy}\cos^2\theta)]\ ds \\ + \int_{\partial\Omega} v\ [P(u) - (1-\nu)]\frac{d}{ds}[(u_{yy}-u_{xx})\cos\theta\sin\theta + u_{xy}(\cos^2\theta - \sin^2\theta)]\ ds = \\ \iint_{\Omega} \Delta u\Delta v\ dxdy - \int_{\partial\Omega} (1-\nu)\frac{\partial v}{\partial \vec{n}}\left[u_{xx}\sin^2\theta - 2u_{xy}\cos\theta\sin\theta + u_{yy}\cos^2\theta\right]\ ds + \\ + \int_{\partial\Omega} (1-\nu)\ v\frac{d}{ds}\left[(u_{xx}-u_{yy})\cos\theta\sin\theta + u_{xy}(\sin^2\theta - \cos^2\theta)\right] ds + \\ + \int_{\partial\Omega} \frac{\partial v}{\partial \vec{n}} M(u) + \int_{\partial\Omega} v\ P(u)\ ds \end{aligned} \tag{1.55}$$

A partir daqui trabalharemos com a segunda e terceira parcelas de (1.55). Para facilitar, chamaremos essas parcelas de (A) e (B).

$$\begin{aligned}\underbrace{\int_{\partial\Omega} (1-\nu)v\frac{d}{ds}\left[(u_{xx}-u_{yy})\cos\theta\sin\theta + u_{xy}(\sin^2\theta - \cos^2\theta)\right] ds +}_{(A)} \\ \underbrace{- \int_{\partial\Omega} (1-\nu)\frac{\partial v}{\partial \vec{n}}\left[u_{xx}\sin^2\theta - 2u_{xy}\cos\theta\sin\theta + u_{yy}\cos^2\theta\right]\ ds}_{(B)}\end{aligned} \tag{1.56}$$

Em (A), vamos integrar por partes em s, e usar como as derivadas parciais de v nas direções canônicas se relacionam com as derivadas parciais de v nas direções normal e tangente:

$$\begin{bmatrix}\partial_{\vec{n}} v \\ \partial_s v\end{bmatrix} = \begin{bmatrix}\cos\theta & \sin\theta \\ -\sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}v_x \\ v_y\end{bmatrix}\ . \tag{1.57}$$

Daí, temos que:

$$
\begin{aligned}
(A) &= \int_{\partial\Omega} (1-\nu) v \frac{d}{ds}\left[(u_{xx}-u_{yy})\cos\theta\sin\theta + u_{xy}(\sin^2\theta-\cos^2\theta)\right] ds = \\
&= (1-\nu)\underbrace{v\left[\sin\theta\cos\theta(u_{xx}-u_{yy}) + (\sin^2\theta-\cos^2\theta)u_{xy}\right]\Big|_{\partial(\partial\Omega)}}_{=0 \text{ pois } \partial(\partial\Omega)=\emptyset} + \\
&\quad (1-\nu)\int_{\partial\Omega} \frac{dv}{ds}\left[\sin\theta\cos\theta(u_{yy}-u_{xx}) + (\cos^2\theta-\sin^2\theta)u_{xy}\right] ds = \\
&= (1-\nu)\int_{\partial\Omega} (-\sin\theta v_x + \cos\theta v_y)\left[\sin\theta\cos\theta(u_{yy}-u_{xx}) + (\cos^2\theta-\sin^2\theta)u_{xy}\right] ds. \\
(A) &= (1-\nu)\int_{\partial\Omega} -\sin^2\theta\cos\theta v_x(u_{yy}-u_{xx}) + (-\sin\theta)v_x(\cos^2\theta-\sin^2\theta)u_{xy}\ ds + \\
&\quad (1-\nu)\int_{\partial\Omega} \sin\theta\cos^2\theta v_y(u_{yy}-u_{xx}) + (\cos\theta)v_y(\cos^2\theta-\sin^2\theta)u_{xy}\ ds\ . \qquad (1.58)
\end{aligned}
$$

Em (B), usando novamente a relação (1.57):

$$
\begin{aligned}
(B) &= -\int_{\partial\Omega} (1-\nu)\frac{\partial v}{\partial \vec{n}}\left[u_{xx}\sin^2\theta - 2u_{xy}\cos\theta\sin\theta + u_{yy}\cos^2\theta\right]\ ds = \\
&-(1-\nu)\int_{\partial\Omega} (\cos\theta v_x + \sin\theta v_y)\left[u_{xx}\sin^2\theta - 2u_{xy}\cos\theta\sin\theta + u_{yy}\cos^2\theta\right]\ ds = \\
&-(1-\nu)\int_{\partial\Omega} v_x u_{xx}\cos\theta\sin^2\theta + v_x u_{yy}\cos\theta\cos^2\theta - 2v_x u_{xy}\cos^2\theta\sin\theta\ ds + \\
&-(1-\nu)\int_{\partial\Omega} v_y u_{xx}\sin\theta\sin^2\theta + v_y u_{yy}\sin\theta\cos^2\theta - 2v_y u_{xy}\cos\theta\sin^2\theta\ ds \qquad (1.59)
\end{aligned}
$$

Somando (A) e (B), e colocando w_{xx}, w_{xy} e w_{yy} em evidência temos:

$$
\begin{aligned}
(A)+(B) =& (1-\nu)\int\limits_{\partial\Omega} -\sin^2\theta\cos\theta v_x(u_{yy}-u_{xx})+(-\sin\theta)v_x(\cos^2\theta-\sin^2\theta)u_{xy}\ ds + \\
& (1-\nu)\int\limits_{\partial\Omega} \sin\theta\cos^2\theta v_y(u_{yy}-u_{xx})+(\cos\theta)v_y(\cos^2\theta-\sin^2\theta)u_{xy}\ ds + \\
& -(1-\nu)\int\limits_{\partial\Omega} v_x\cos\theta u_{xx}\sin^2\theta+v_x\cos\theta u_{yy}\cos^2\theta-2v_x\cos^2\theta\sin\theta u_{xy}\ ds + \\
& -(1-\nu)\int\limits_{\partial\Omega} v_y\sin\theta u_{xx}\sin^2\theta+v_y\sin\theta u_{yy}\cos^2\theta-2v_y\cos\theta\sin^2\theta u_{xy}\ ds = \\
& -(1-\nu)\int\limits_{\partial\Omega} u_{xx}\left[-\sin^2\theta\cos\theta v_x+\sin^2\theta\sin\theta v_y+\sin^2\theta\cos\theta v_x+\sin\theta\cos^2\theta v_y\right]\ ds + \\
& -(1-\nu)\int\limits_{\partial\Omega} u_{yy}\left[\cos\theta\cos^2\theta v_x+\cos^2\theta\sin\theta v_y+\sin^2\theta\cos\theta v_x-\sin\theta\cos^2\theta v_y\right]\ ds + \\
& -(1-\nu)\int\limits_{\partial\Omega} u_{xy}\left[-2\cos^2\theta\sin\theta v_x-2\sin^2\theta\cos\theta v_y+\sin\theta(\cos^2\theta-\sin^2\theta)v_x\right] ds + \\
& -(1-\nu)\int\limits_{\partial\Omega} -u_{xy}\left[v_y\cos\theta(\cos^2\theta-\sin^2\theta)\right]\ ds\ = \\
& -(1-\nu)\int\limits_{\partial\Omega} u_{xx}\left[\sin\theta v_y(\sin^2\theta+\cos^2\theta)\right]+u_{yy}\left[\cos\theta v_x(\cos^2\theta+\sin^2\theta)\right]\ ds + \\
& -(1-\nu)\int\limits_{\partial\Omega} u_{xy}\left[-\cos^2\theta\sin\theta v_x-\sin^2\theta\cos\theta v_y-\sin^3\theta v_x-\cos^3\theta v_y\right]\ ds = \\
& -(1-\nu)\int\limits_{\partial\Omega} \sin\theta(u_{xx}v_y-\sin^2\theta v_x u_{xy}-\cos^2 v_x u_{xy})+\cos\theta(u_{yy}v_x-\sin^2\theta u_{xy}v_y-\cos^2 u_{xy}v_y)\ ds = \\
& -(1-\nu)\int\limits_{\partial\Omega} \sin\theta(u_{xx}v_y-v_x u_{xy})+\cos\theta(u_{yy}v_x-u_{xy}v_y)\ ds = \ (*) \qquad (1.60)
\end{aligned}
$$

Substituindo em (*): $\sin\theta = -\dfrac{dx}{ds}$ e $\cos\theta = \dfrac{dy}{ds}$, e usando o Teorema de Green na primeira igualdade temos que:

$$
\begin{aligned}
(A)+(B) =& -(1-\nu)\int\limits_{\partial\Omega}\left[-(u_{xx}v_y-v_x u_{xy})\left(\frac{dx}{ds}\right)+(u_{yy}v_x-u_{xy}v_y)\left(\frac{dy}{ds}\right)\right]\ ds = \\
& -(1-\nu)\iint\limits_{\Omega}\frac{d}{dx}(u_{yy}v_x-u_{xy}v_y)+\frac{d}{dy}(u_{xx}v_y-u_{xy}v_x)\ dxdy = \\
& -(1-\nu)\iint\limits_{\Omega} u_{yyx}v_x+u_{yy}v_{xx}-u_{xxy}v_y-u_{xy}v_{yx}+u_{xxy}v_y+u_{xx}v_{yy}-u_{xy}v_y x-v_x u_{xyy}\ dxdy = \\
& -(1-\nu)\iint\limits_{\Omega} u_{yy}v_{xx}-u_{xy}v_{yx}+u_{xx}v_{yy}-u_{xy}v_{yx}\ dxdy = \\
& -(1-\nu)\iint\limits_{\Omega} u_{yy}v_{xx}-2u_{xy}v_{yx}+u_{xx}v_{yy}\ dxdy \qquad (1.61)
\end{aligned}
$$

Voltando à equação (1.55) e substituindo o desenvolvimento de (A)+(B) temos

$$\iint_\Omega \Delta u \Delta v \underbrace{-(1-\nu)\left[u_{yy}v_{xx} - 2u_{xy}v_{yx} + u_{xx}v_{yy}\right]\, dxdy}_{(A)+(B)} + \\ + \int_{\partial\Omega} vP(u)\, ds + \int_{\partial\Omega} \frac{\partial v}{\partial \vec{n}} M(u)\, ds \tag{1.62}$$

e da equação (1.52), chegamos em

$$\iint_\Omega \Delta u \Delta v - (1-\nu)\left[u_{yy}v_{xx} - 2u_{xy}v_{yx} + u_{xx}v_{yy}\right]\, dxdy + \\ + \int_{\partial\Omega} v\ P(u)\, ds + \int_{\partial\Omega} \frac{\partial v}{\partial \vec{n}} M(u)\, ds = \iint_\Omega \lambda u v\, dxdy \tag{1.63}$$

A partir desta expressão (1.63), podemos definir a forma fraca para o problema de autovalor para diferentes condições de contorno, e, levando em conta que tal expressão faz sentido se considerarmos funções u e v cujas derivadas até segunda ordem são integráveis (ao invés de considerarmos $u \in C^4(\Omega)$ e $v \in C^1(\Omega)$) utilizaremos espaços de Sobolev para isso.

1. Para
$$\begin{cases} \Delta\Delta u &= \lambda u \\ u = 0 \quad \text{e} \quad \frac{\partial u}{\partial \vec{n}} &= 0 \end{cases}$$
Temos a condição de *bordo grampeado*. Nesse caso, as duas integrais de fronteira desaparecem e a forma fraca se torna: *Encontrar* $u \in H_0^2(\Omega)$ *tal que*
$$\iint_\Omega \Delta u \Delta v - (1-\nu)\left[u_{yy}v_{xx} - 2u_{xy}v_{xy} + v_{xx}u_{yy}\right]\, dxdy = \lambda \iint_\Omega uv\, dxdy \qquad \forall v \in H_0^2(\Omega)\ . \tag{1.64}$$

2. Para
$$\begin{cases} \Delta\Delta u &= \lambda u \\ u = 0 \quad \text{e} \quad M(u) &= 0 \quad (\frac{\partial u}{\partial \vec{n}} \text{ qualquer}) \end{cases}$$
Temos a condição de contorno *bordo simplesmente suportado*. As integrais de fronteira também desaparecem e, para definir a forma fraca, é preciso considerar um espaço de Sobolev V em que as condições de contorno $u = M(u) = 0$ façam sentido. Para a condição $u = 0$ em $\partial\Omega$ podemos considerar o espaço $H^2(\Omega) \cap H_0^1(\Omega)) = \left\{u \in H^2(\Omega),\ v = 0 \text{ sobre } \partial\Omega\right\}$.
Encontrar $u \in V$ *tal que*
$$\iint_\Omega \Delta u \Delta v - (1-\nu)\left[u_{yy}v_{xx} - 2u_{xy}v_{xy} + v_{xx}u_{yy}\right]\, dxdy = \lambda \iint_\Omega uv\, dxdy \\ \forall v \in V. \tag{1.65}$$

3. Para
$$\begin{cases} \Delta\Delta u &= \lambda u \\ M(u) = 0 \quad \text{e} \quad P(u) &= 0 \end{cases}$$

Temos a condição de contorno *bordo livre*. As duas integrais de fronteira desaparecem e, se V é um espaço de Sobolev em que as condições de contorno $P(u) = M(u) = 0$ fazem sentido, obtemos a forma fraca:
Encontrar $u \in V$ tal que

$$\iint_\Omega \Delta u \Delta v - (1-\nu)\left[u_{yy}v_{xx} - 2u_{xy}v_{xy} + v_{xx}u_{yy}\right] dxdy = \lambda \iint_\Omega uv \; dxdy \qquad \forall v \in V \; . \quad (1.66)$$

Aqui vale a pena retomar a discussão iniciada após a equação (1.52). Agora a forma fraca se traduz na equação integral

$$\iint_\Omega \Delta u \Delta v - (1-\nu)\left[u_{yy}v_{xx} - 2u_{xy}v_{xy} + v_{xx}u_{yy}\right] dxdy = \lambda \iint_\Omega uv \; dxdy \; , \quad (1.67)$$

que deve ser usada no problema de *fronteira livre*, mas também pode ser usada no problema de *bordo grampeado*.

Então, para o bordo grampeado quaisquer das formulações (1.53) e (1.67) são igualmente válidas.

Sendo assim, no que tange às soluções de (1.67) – autofunções e autovalores, o valor de ν é irrelevante, já que não aparece em (1.53). Ou seja, na condição de bordo grampeado o coeficiente de Poisson não influencia nos autovalores e nas autofunções.

Por outro lado, não se pode esquecer que o coeficiente de Poisson também está associado à frequência de vibração. Vimos na Seção (1.3) que

$$\omega = \sqrt{\frac{\lambda D}{\rho}} \; ,$$

em que $D = \dfrac{Eh^3}{12(1-\nu^2)}$. Quando ν tende a 1 a frequência tende a infinito. No entanto, estando a parte temporal separada da espacial, pode-se muito bem considerar o coeficiente de Poisson correto (i.e. adequado ao tipo de material) para a frequência de vibração, ao mesmo tempo escolhendo um valor arbitrário, digamos $\tilde{\nu}$ para ser usado na equação (1.67).

Este assunto será retomado na Subseção (4.4) quando testaremos a dependência em ν das soluções espaciais, do ponto de vista do resultado numérico.

1.4.3 Existência de solução, caracterização do espectro e autovalores

Vamos continuar nosso estudo teórico do problema de autovalor (1.45). O objetivo desta subseção é mostrar que a solução fraca para esse problema existe e apresentar uma maneira de buscar (numericamente) essa solução.

Considere o problema de autovalor (1.45)

$$\begin{cases} \Delta\Delta u &= \lambda u \\ u = 0 \quad \text{e} \quad & \frac{\partial u}{\partial \vec{n}} = 0 \end{cases}$$

em que $u \in C^4(\Omega) \cap C^1(\overline{\Omega})$ e $\lambda \in \mathbb{R}^*_+$, cuja formulação fraca é encontrar $u \in H^2_0(\Omega)$ e $\lambda \in \mathbb{R}^*_+$ tais

que

$$\iint_\Omega \Delta u \Delta v - (1-\nu)\left[u_{yy}v_{xx} - 2u_{xy}v_{xy} + v_{xx}u_{yy}\right]\, dxdy = \lambda \iint_\Omega uv\, dxdy \qquad \forall v \in H_0^2(\Omega). \quad (1.68)$$

com $0 \leq \nu < 1$.

Vamos mostrar, com base na literatura, que existe uma base ortonormal enumerável $\{u_i\}_{i\in\mathbb{N}}$ de $H_0^2(\Omega)$ e uma sequência $\{\lambda_i\}_{i\in\mathbb{N}} \subset \mathbb{R}_+^*$ tal que $\lambda_i \to \infty$, tais que para cada $i \in \mathbb{N}$, o par (u_i, λ_i) satisfaz (1.68). Essa é uma forma de se responder ao problema de autovalor formulado em (1.45). Vamos reduzir o estudo desse problema variacional ao estudo do espectro de um operador compacto e auto-ajunto em um espaço de Hilbert adequado.

Para isso, precisaremos de mais alguns conceitos e resultados importantes que apresentaremos a seguir. Nos baseamos nas referências [1], [7] [11], que podem ser consultadas para um estudo mais profundo.

Lema 1.4.1. *Quando o conjunto Ω é limitado, a seminorma $|\cdot|_{H_0^l(\Omega)}$ é uma norma em $H_0^l(\Omega)$, equivalente [2] à noma $||\cdot||_{H_0^l(\Omega)}$.*

A demonstração pode ser consultada em [1].

Lema 1.4.2. *Para toda função $v \in H_0^2(\Omega)$, $||\Delta v||_{L_2(\Omega)} = |v|_{H_0^2(\Omega)}$. Isso implica que, sobre $H_0^2(\Omega)$, a semi-norma $v \mapsto ||\Delta v||_{L_2(\Omega)}$ é uma norma equivalente a norma $||\cdot||_{H_0^2(\Omega)}$.*

Demonstração:

$$\left(||\Delta v||_{L_2(\Omega)}\right)^2 = \iint_\Omega |\Delta u|^2\, dxdy = \iint_\Omega (u_{xx}+u_{yy})^2\, dxdy = \iint_\Omega u_{xx}^2 + u_{yy}^2 + 2u_{xx}u_{yy}\, dxdy$$

e

$$\left(|v|_{H_0^2(\Omega)}\right)^2 = \iint_\Omega \sum_{|\alpha|=2} |\partial^\alpha u|^2\, dxdy = \iint_\Omega u_{xx}^2 + u_{yy}^2 + 2u_{xy}^2\, dxdy\ .$$

Para mostrar a igualdade entre as normas, basta mostrar que $\iint_\Omega u_{xx}u_{yy}\, dxdy = \iint_\Omega u_{xy}^2\, dxdy$. Isso é feito usando a fórmula de integral por partes (1.48) duas vezes:

$$\begin{aligned}
\iint_\Omega u_{xy}u_{xy}\, dxdy &= \int_{\partial\Omega} u_x u_{xy}\, dxdy - \int_\Omega u_x u_{xyy}\, dxdy = \\
&= \int_{\partial\Omega} u_x u_{xy}\, dxdy - \left[\int_{\partial\Omega} u_x u_{yy}\, dxdy - \int_\Omega u_{xx}u_{yy}\, dxdy\right] = \\
&\underbrace{\int_{\partial\Omega} u_x u_{xy}\, dxdy - \int_{\partial\Omega} u_x u_{yy}\, dxdy}_{=0,\ \text{pois } u\in H_0^2(\Omega)} + \int_\Omega u_{xx}u_{yy}\, dxdy = \\
&\int_\Omega u_{xx}u_{yy}\, dxdy
\end{aligned} \quad (1.69)$$

Logo, $||\Delta v||_{L_2(\Omega)} = |v|_{H_0^2(\Omega)}$, como queríamos. □

[2] Duas normas $||\cdot||_1$ e $||\cdot||_2$, num mesmo espaço normado X, são equivalentes se existirem constantes reais e positivas β_1, β_2 tais que $\beta_1||u||_1 \leq ||u||_2 \leq \beta_2||u||_1$, $\forall\ x \in X$.

Definição 1.4.3 (Coercividade)**.** *Seja V um espaço vetorial normado sobre $\mathbb{R}$. Uma forma bilinear $a : V \times V \to \mathbb{R}$ é dita* coerciva (ou V-elíptica) *se existe uma constante $\alpha > 0$ tal que*

$$a(u,u) \geq \alpha ||u||^2 \quad \forall\ u \in V\ .$$

Teorema 1.4.1 (Lax-Milgram)**.** *Sejam V um espaço de Hilbert, $f : V \longrightarrow \mathbb{R}$ um funcional linear e $a : V \times V \longrightarrow \mathbb{R}$ uma forma bilinear contínua e coerciva. Então o problema variacional abstrato*

$$a(u,v) = f(v)\ , \qquad \forall\ v \in V$$

possui uma única solução $u \in V$.

Seja $a(u,v)$ o lado esquerdo da equação (1.68):

$$a(u,v) = \iint\limits_{\Omega} \Delta u \Delta v - (1-\nu)\left[u_{yy}v_{xx} - 2u_{xy}v_{xy} + v_{xx}u_{yy}\right]\ dxdy\ .$$

Teorema 1.4.2. *A função $a : H_0^2(\Omega) \times H_0^2(\Omega) \longrightarrow \mathbb{R}$ definida pela expressão acima é uma forma bilinear, simétrica, contínua, coerciva e definida positiva.*

Dem.: A simetria e a bilinearidade de $a(u,v)$ são imediatas. Continuidade:

$$\begin{aligned}
|a(u,v)| &= \left| \iint\limits_{\Omega} \Delta u \Delta v\ dxdy - (1-\nu) \iint\limits_{\Omega} \left[u_{yy}v_{xx} - 2u_{xy}v_{xy} + v_{xx}u_{yy}\right]\ dxdy \right| \leq \\
&\leq \left| \iint\limits_{\Omega} \Delta u \Delta v\ dxdy \right| + |(1-\nu)| \left| \iint\limits_{\Omega} \left[u_{yy}v_{xx} - 2u_{xy}v_{xy} + v_{xx}u_{yy}\right]\ dxdy \right| \leq \\
&\leq \left| \iint\limits_{\Omega} \Delta u \Delta v\ dxdy \right| + \left\{ \left| \iint\limits_{\Omega} u_{yy}v_{xx}\ dxdy \right| + \left| \iint\limits_{\Omega} 2u_{xy}v_{xy}\ dxdy \right| + \left| \iint\limits_{\Omega} v_{xx}u_{yy}\ dxdy \right| \right\}\ ,
\end{aligned} \tag{1.70}$$

onde usamos a desigualdade triangular e o fato de $0 \leq \nu < 1$. Aplicando a desigualdade de Hölder[3] em cada parcela, utilizando o Lema 1.4.2 e considerando que $||D^\alpha u||_{L_2} \leq |u|_{H_0^2(\Omega)}$, $\forall\ \alpha$ tal que $|\alpha| = 2$ obtemos

$$\begin{aligned}
|a(u,v)| &\leq ||\Delta u||_{L_2(\Omega)} ||\Delta v||_{L_2(\Omega)} + ||u_{xx}||_{L_2(\Omega)} ||v_{yy}||_{L_2(\Omega)} + ||u_{yy}||_{L_2(\Omega)} ||v_{xx}||_{L_2(\Omega)} + 2||u_{xy}||_{L_2(\Omega)} ||v_{xy}||_{L_2(\Omega)} \leq \\
&\leq |u|_{H_0^2(\Omega)} |v|_{H_0^2(\Omega)} + |u|_{H_0^2(\Omega)} |v|_{H_0^2(\Omega)} + |u|_{H_0^2(\Omega)} |v|_{H_0^2(\Omega)} + 2|u|_{H_0^2(\Omega)} |v|_{H_0^2(\Omega)} = 5|u|_{H_0^2(\Omega)} |v|_{H_0^2(\Omega)}\ .
\end{aligned} \tag{1.71}$$

A equivalência entre a norma e a seminorma de $H_0^2(\Omega)$, (lema 1.4.1) nos permite concluir que existe uma constante $M > 0$ tal que

$$|a(u,v)| \leq M ||u||_{H_0^2} ||v||_{H_0^2}$$

e, portando, a forma bilinear $a(u,v)$ é contínua.

[3] A desigualdade de Hölder afirma que $\left| \int\limits_{\Omega} f(x)g(x)\ dx \right| \leq \left(\int\limits_{\Omega} |f(x)|^p \right)^{1/p} \left(\int\limits_{\Omega} |g(x)|^q \right)^{1/q}$ em que $f, g : \Omega \subset \mathbb{R}^n \longrightarrow \mathbb{R}$, $f \in L^p(\Omega)$ e $g \in L^q(\Omega)$. Aqui, a desigualdade foi usada para $p = q = 2$.

Coercividade: $a(u,u) = \iint_\Omega (\Delta u)^2 \; dxdy - (1-\nu) \iint_\Omega \left[2u_{yy}u_{xx} - 2u_{xy}^2\right] \; dxdy$ pode ser escrita da forma $a(u,u) = \iint_\Omega \nu(\Delta u)^2 + (1-\nu)u_{xx}^2 + u_{yy}^2 + 2u_{xy}^2 \; dxdy$. A partir daí temos

$$a(u,u) = \iint_\Omega \nu(\Delta u)^2 + (1-\nu)u_{xx}^2 + u_{yy}^2 + 2u_{xy}^2 \; dxdy =$$

$$\nu \iint_\Omega |\Delta u|^2 \; dxdy + (1-\nu) \iint_\Omega |u_{xx}|^2 + |u_{yy}|^2 + 2|u_{xy}|^2 \; dxdy =$$

$$\nu(||\Delta u||_{L_2(\Omega)})^2 + (1-\nu)|u|_{H_0^2}^2 = \nu(|u|_{H_0^2})^2 + (1-\nu)|u|_{H_0^2}^2 = |u|_{H_0^2}^2 \geq \alpha||u||_{H_0^2}^2 \text{ , com } \alpha > 0 \; . \quad (1.72)$$

em que na penúltima igualdade utilizamos o Lema 1.4.2 e a constante $\alpha > 0$ vem da equivalência entre a norma e a seminorma de $H_0^2(\Omega)$ (Lema 1.4.1).

A coercividade de garante que $a(u,v)$ é definida positiva. □

Agora, considere a função

$$\begin{aligned} \Lambda : \; H_0^2(\Omega) &\longrightarrow (H_0^2(\Omega))' \\ u &\longmapsto \Lambda[u] \end{aligned} \quad (1.73)$$

que mapeia o espaço de Hilbert $H_0^2(\Omega)$ no seu dual. Para cada $u \in H_0^2(\Omega)$, o funcional linear

$$\begin{aligned} \Lambda[u] : H_0^2(\Omega) &\longrightarrow \mathbb{R} \\ v &\longmapsto \Lambda[u](v) \end{aligned} \quad (1.74)$$

é definido por $\Lambda[u](v) = a(u,v)$, para $v \in H_0^2(\Omega)$.

Lema 1.4.3. $\Lambda : H_0^2(\Omega) \longrightarrow (H_0^2(\Omega))'$ *é um homeomorfismo linear entre* $H_0^2(\Omega)$ *e seu dual.*

Demonstração: A bilinearidade, a continuidade e a coercividade de $a(u,v)$, estabelecidas pelo Teorema 1.4.2, colocam a forma bilinear $a(u,v)$ nas condições do Teorema de Lax-Milgram (1.4.1) e, portanto, para cada funcional $f \in (H_0^2(\Omega))'$ existe uma única $u \in H_0^2(\Omega)$ tal que $a(u,v) = f(v) \; \forall \; v \in H_0^2(\Omega)$. Assim, temos que $\Lambda[u](v) = a(u,v) = f(v) \; \forall v \in H_0^2(\Omega)$, de onde segue a identidade de $\Lambda[u] = f$, que nos mostra que Λ é uma bijeção, e Λ é contínua pois a é contínua.

Além disso, a inversa Λ^{-1} também é contínua: dada uma $f \in (H_0^2(\Omega))'$, arbitrária, considere $u \in H_0^2(\Omega)$, tal que $\Lambda[u] = f$. Do Teorema de Lax-Milgram temos que $a(u,v) = f(v)$, $\forall v \in H_0^2(\Omega)$, então, em particular, para $v = u$ tem-se que $a(u,u) = f(u)$. Da coercividade de $a(u,v)$ e da continuidade de f segue que

$$\alpha||u||_{H_0^2}^2 \leq a(u,u) = f(u) \leq ||f||^*||u||_{H_0^2}$$

em que $||\cdot||^*$ denota a norma em $(H_0^2(\Omega))'$. Logo,

$$||\Lambda^{-1}(f)||_{H_0^2(\Omega)} = ||u||_{H_0^2(\Omega)} \leq \frac{1}{\alpha}||f||^* \; , \; \forall f \in (H_0^2(\Omega))' \; . \; \square$$

Considere a inclusão canônica $i : H_0^2(\Omega) \hookrightarrow L_2(\Omega)$. Pela definição do espaço $H_0^2(\Omega)$, é claro que $H_0^2(\Omega) \subset L_2(\Omega)$. Além disso, tal inclusão é um mergulho compacto. A compacidade dessa inclusão é

uma consequência do Teorema de Rellich-Kondrachov[4]. Assim, o espaço $H_0^2(\Omega)$ está compactamente mergulhado em $L_2(\Omega)$, o que significa que existe uma constante $K > 0$ tal que

$$||u||_{L_2(\Omega)} \leq K||u||_{H_0^2(\Omega)} \ .$$

Considere também

$$\begin{aligned} J : L_2(\Omega) &\longrightarrow (H_0^2(\Omega))' \\ \phi &\mapsto J[\phi] \end{aligned} \tag{1.75}$$

em que

$$\begin{aligned} J[\phi] : H_0^2(\Omega) &\longrightarrow \mathbb{R} \\ v &\mapsto J[\phi](v) = \iint_\Omega \phi v \ dxdy \ . \end{aligned} \tag{1.76}$$

J é linear, contínua e injetora. A linearidade é imediata. A continuidade vem do fato que

$$\begin{aligned} |J[\phi](v)| = \left| \iint_\Omega \phi v \ dxdy \right| \leq \left(\iint_\Omega |\phi|^2 \ dxdy \right)^{1/2} \left(\iint_\Omega |v|^2 \ dxdy \right)^{1/2} = \\ ||\phi||_{L_2(\Omega)} ||v||_{L_2(\Omega)} \leq K||\phi||_{L_2(\Omega)} ||v||_{H_0^2(\Omega)} \ , \text{ com } K > 0. \end{aligned} \tag{1.77}$$

Então,

$$||J[\phi] \ ||^* = \sup_{v \neq 0} \ \frac{|J[\phi](v)|}{||v||_{H_0^2}} \leq K||\phi||_{L_2(\Omega)} \ .$$

Além disso, se $J[\phi] \equiv 0$, então $J[\phi](v) = \iint_\Omega \phi v \ dxdy = 0 \ , \ \forall v \in H_0^2(\Omega)$. Logo, $\phi \equiv 0$. Assim, $KerJ = \{0\}$ e, portanto, J é injetora.

Considere agora a função $T : H_0^2(\Omega) \longrightarrow H_0^2(\Omega)$ dada pela composição a seguir:

$$H_0^2(\Omega) \xrightarrow{\ i\ } L_2(\Omega) \xrightarrow{\ J\ } H_0^2(\Omega)' \xrightarrow{\ \Lambda^{-1}\ } H_0^2(\Omega) \ \ .$$
$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{T = \Lambda^{-1} \circ J \circ i}$$

Como $a(u, v)$ é forma bilinear, simétrica e definida positiva (1.4.2), então $a(u, v)$ é um produto interno em $H_0^2(\Omega)$. Além disso, por causa da continuidade e coercividade de $a(u, v)$, a norma induzida por este produto interno é uma norma equivalente à norma $|| \cdot ||_{H_0^2(\Omega)}$ em $H_0^2(\Omega)$. Chamaremos de $\mathscr{H}_0^2(\Omega)$ o espaço de Hilbert $H_0^2(\Omega)$ equipado com o produto interno $\langle \cdot, \cdot \rangle_{\mathscr{H}_0^2} \equiv a(\cdot, \cdot)$.

É com o operador linear $T : \mathscr{H}_0^2(\Omega) \longrightarrow \mathscr{H}_0^2(\Omega)$, sobre o espaço de Hilbert $\mathscr{H}_0^2(\Omega)$ que trabalharemos para caracterizar as soluções do nosso problema (1.68).

Teorema 1.4.3. *T é um operador compacto e autoadjunto em $\mathscr{H}_0^2(\Omega)$. Além disso, seus autovalores correspondem aos inversos dos autovalores do problema (1.45).*

Antes da demonstração desse teorema, vale lembrar que, na Seção 1.3, estabelecemos que os

[4] O Teorema Rellich-Kondrachov estabelece inclusões entre espaços de Sobolev e espaços L_p. Em particular, estabelece que a inclusão $H^1(\Omega) \subset L_2(\Omega)$ é compacta. Como, por definição, $H^2(\Omega) \subset H^1(\Omega)$, concluímos que a inclusão $H^2(\Omega) \subset L_2(\Omega)$ é compacta. O enunciado completo pode ser consultado em [1] e [3].

autovalores do problema $\Delta\Delta u = \lambda u$ são estritamente positivos. Logo, quando olhamos para a formulação fraca desse problema, consideramos $\lambda > 0$ e, portanto, faz sentido falarmos nos inversos λ^{-1}.

Demonstração: A compacidade de T segue da compacidade de i e da continuidade J e Λ^{-1}. Ainda,

$$\begin{aligned}\langle Tu, v\rangle_{\mathscr{H}_0^2} &= \langle \Lambda^{-1}\circ J\circ i(u), v\rangle_{\mathscr{H}_0^2} = a(\Lambda^{-1}\circ J\circ i(u), v) = \Lambda[\Lambda^{-1}\circ J\circ i(u)](v) = \\ &= J[i(u)](v) = \iint\limits_{\Omega} i(u)v\ dxdy = \iint\limits_{\Omega} i(v)u\ dxdy = J[i(v)](u) = \\ &= \Lambda[\Lambda^{-1}\circ J\circ i(v)](u) = a(\Lambda^{-1}\circ J\circ i(v), u) = \langle Tv, u\rangle_{\mathscr{H}_0^2} = \langle u, Tv\rangle_{\mathscr{H}_0^2}\ ,\end{aligned}$$

que mostra que T é autoadjunto.

O problema de autovalor

$$Tu = \lambda^{-1}u$$

equivale a

$$\lambda(\Lambda^{-1}\circ J\circ i)u = u$$

e, aplicando a função Λ de ambos os lados, obtemos funcionais lineares

$$\lambda J[i(u)] \equiv \Lambda[u]\ ,$$

que são tais que

$$\lambda J[i(u)](v) = \Lambda[u](v) = a(u, v)\ ,\quad \forall v \in H_0^2(\Omega)\ .$$

A expressão acima equivale ao problema (1.68), o que nos mostra que o problema

$$Tu = \lambda^{-1}u\ ,\quad \lambda \in \mathbb{R}_+^*\ \text{e}\ u \in \mathscr{H}_0^2(\Omega)$$

é equivalente ao problema (1.68). □

A Teoria Espectral para operadores compactos e autoadjuntos nos garante que os autovalores de T são reais, formam uma sequência decrescente $\left\{\dfrac{1}{\lambda_i}\right\}_{i\in\mathbb{N}} \subset \mathbb{R}_+^*$ tal que $\dfrac{1}{\lambda_i} \to 0$ e que os autovetores associados aos autovalores formam uma base ortonormal enumerável (uma base de Hilbert) $\{u_i\}_{i\in\mathbb{N}}$ para o subespaço complemento ortogonal de Ker$T$. Como J é injetora, então T também é injetora, Ker$T = \{0\}$ e, portanto, o complemento ortogonal de Ker$T$ é $\mathscr{H}_0^2(\Omega)$.

Assim, concluímos que os autovalores do problema (1.68) são reais, estritamente positivos, formam uma sequência crescente $\{\lambda_i\}$, tal que $\lambda_i \to \infty$ e os autovetores formam uma base enumerável de $H_0^2(\Omega)$. Assim, caracterizamos os autovalores e autofunções para o problema (1.68).

Uma maneira de buscar a solução é observar que

Proposição 1.4.1. *Seja V um espaço de Hilbert e $a : V \times V \longrightarrow \mathbb{R}$ uma forma bilinear simétrica. Então u satisfaz a expressão*

$$a(u, v) = \lambda \iint_{\Omega} uv\ dxdy\ ,\qquad \forall v \in V$$

se, e somente se, u é ponto crítico de $Q(u) = a(u,u) - \lambda \iint\limits_{\Omega} u^2 \; dxdy$.

Demonstração: u é ponto crítico de $Q(u)$ se, e somente se

$$\lim_{h\to 0} \frac{Q(u+hv) - Q(u)}{h} = 0 \; , \forall v \in V \; .$$

$$\begin{aligned}
\frac{Q(u+hv) - Q(u)}{h} &= \frac{a(u+hv,u) - \lambda \iint_\Omega (u+hv)^2 \; dxdy - \left[a(u,u) - \lambda \iint_\Omega u^2 \; dxdy\right]}{h} = \\
&= \frac{2h\left[a(u,v) - \lambda \iint_\Omega uv \; dxdy\right] + h^2\left[a(v,v) - \lambda \iint_\Omega v^2 \; dxdy\right]}{h} = \\
&= 2\left[a(u,v) - \lambda \iint_\Omega uv \; dxdy\right] + h\left[a(v,v) - \lambda \iint_\Omega v^2 \; dxdy\right]. \qquad (1.78)
\end{aligned}$$

Assim,

$$\begin{aligned}
&\lim_{h\to 0} \frac{Q(u+hv) - Q(u)}{h} = 0 \quad \forall \; v \in V \quad \text{ se, e somente se} \\
&\lim_{h\to 0} \; 2\left[a(u,v) - \lambda \iint_\Omega uv \; dxdy\right] + h\left[a(v,v) - \lambda \iint_\Omega v^2 \; dxdy\right] = 0 \; \forall \; v \in V \quad \text{ se, e somente se} \\
&a(u,v) - \lambda \iint_\Omega uv \; dxdy = 0 \quad \forall \; v \in V \; . \; \square
\end{aligned} \qquad (1.79)$$

Assim, para buscar numericamente a solução para nosso problema, buscaremos pontos críticos do funcional $Q(u)$ descrito acima. A implementação numérica baseada nessa proposição será descrita no próximo capítulo.

Capítulo 2

Implementação

O objetivo deste capítulo é apresentar a implementação numérica que utilizamos para resolver o nosso problema. O Método escolhido foi o Método de Elementos Finitos (MEF).

Nosso objetivo neste trabalho foi implementar um software capaz de fornecer dados para a geração de imagens das linhas nodais, mas para o caso da vibração de uma placa fina de metal. Utilizamos o mesmo método (o MEF), e partimos do modelo de vibração de uma placa fina de metal, que consiste numa EDP de ordem 4, e cuja formulação foi descrita no Capítulo 1. Apresentamos também dois tipos de formulação (conforme e não conforme) para o mesmo problema, e as implementações feitas com base nesses tipos.

2.1 O problema discreto

Como vimos no Capítulo 1, estamos interessados em resolver, numericamente, o problema de autovalor

$$\Delta\Delta u + \lambda u = 0 \quad \text{em} \quad \Omega \subset \mathbb{R}^2 \ , \tag{2.1}$$

com a condição de contorno de *bordo grampeado* em que

$$u(x,y) = 0 \ \text{ e } \ \frac{\partial u}{\partial \vec{n}} = 0 \ \text{ em } \ \partial\Omega \ . \tag{2.2}$$

O método de elementos finitos está baseado na formulação fraca (ou variacional) do problema. Para implementá-lo, é preciso transformar a forma quadrática $Q(u) = a(u,u) - \lambda \iint_\Omega u^2 dxdy$, baseada na integral variacional (1.64), num problema discreto e, em seguida, à luz do que foi feito no Capítulo 1, encontrar pontos críticos não triviais da forma quadrática

$$\mathscr{I} = Q(u) = \iint_\Omega (\Delta u)^2 - (1-\nu)\left[u_{yy}u_{xx} - 2{u_{xy}}^2 + u_{xx}u_{yy}\right] - \lambda u^2 \ dxdy. \tag{2.3}$$

2.1.1 Discretização do domínio: espaços de elementos finitos, elementos conformes e não conformes

Para resolver, numericamente o problema de autovalor, começamos com a discretização do domínio $\Omega \subset \mathbb{R}^2$ onde o problema será resolvido. Nesta etapa, o domínio é dividido em regiões poligonais fechadas, não vazias, chamadas de *elementos*. O conjunto dessas regiões e, portanto, o domínio discretizado, é chamado de *malha*.

Nosso domínio Ω será sempre fechado, limitado e poligonal. O tipo de *elemento* escolhido para a discretização foi o triangular. Com isso podemos chamar nossa discretização do domínio Ω de *triangulação*. Cada elemento triangular da malha será chamado de T_i. A triangularização tem as seguintes características:

(i) Não contém buracos ou superposições. Os elementos se intersectam apenas nas arestas dos triângulos.

(ii) A união de todos os elementos aproxima o domínio Ω: $\bigcup_{i=1}^{N} T_i = \Omega$. (Se o domínio não fosse exatamente um polígono, então teríamos $\bigcup_{i=1}^{N} T_i \subseteq \Omega$).

Em cada elemento triangular T_i, a função $u(x, y)$ será aproximada por uma função polinomial $u_i(x, y)$, que se anula fora de T_i. Assim, $u(x, y) \approx \sum_{i=1}^{N} u_i(x, y)$, onde N é o número total de elementos triangulares da malha.

Os pontos de T_i onde a função aproximadora u_i (e, possivelmente, suas derivadas) será calculada são chamados de *nós*. Cada elemento finito pode ser caracterizado pela quantidade, localização de seus nós e pelo tipo de função (valor da função u_i ou de suas derivadas) calculada neles. As funções escolhidas para terem seus valores calculados nos nós da malha são chamadas de *variáveis nodais*. Os parâmetros da função u_i são unicamente determinados pelas variáveis nodais.

Ao definir a discretização de Ω em elementos T_i, e os polinômios u_i para cada T_i, estamos definido um espaço de dimensão finita onde buscaremos a solução discreta. Esse espaço contém funções polinomiais por partes, $p(x, y)$, definidas em todo o domínio (poligonal e fechado) Ω. Sua restrição a cada elemento T_i nos fornece o polinômio aproximador u_i, isto é, $u_i = p|_{T_i}$. Este espaço será chamado de *espaço de elementos finitos* e, aqui, será denotado por E. E o espaço dos polinômios u_i, para cada T_i, será chamado de P_{T_i}: $P_{T_i} = \{u_i = p|_{T_i}; p \in E\}$.

Uma característica importante do espaço de elementos finitos é que sob certas condições sobre os polinômios p e u_i, E se torna um subespaço do espaço de funções onde buscamos a solução e, que no nosso caso é o $H_0^2(\Omega)$. Quando isso acontece, utilizamos a terminologia *elemento finito conforme* para caracterizar a aproximação e o elemento finito utilizados, caso contrário, dizemos que a aproximação é *não conforme*.

A seguir, apresentaremos um teorema que apresenta condições para que o espaço de elementos finitos E seja um subespaço do espaço de funções em que estamos trabalhando: $H_0^2(\Omega)$, e portanto, para que a aproximação seja considerada conforme. Esse resultado foi baseado na referência [1].

Teorema 2.1.1. *Considere E um espaço de elementos finitos como descrito acima. Se $E \subset C^1(\bar{\Omega})$ e $P_{T_i} \subset H^2(T_i)$, $\forall i$, $1 \leq i \leq N$, então as inclusões a seguir são satisfeitas:*

$$\begin{aligned} &E \subset H^2(\Omega) \\ &E_0 = \{p \in E : p = 0 \text{ sobre } \partial\Omega\} \subset H^2(\Omega) \cap H_0^1(\Omega) \quad e \\ &E_{00} = \{p \in E : p = \partial_n p = 0 \text{ sobre } \partial\Omega\} \subset H_0^2(\Omega) \ . \end{aligned} \tag{2.4}$$

Demonstração: Vamos mostrar a primeira inclusão. Seja $u \in E$. Como $u|_{T_i}$ é um polinômio, em cada T_i, $u|_{T_i}$ é L_2 integrável, i.e., $\iint\limits_{T_i} (u|_{T_i})^2 \, dxdy < \infty$. Assim, como a integral de u^2 em Ω consiste de uma soma finita de integrais de $(u|_{T_i})^2$ em cada T_i e, como ∂T_i tem medida nula, $\iint\limits_{\Omega} u^2 \, dxdy < \infty$ e portanto, $u \in L_2(\Omega)$.

Para mostrar que $u \in H_0^2(\Omega)$, precisamos mostrar que todas as derivadas (fracas) de ordem α, com $|\alpha| \leq 2$, existem e pertencem a $L_2(\Omega)$. Vamos começar com as derivadas de primeira ordem ($|\alpha| = 1$), e mostrar que, para cada $k = 1, 2$, existem funções $\partial_k u$ que satisfazem a definição de derivada fraca (1.50)

$$\iint_\Omega (\partial_k u)v \; dxdy = -\iint_\Omega u(\partial_k v) \; dxdy \; , \quad \forall \; v \in C_0^\infty(\Omega) \; . \tag{2.5}$$

Para cada $k = 1, 2$, escolhemos como candidato à $\partial_k u$, a função θ, que é a derivada da restrição de u no interior de cada elemento T_i: $\theta|_{T_i} = \partial_k \left(u|_{T_i}\right)$. Assim como argumentamos para u, também θ está em $L_2(\Omega)$. Aplicando a fórmula da integral por partes (1.48) em T_i obtemos

$$\iint_{T_i} (\theta|_{T_i})v \; dxdy = \iint_{T_i} \partial_k(u|_{T_i})v \; dxdy = -\iint_{T_i} u\partial_k v \; dxdy + \int_{\partial T_i} uvn_k \; ds \; , \quad \forall v \in C_0^\infty(\Omega)$$

em que n_k é a k-ésima componente do vetor normal $\vec{n}$ ao longo da fronteira ∂T_i. Somando em todos os N elementos T_i, obtemos

$$\iint_\Omega \theta v \; dxdy = \sum_{i=1}^N \left(\iint_{T_i} (\theta|_{T_i})v \; dxdy \right) = -\sum_{i=1}^N \left(\iint_{T_i} u\partial_k v \; dxdy \right) + \sum_{i=1}^N \left(\int_{\partial T_i} uvn_k \; ds \right)$$

e, portanto,

$$\iint_\Omega \theta v \; dxdy = -\iint_\Omega u\partial_k v \; dxdy + \sum_{i=1}^N \left(\int_{\partial T_i} uvn_k \; ds \right) \; .$$

Na última parcela, cada aresta da triangulação que não está contida em $\partial\Omega$ aparece duas vezes na soma, com integrandos de mesmo valor absoluto mas com sinais trocados, por causa da inversão no sentido da normal em triângulos adjacentes. O valor absoluto é o mesmo porque u e v são funções contínuas. Com isso, sobram apenas as arestas em $\partial\Omega$ e, portanto,

$$\sum_{i=1}^N \int_{\partial T_i} uvn_k ds = \int_{\partial\Omega} uvn_k ds \; ,$$

que, por sua vez, é uma integral nula, já que $v \in C_0^\infty(\Omega)$.

Portanto

$$\iint_\Omega u(\partial_k v) \; dxdy = -\iint_\Omega \theta v \; dxdy \; , \quad \forall v \in C_0^\infty(\Omega) \; , \forall k = 1, 2 \; ,$$

isto é, θ é a derivada fraca procurada $\partial_k u$. Isto mostra que $u \in H^1(\Omega)$.

Agora, precisamos mostrar que as derivadas de segunda ordem ($|\alpha| = 2$) existem e estão em $L_2(\Omega)$. Para isso, para cada derivada de primeira ordem u_k, vamos mostrar que existe, para cada $j = 1, 2$, uma função (u_{kj}) que satisfaz a condição (2.5). A candidata natural será a função $u_{kj} = \partial_{kj}(u|_{T_i})$. Novamente, podemos utilizar a fórmula de integração por partes em T_i e obtemos, para

cada $j = 1, 2$:

$$\iint\limits_{T_i} u_{kj} v \; dxdy = \iint\limits_{T_i} (\partial_{kj}(u|_{T_i}) v \; dxdy = -\iint\limits_{T_i} (\partial_k u)(\partial_j v) \; dxdy + \int\limits_{\partial_{T_i}} u_k v n_j \; ds \; .$$

Aplicamos de novo a integração por partes no primeiro termo, de onde segue

$$\iint\limits_{T_i} u_{kj} v \; dxdy = \iint\limits_{T_i} u \partial_{jk} v \; dxdy - \int\limits_{\partial T_i} u(\partial_j v) n_k \; ds + \int\limits_{\partial T_i} (\partial_k u) v n_j \; ds \; .$$

Como u e $\partial_j v$ são contínuas,

$$\sum_{i=1}^{N} \int\limits_{\partial T_i} u \partial_j v n_k \; ds = \int\limits_{\partial \Omega} u \partial_j v n_k \; ds$$

e, como $\partial_k u$ e v são contínuas,

$$\sum_{i=1}^{N} \int\limits_{\partial T_i} (\partial_k u) v n_j \; ds = \int\limits_{\partial \Omega} (\partial_k u) v n_j \; ds \; .$$

Ambos os termos, porém são nulos, porque $v \in C_0^\infty(\Omega)$ implica que $v = 0$ e $\partial_j v = 0$ em $\partial\Omega$. Logo

$$\iint\limits_{\Omega} u \partial_{kj} v \; dxdy = \iint\limits_{\Omega} u \partial_{jk} v \; dxdy = \sum_{i=1}^{N} \iint\limits_{T_i} u \partial_{jk} v \; dxdy = \sum_{i=1}^{N} \iint\limits_{T_i} u_{kj} v \; dxdy = (-1)^2 \iint\limits_{\Omega} u_{kj} v \; dxdy \; .$$

Isto prova a primeira inclusão. As outras inclusões seguem da caracterização dos espaços $H_0^1(\Omega)$ e $H_0^2(\Omega)$. $\square$

Assim, para que a aproximação seja considerada *conforme*, é necessário garantir que todas as derivadas do polinômio local u_i, de ordem menor que 2, sejam contínuas na fronteira entre os elementos T_i, para que o espaço de elementos finitos E esteja contido em $C^1(\bar{\Omega})$. Essa condição será usada nas próximas seções para caracterizarmos os elementos finitos escolhidos segundo esse critério e com a seguinte caracterização:

Considere dois elementos adjacentes T_1 e T_2 e a aresta comum a eles. A parametrização da aresta pelo parâmetro s, é feita com base nos pontos inicial (x_I, y_I) e final (x_F, y_F) da aresta:

$$x = x_I + (x_F - x_I)s \quad ; \quad y = y_I + (y_F - y_I)s$$

com $0 \leq s \leq 1$, como indica a Figura 2.1. Um polinômio aproximador u_i de grau m é, ao longo da aresta, um polinômio de grau m na variável s:

$$u_i(x(s), y(s)) = u_i(s) = e_1 + e_2 s + ... + e_{m+1} s^m$$

e é unicamente determinado por $m+1$ parâmetros. Garantir a continuidade de u_i é garantir que seus $m + 1$ parâmetros podem ser unicamente determinados pelas variáveis nodais presentes na aresta

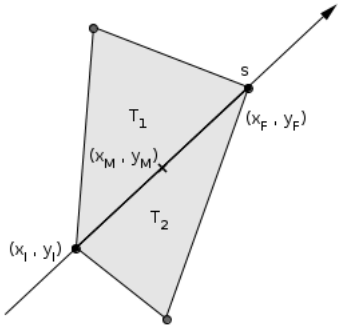


Figura 2.1: *Aresta comum aos elementos T_1 e T_2. Os índices I, F e M indicam os pontos inicial, final e médio da aresta, respectivamente.*

comum aos elementos. Também é preciso garantir a continuidade da derivada normal à aresta $\dfrac{\partial u_i}{\partial \vec{n}}$, o que pode ser feito pelo mesmo critério, já que, ao longo da aresta, a derivada normal também é um polinômio na variável s.

A condição de conformidade da aproximação impõe condições sobre o grau do polinômio aproximador, e, inversamente, ao escolhermos o grau do polinômio aproximador, já é possível caracterizar a aproximação segundo esse critério e definir as variáveis nodais apropriadas. Trabalhar com elementos conformes pode trazer dificuldades computacionais, devido ao alto grau exigido para os polinômios aproximadores [1].

Neste trabalho utilizamos esses dois tipos de aproximação: uma quadrática não conforme e uma de quinto grau, conforme. A escolha da aproximação conforme se justifica pelo fato de que, além de a aproximação com elementos conformes ser feita "por dentro" do espaço de funções $H_0^2(\Omega)$, as variáveis nodais que aparecem em elementos desse tipo são favoráveis para o tratamento do problema com outras condições de contorno, por exemplo, a de bordo livre ou simplesmente suportado, como veremos nas próximas seções, porque envolvem derivadas de ordem mais alta.

2.1.2 Discretização do funcional

A forma quadrática (2.3) será calculada em cada triângulo T_i da malha. Por isso, separamos $\mathscr{I}$ em integrais *locais* $\mathscr{I}_i$, com $i \in 1, 2, 3, ..., N$ (uma para cada elemento triangular). Em cada T_i a integral $\mathscr{I}_i$ é calculada utilizando a função aproximadora u_i e suas derivadas.

Para facilitar a discretização, separamos da integral $\mathscr{I}$ (2.3) as duas integrais:

$$\mathscr{I}^{(1)} = \iint_{T_i} u_{xx}^2 + u_{yy}^2 + 2\nu u_{xx}u_{yy} + 2(1-\nu)u_{xy}^2 \; dxdy \tag{2.6}$$

e

$$\mathscr{I}^{(2)} = \iint_{T_i} u^2 \; dxdy \; . \tag{2.7}$$

Em cada triângulo, elas serão representadas por formas quadráticas do tipo $\vec{u_i}^T S_i \vec{u_i}$ e $\vec{u_i}^T M_i \vec{u_i}$, em que $\vec{u_i} \in \mathbb{R}^n$ é o vetor de variáveis nodais, e S_i, M_i são matrizes quadradas simétricas, chamadas de *matriz de rigidez* e *matriz de massa* locais, respectivamente. Quando olhamos para o problema globalmente, i.e., considerando todos os elementos triangulares e, portanto, a malha sobre Ω como um todo, teremos matrizes simétricas de rigidez e de massa globais, S e M, e um vetor de variáveis

nodais globais $\vec{\mathbf{u}} \in \mathbb{R}^m$, onde m é o número de variáveis nodais da malha. As matrizes S e M serão montadas a partir das matrizes de rigidez e massa locais, S_i e M_i, assim como o vetor $\vec{\mathbf{u}}$.

As matrizes de rigidez e massa globais, S e M, representam as formas quadráticas discretizadas $\mathscr{I}^{(1)}$ e $\mathscr{I}^{(2)}$ respectivamente, a partir das quais obtemos um problema discreto a partir do problema contínuo, no qual o objetivo é determinar o vetor de variáveis nodais global $\vec{\mathbf{u}}$ ao invés da função u. O próximo passo é achar pontos críticos não triviais de $\mathscr{I} = \mathscr{I}^{(1)} - \lambda\mathscr{I}^{(2)}$, já discretizado, ou seja, achar pontos críticos (não triviais) de $\vec{\mathbf{u}} \mapsto \vec{\mathbf{u}}^T(S - \lambda M)\vec{\mathbf{u}}$. Vejamos qual equação discreta o vetor $\vec{\mathbf{u}}$ deverá satisfazer.

Seja A matriz simétrica e $Q(\vec{\mathbf{u}}) = \vec{\mathbf{u}}^T A\vec{\mathbf{u}} = \langle \vec{\mathbf{u}}, A\vec{\mathbf{u}} \rangle$.

Então, $\vec{\mathbf{u}}$ é ponto crítico de Q se e somente se

$$\lim_{t\to 0} \frac{Q(\vec{\mathbf{u}} + tv) - Q(\vec{\mathbf{u}})}{t} = 0$$

para qualquer $v \in \mathbb{R}^m$. Mas

$$Q(\vec{\mathbf{u}} + tv) - Q(\vec{\mathbf{u}}) = \langle \vec{\mathbf{u}} + tv, A(\vec{\mathbf{u}} + tv) \rangle - \langle \vec{\mathbf{u}}, A\vec{\mathbf{u}} \rangle = t\langle v, A\vec{\mathbf{u}} \rangle + t\langle \vec{\mathbf{u}}, Av \rangle + t^2\langle v, Av \rangle \ ,$$

e, sendo A simétrica,

$$\lim_{t\to 0} \frac{Q(\vec{\mathbf{u}} + tv) - Q(\vec{\mathbf{u}})}{t} = \frac{2t\langle v, A\vec{\mathbf{u}} \rangle + t^2\langle v, A\vec{\mathbf{u}} \rangle}{t} = 2\langle v, A\vec{\mathbf{u}} \rangle \ .$$

Portanto, $\vec{\mathbf{u}}$ é crítico de Q se e somente se $\langle v, A\vec{\mathbf{u}} \rangle = 0$, para todo $v \in \mathbb{R}^m$, que equivale à condição $A\vec{\mathbf{u}} = 0$.

Como S e M são simétricas, então $\vec{\mathbf{u}}$ é ponto crítico de $\vec{\mathbf{u}} \mapsto \vec{\mathbf{u}}^T(S - \lambda M)\vec{\mathbf{u}}$ se e somente se

$$S\vec{\mathbf{u}} = \lambda M\vec{\mathbf{u}} \ .$$

Obtemos assim um problema de autovalor (generalizado) discreto relativo ao problema inicial (2.1).

A discretização do funcional consiste, essencialmente, na escolha do polinômio aproximador u_i em cada T_i e na montagem das matrizes de rigidez e massa locais e globais. Neste trabalho, usamos dois tipos de aproximação polinomial: uma em que o polinômio u_i é de segundo grau (quadrática) e outra em que o polinômio u_i é de quinto grau, das quais trataremos na próxima seção. Apesar do grau distinto das aproximações, é possível olhar esse processo de uma maneira geral, antes da escolha do grau do polinômio.

2.1.3 Técnicas para discretização do funcional

A integral discretizada pode ser organizada e escrita com uma estrutura matricial, envolvendo produto de matrizes. Cada etapa do processo de integração poderá ser representada (ou caracterizada) por uma matriz. Desse modo, as matrizes de rigidez e massa locais poderão ser representadas por uma composição (multiplicação) das matrizes de cada etapa. Nessa pequena subseção, falaremos de etapas que podem estar presentes no processo de discretização e integração. Isso será útil na hora de olharmos para os casos particulares.

Seja p_i um polinômio de grau n escolhido para aproximar a função $u(x, y)$ num elemento trian-

gular T_i.

$$p_i(x,y) = a_1 + a_2x + a_3y + a_4x^2 + a_5xy + a_6y^2 + a_7x^3 + ... + a_{k-1}xy^{n-1} + a_ky^n \ ,$$

em que $k = (n+1)(n+2)/2$.

(i) Matrizes de derivação

Para compor a parte 1 (2.6) da integral variacional (2.3), precisaremos das derivadas de segunda ordem de p_i. Quando derivamos um polinômio p, de grau n, 2 vezes, o resultado é um novo polinômio q, cujo grau é $m = n-2$. Esse novo polinômio tem menos termos e portanto um número menor de coeficientes. Além disso, os coeficientes de q estão linearmente relacionados com os coeficientes de p, através do operador linear que chamaremos aqui de *operador derivação*.

Vamos olhar para o operador derivação D_{xx} que deriva p duas vezes com relação à variável x.

$$\begin{aligned} D_{xx} : \mathbb{R}^k &\longrightarrow \mathbb{R}^l \\ (a_1, a_2, ..., a_k) &\mapsto (b_1, b_2, ..., b_l) \end{aligned} \tag{2.8}$$

em que $\vec{a} = (a_1, a_2, ..., a_k)$ é o vetor de coeficientes de p e $\vec{b} = (b_1, b_2, ..., b_l)$ vetor de coeficientes de q com $l = (m+1)(m+2)/2$. O operador D_{xx} pode ser representado por uma matriz $[D_{xx}]_{l\times k}$, que chamaremos de *matriz de derivação* e que nos fornece a relação matricial

$$\vec{b} = [D_{xx}]\vec{a} \ .$$

(ii) Relação entre variáveis nodais e coeficientes da aproximação polinomial

Os coeficientes $(a_1, a_2, ..., a_k)$ de p estão relacionados com as variáveis nodais locais $\vec{u}_i$ de cada T_i. Essa relação pode ser estabelecida de forma linear – calculando $p_i(x,y)$ em cada nó do elemento triangular – e pode ser representada através de uma matriz, que chamaremos de C_i:

$$\vec{u}_i = C_i\vec{a}$$

e

$$\vec{a} = \bar{C_i}^{-1}\vec{u}_i \ .$$

(iii) Mudança de coordenadas

Em alguns casos, dependendo do tipo de variáveis nodais usadas no elemento finito, é conveniente o uso de uma mudança de coordenadas, que leva o triângulo T_i, de vértices $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ e $P_3 = (x_3, y_3)$, num triângulo padrão T_0, de coordenadas (0,0), (1,0) e (0,1). Para escrevê-la, localizaremos esse triângulo padrão no plano $\xi\eta$. Essa mudança de coordenadas é dada por

$$\begin{aligned} F : \mathbb{R}^2 &\longrightarrow \mathbb{R}^2 \\ (x,y) &\mapsto (\xi, \eta) \ . \end{aligned} \tag{2.9}$$

Sua inversa, $F^{-1} : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$, que leva T_0 em T_i é tal que

$$\left\{ \begin{array}{rcccccc} x & = & x_1 & + & (x_2 - x_1)\xi & + & (x_3 - x_1)\eta \\ y & = & y_1 & + & (y_2 - y_1)\xi & + & (y_3 - y_1)\eta \end{array} \right. \tag{2.10}$$

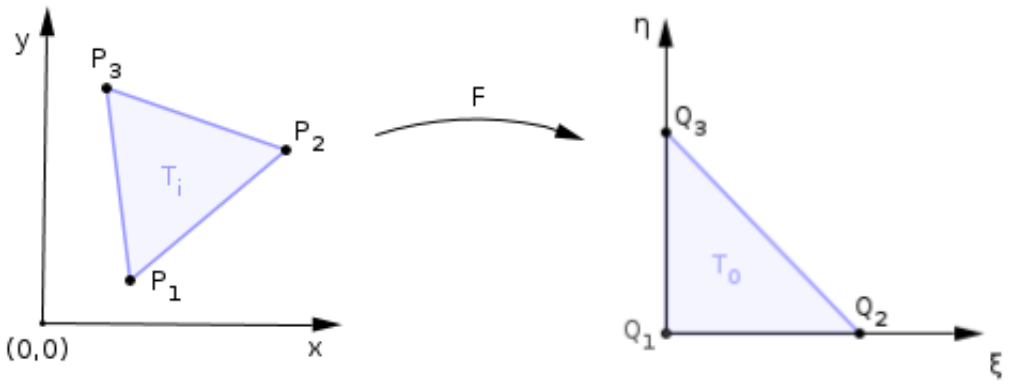


Figura 2.2: *Mudança de coordenadas.* $P_j = (x_j, y_j)$*, para* $j = 1, 2, 3$ *e* $Q_1 = (0, 0)$ *;* $Q_2 = (1, 0)$ *e* $Q_3 = (0, 1)$.

A aproximação polinomial pode ser feita diretamente em T_0 e, se essa transformação for usada para isso, a matriz C_i se torna constante (independente da malha).

A transformação acima também pode ser útil simplesmente para facilitar o cálculo de integrais em T_i, usando o teorema de mudança de variável na integral. Nesse caso, obteremos do polinômio p, definido no plano xy, um novo polinômio $\tilde{p} = (p \circ F^{-1})(\xi, \eta)$, de mesmo grau, definido no plano $\xi\eta$. Se uma mudança de variáveis na integral for feita, obteremos:

$$\iint_{T_i} p(x, y)\ dxdy = \iint_{T_0} J_i\ (p \circ F^{-1})(\xi, \eta)\ d\xi d\eta \tag{2.11}$$

em que J_i é o determinante da matriz jacobiana da transformação afim F^{-1}:

$$J_i = \begin{vmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \end{vmatrix} = \begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix} = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) = \overline{x}_2\overline{y}_3 - \overline{x}_3\overline{y}_2\ , \tag{2.12}$$

em que $\overline{x}_j = x_j - x_1$ para $j = 2, 3$.

Os coeficientes dos polinômios p e $\tilde{p}$ se relacionam linearmente, e essa relação também pode ser representada por uma matriz. Se $(b_1, b_2, ..., b_k)$ são os coeficientes de p no plano xy e $(\beta_1, \beta_2, ..., \beta_k)$ são os coeficientes de $\tilde{p}$ em $\xi\eta$, a relação entre essas duas famílias de coeficientes é dada por

$$\beta = Db\ . \tag{2.13}$$

Tal relação pode ser estabelecida substituindo $(x, y) = F^{-1}(\xi, \eta)$ na expressão do polinômio $p(x, y)$. E é o que veremos a seguir.

A transformação $F : T_i \subset \mathbb{R}^2 \longrightarrow T_0 \subset \mathbb{R}^2$ é composta por uma translação e por uma parte linear. Vamos considerar essas duas partes separadamente. Chamamos a translação de F_1 e a parte linear de F_2.

Começamos com a translação, que leva o polinômio $p(x, y)$ do plano xy, com coeficientes $a_i's$,

num polinômio no plano $\bar{x}\bar{y}$ e com coeficientes $b_i's$:

$$\begin{aligned} F_1 : \mathbb{R}^2 &\longrightarrow \mathbb{R}^2 \\ (x, y) &\mapsto (\overline{x}, \overline{y}) \end{aligned} \tag{2.14}$$

é tal que

$$\begin{cases} \overline{x} = x - x_1 \\ \overline{y} = y - y_1 \end{cases}.$$

Substituindo tal translação no polinômio p com coeficientes $a_i's$, obtemos outro polinômio $\overline{p}$, de mesmo grau, agora com coeficientes $b_i's$. A relação entre os coeficientes pela translação pode ser representada pela matriz $D_1 : \vec{b} = D_1\vec{a}$.

D_1 é uma matriz diagonal superior, possui entrada igual a 1 na diagonal principal, suas entradas dependem apenas do ponto (ou as coordenadas do vetor de translação) (x_1, y_1)) e sua primeira linha possui as parcelas do polinômio aplicado a este ponto.

No caso de um polinômio cúbico, por exemplo,

$$D_1 = \begin{bmatrix} 1 & x_1 & y_1 & x_1^2 & x_1y_1 & y_1^2 & x_1^3 & x_1^2y_1 & x_1y_1^2 & y_1^3 \\ & 1 & 0 & 2x_1 & y_1 & 0 & 3x_1^2 & 2x_1y_1 & y_1^2 & 0 \\ & & 1 & 0 & x_1 & 2y_1 & 0 & x_1^2 & 2x_1y_1 & 3y_1^2 \\ & & & 1 & 0 & 0 & 3x_1 & y_1 & 0 & 0 \\ & & & & 1 & 0 & 0 & 2x_1 & 2y_1 & 0 \\ & & & & & 1 & 0 & 0 & x_1 & 3y_1 \\ & & 0 & & & & 1 & 0 & 0 & 0 \\ & & & & & & & 1 & 0 & 0 \\ & & & & & & & & 1 & 0 \\ & & & & & & & & & 1 \end{bmatrix}$$

Agora, consideramos a transformação F_2 que é a parte linear de F. Sem a parte da translação, as substituições ficam mais simples.

$$\begin{aligned} F_2 : \mathbb{R}^2 &\longrightarrow \mathbb{R}^2 \\ (\overline{x}, \overline{y}) &\mapsto (\xi, \eta) \end{aligned} \tag{2.15}$$

e F_2^{-1} é tal que

$$\begin{cases} x = \overline{x}_2\xi + \overline{x}_3\eta \\ y = \overline{y}_2\xi + \overline{y}_3\eta \end{cases}. \tag{2.16}$$

Para obter a matriz que representa F_2, representaremos um polinômio p em duas variáveis da seguinte forma:

$$\sum_{j=0}^{gr(p)} \sum_{l=0}^{j} b_{j-l,l}\ x^{j-l}y^j$$

Essa notação é diferente, daquela apresentada no início da seção: ela apresenta dois somatórios e coeficientes duplamente indexados, mas é conveniente para expressar a relação que queremos para um polinômio de grau arbitrário. Seja $N = gr(p)$.

Aplicando a transformação F_2 no polinômio

$$\sum_{j=0}^{N}\sum_{l=0}^{j} b_{j-l,l}x^{j-l}y^{j}$$

obtemos um polinômio

$$\sum_{j=0}^{N}\sum_{l=0}^{j} \alpha_{j-l,l}\xi^{j-l}\eta^{j}.$$

Os coeficientes $b_{j-l,l}$'s e $\alpha_{j-l,l}$'s se relacionam pela equação (cuja demonstração pode ser encontrada em [10])

$$\alpha_{j-l,l} = \sum_{p=0}^{j} b_{j-p,p}\left(\sum_{m=0}^{l}\binom{j-p}{l-m}\binom{p}{m}\ \overline{x}_3{}^{l-m}\ \overline{x}_2{}^{j-p-l+m}\ \overline{y}_3{}^{m}\ \overline{y}_2{}^{p-m}\right)\ .$$

Assim, podemos escrever $\vec{\alpha} = D_2\vec{b}$, onde: $\vec{\alpha} = (\alpha_{00}, \alpha_{10}, \alpha_{01}, \alpha_{20}, \alpha_{11}, \alpha_{02}, ..., \alpha_{N,0}, ..., \alpha_{N-l,l}, \alpha_{0N})$, $\vec{b} = (b_{00}, b_{10}, b_{01}, b_{20}, b_{11}, b_{02}, ..., b_{N,0}, ..., b_{N-l,l}, b_{0N})$ e D_2 tem estrutura diagonal em blocos.

No caso de um polinômio cúbico, por exemplo,

$$D_2 = \begin{bmatrix}
1 & & & & & & & & & \\
\overline{x}_2 & \overline{y}_2 & & 0 & & & 0 & & & \\
\overline{x}_3 & \overline{y}_3 & & & & & & & & \\
 & & \overline{x}_2{}^2 & \overline{y}_2\overline{x}_2 & \overline{y}_2{}^2 & & & & & \\
0 & & 2\overline{x}_2\overline{x}_3 & \overline{y}_2\overline{x}_3 + \overline{y}_3\overline{x}_2 & 2\overline{y}_2\overline{y}_3 & & & & & \\
 & & \overline{x}_3{}^2 & \overline{y}_3\overline{x}_3 & \overline{y}_3{}^2 & & 0 & & & \\
 & & & & & \overline{x}_2^3 & \overline{x}_2{}^2\overline{y}_2 & \overline{y}_2{}^2\overline{x}_2 & \overline{y}_2{}^3 \\
 & & & & & \overline{x}_3\overline{x}_2{}^2 & 2\overline{x}_3\overline{x}_2\overline{y}_2 + \overline{x}_2{}^2\overline{y}_3 & 2\overline{x}_2\overline{y}_3\overline{y}_2 + \overline{x}_3\overline{y}_2{}^2 & 3\overline{y}_2{}^2\overline{y}_3 \\
 & & 0 & & & \overline{x}_3{}^2\overline{x}_2 & 2\overline{x}_3\overline{y}_2\overline{y}_3 + \overline{x}_3{}^2\overline{y}_2 & 2\overline{x}_3\overline{y}_3\overline{y}_2 + \overline{x}_2\overline{y}_3{}^2 & 3\overline{y}_2\overline{y}_3{}^2 \\
 & & & & & \overline{x}_3{}^3 & \overline{y}_3\overline{x}_3{}^2 & \overline{x}_3\overline{y}_3{}^2 & \overline{y}_3{}^3
\end{bmatrix}$$

A matriz D, que relaciona os coeficientes no plano xy com aqueles do plano $\xi\eta$ é dada então pela composição $D_2 \circ D_1$, pois $\vec{\alpha} = D_2\vec{b} = D_2 \circ D_1\vec{a}$ logo $\vec{\alpha} = D\vec{a}$ com $D = D_2 \circ D_1$.

(iv) Integral de produtos de dois polinômios

Integrar polinômios em duas variáveis no triângulo padrão T_0 é calcular a integral de parcelas do tipo $\iint_{T_0} \xi^p\eta^q\ d\xi d\eta$, em que $p, q \in \mathbb{N}$. Para fazer esse tipo de cálculo, podemos utilizar o seguinte resultado:

Proposição 2.1.1. *Seja $T_0 \subset \mathbb{R}^2$ o triângulo de coordenadas $(0,0), (1,0)$ e $(0,1)$. Se $p, q \in \mathbb{N}$, então*

$$I_{pq} \overset{def}{=} \iint_{T_0} \xi^p\eta^q\ d\xi d\eta = \frac{p!q!}{(p+q+2)!}\ .$$

Em nossa implementação, precisaremos integrar produtos de dois polinômios de mesmo grau N, com vetores de coeficientes α e β, resultando numa forma bilinear que pode ser representada matricialmente por

$$\alpha^T P\beta\ ,$$

em que P é matriz cujas entradas são as integrais dos monômios $\iint_{T_0} \xi^p \eta^q \, d\xi d\eta$. A seguir descreveremos o comportamento da matriz P em função dos polinômios cujo produto será integrado, e apresentaremos uma maneira sistemática de montar a matriz P.

A matriz P está associada à forma bilinear simétrica proveniente da integral, do produto de dois polinômios de mesmo grau N, no triângulo padrão T_0. A dimensão k de P depende do grau N dos polinômios envolvidos da seguinte forma:

$$k = \textit{soma dos } N \textit{ primeiros termos da progressão aritmética (P.A) de razão } 1 = \frac{(N+1)(N+2)}{2} \ .$$

Para representar os polinômios envolvidos e seus produtos, usaremos uma notação (diferente daquela usada no item anterior) que permite expressar o polinômio com apenas um somatório e de modo que os coeficientes $\alpha's$ ou $\beta's$ sejam indexados por apenas uma variável. Isso facilitará a escrita do produto de tais polinômios. Para isso, faremos uso da Tabela (2.1) que associa cada índice i (ou j) a um par (i_1, i_2) (ou (j_1, j_2)).

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**...**	$k-N$	$k-N+1$	**...**	**k-1**	**k**
i_1	0	1	0	2	1	0	3	2	1	0	...	N	N -1	...	1	0
i_2	0	0	1	0	1	2	0	1	2	3	...	0	1	...	$N-1$	N

Tabela 2.1: *Potências dos monômios de um polinômio em duas variáveis.*

A partir daí escrevemos

$$p(\xi, \eta) = \sum_{i=1}^{k} \alpha_i \xi^{i_1} \eta^{i_2}$$

e

$$q(\xi, \eta) = \sum_{j=1}^{k} \beta_j \xi^{j_1} \eta^{j_2} \ .$$

O produto fica assim:

$$p(\xi,\eta)q(\xi,\eta) = \left(\sum_{i=1}^{k} \alpha_i \xi^{i_1} \eta^{i_2}\right)\left(\sum_{j=1}^{k} \beta_j \xi^{j_1} \eta^{j_2}\right) = \sum_{i=1}^{k}\sum_{j=1}^{k} \alpha_i\beta_j \xi^{i_1+j_1} \eta^{i_2+j_2} \ .$$

Então,

$$\iint_{T_0} \sum_{i=1}^{k}\sum_{j=1}^{k} \alpha_i\beta_j \xi^{i_1+j_1} \eta^{i_2+j_2} \, d\xi d\eta = \sum_{i=1}^{k}\sum_{j=1}^{k} \alpha_i\beta_j \iint_{T_0} \xi^{i_1+j_1} \eta^{i_2+j_2} \, d\xi d\eta = \sum_{i=1}^{k}\sum_{j=1}^{k} \alpha_i\beta_j p_{ij} = \vec{\alpha}^T P \vec{\beta}.$$

A matriz P pode ser definida como $P = (p_{ij})_{k\times k}$ onde

$$p_{ij} = \iint_{T_0} \xi^{i_1+j_1} \eta^{i_2+j_2} \, d\xi d\eta = I_{i_1+j_1, i_2+j_2} = \frac{(i_1+j_1)!(i_2+j_2)!}{[(i_1+j_1)+(i_2+j_2)+2]!} \ .$$

A última igualdade vem da Proposição (2.1.1), a partir do qual também podemos observar a simetria de P, uma vez que $I_{pq} = I_{qp}$. Quando os polinômios envolvidos no produtos são iguais, então P está associada a uma forma quadrática:

$$\iint_{T_0} p(\xi,\eta)^2 d\xi d\eta = \iint_{T_0} \left(\sum_{i=1}^{k} \alpha_i \xi^{i_1}\eta^{i_2}\right)^2 = \iint_{T_0} \left(\sum_{i=1}^{k}\sum_{j=1}^{k} \alpha_i\alpha_j \xi^{i_1+j_1}\eta^{i_2+j_2}\right) d\xi d\eta = \sum_{i=1}^{k}\sum_{j=1}^{k} \alpha_i\alpha_j \iint_{T_0} \xi^{i_1+j_1}\eta^{i_2+j_2} = \vec{\alpha}^T P \vec{\alpha}\ .$$

Nas próximas seções apresentaremos as diferentes aproximações que usamos para implementar o problema numericamente: a aproximação quadrática e a de quinto grau. Usaremos, para tanto, as considerações feitas nesta seção.

2.2 Aproximação quadrática

O objetivo dessa seção é apresentar a implementação numérica do método de elementos finitos para o problema de autovalor (2.1) usando uma aproximação polinomial de segundo grau. Tal implementação consiste na montagem das matrizes de massa e rigidez que compõem o problema de autovalor.

Primeiramente, apresentaremos as características do elemento finito triangular que usamos nesse caso; em seguida, a aproximação polinomial que será usada localmente e como ela transforma o funcional contínuo em um discreto. Por fim, apresentamos a "cara" das matrizes locais e a montagem das matrizes globais a partir delas.

O elemento finito que será utilizado nesse caso possui seis variáveis nodais que estão distribuídas nos vértices e pontos médios das arestas de cada triângulo T_i como mostra a figura (2.3).

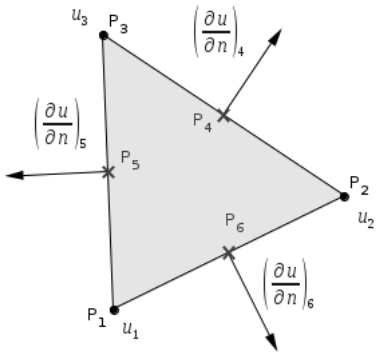


Figura 2.3: *Elemento finito da aproximação quadrática*

As variáveis $u_1, u_2, u_3, \left(\frac{\partial u}{\partial \vec{n}_1}\right)_4, \left(\frac{\partial u}{\partial \vec{n}_2}\right)_5, \left(\frac{\partial u}{\partial \vec{n}_3}\right)_6$ presentes no elemento finito compõem o vetor de variáveis nodais $\vec{u_i}$. Para facilitar a notação, denotaremos a derivada normal $\left(\frac{\partial u}{\partial \vec{n}_j}\right)$ por $\partial_{n_j} u$, e os índices das variáveis nodais, que indicam os nós do elemento triangular serão colocados na parte superior: $\vec{u_i} = \left(u^1, u^2, u^3, (\partial_{n_1}u)^4, (\partial_{n_2}u)^5, (\partial_{n_3}u)^6\right)$.

A aproximação quadrática é *não conforme*. Tendo em mente o Teorema 2.1.1 e a caracterização dada no início do capítulo, isso se dá porque os polinômios aproximadores u_i e u_j não se emendam

continuamente ao longo de uma aresta (arbitrária) entre dois elementos. Ao longo da aresta, u_i é um polinômio de grau 2 na variável s:

$$u_i(s) = u_i(x(s), y(s)) = e_1 + e_2 s + e_3 s^2$$

Entretanto, não há variáveis nodais suficientes para determinar os parâmetros e_1, e_2, e_3 (e, portanto, para determinar u_i de maneira única). As variáveis nodais presentes nos vértices nos fornecem apenas duas relações

$$\begin{aligned} u_i(x_I, y_I) &= u_i(s=0) = e_1 \\ u_i(x_F, x_F) &= u_i(s=1) = e_1 + e_2 + e_3 \end{aligned} \tag{2.17}$$

A derivada normal no ponto médio $\partial_n u_i$ é um polinômio de primeiro grau em parâmetros distintos dos $e_i's$, de modo que ela não contribui para a continuidade de u_i. Portanto, temos garantida a continuidade de u_i apenas nos vértices da aresta e a continuidade da derivada normal $\partial_n u_i$ apenas no ponto médio da aresta.

Considere as duas partes (2.6) e (2.7) do funcional (2.3), num triângulo arbitrário T_i da malha:

$$\mathscr{I}_i^{(1)} = \iint_{T_i} u_{xx}^2 + u_{yy}^2 + 2\nu u_{xx} u_{yy} + 2(1-\nu) u_{xy}^2 \; dxdy$$

e

$$\mathscr{I}_i^{(2)} = \iint_{T_i} u^2 \; dxdy \; .$$

A função $u(x, y)$ foi aproximada por um polinômio u_i, de grau 2, nas variáveis x e y:

$$u_i(x, y) = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy + a_6 y^2 \tag{2.18}$$

Substituindo tal expressão em (2.6), teremos quatro parcelas de integrais de produtos de derivadas segundas de u_i. Essas integrais serão calculadas em função dos coeficientes $a_1, a_2, ..., a_6$ de u_i. Para o cálculo do integrando da primeira integral $\mathscr{I}_i^{(1)}$, podemos usar matrizes de derivação descritas no item (i) da seção anterior. Nesse caso, como estamos derivando duas vezes um polinômio de grau 2, o resultado é um polinômio de grau 0 (constante), e por isso as matrizes de derivação têm dimensão 1×6:

$$[D_{xx}] = \begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 0 \end{bmatrix}, \; [D_{yy}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix} \quad e \quad [D_{xy}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.19}$$

Denotaremos $[D_{xx}](a_1, a_2, a_3, ..., a_6)^T = 2a_4 = D_{xx} u_i$, $[D_{yy}](a_1, a_2, a_3, ..., a_6)^T = 2a_6 = D_{yy} u_i$ e $[D_{xy}](a_1, a_2, a_3, ..., a_6)^T = a_5 = D_{xy} u_i$.

Assim, para a primeira parcela, temos que

$$\iint_{T_i} (u_{i_{xx}})^2 \; dxdy = \iint_{T_i} (D_{xx} u_i)^2 \; dxdy = \iint_{T_i} (2a_4)^2 dxdy = 4{a_4}^2 \iint_{T_i} 1 \; dxdy = 4{a_4}^2 A_i \; ,$$

em que A_i é a área do triângulo T_i. Cada parcela como a integral acima é uma forma quadrática do

vetor de coeficientes $\vec{a} = (a_1, a_2, ..., a_6)$, que pode ser calculada em função de matrizes de derivação. Mas, nesse caso, em que o polinômio resultante da derivação é constante, os cálculos se simplificam, de modo que o uso das matrizes de derivação se torna até desnecessário:

$$\begin{aligned}\mathscr{I}_i^{(1)} &= \iint_{T_i} 4{a_4}^2 + 4{a_6}^2 + 2\nu 4a_4a_6 + 2(1-\nu){a_5}^2 \; dxdy = \\ &= 4{a_4}^2\left(\iint_{T_i} 1 \; dxdy\right) + 4{a_6}^2\left(\iint_{T_i} 1 \; dxdy\right) + 8\nu a_4a_6\left(\iint_{T_i} 1 \; dxdy\right) + 2(1-\nu){a_5}^2\left(\iint_{T_i} 1 \; dxdy\right) = \\ &= 4{a_4}^2A_i + 4{a_6}^2A_i + 8\nu a_4a_6A_i + 2(1-\nu){a_5}^2A_i = A_i\left[4{a_4}^2 + 4{a_6}^2 + 8\nu a_4a_6 + 2(1-\nu){a_5}^2\right] \; ,\end{aligned} \tag{2.20}$$

em que a área A_i do triângulo T_i pode ser calculada através do determinante

$$\begin{aligned}A_i = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} &= x_1y_2 + y_1x_3 + y_3x_2 - (y_2x_3 + y_3x_1 + x_2y_1) = \\ &= y_3(x_2 - x_1) + y_2(x_1 - x_3) + y_1(x_3 - x_2) = \\ &= (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) \; .\end{aligned} \tag{2.21}$$

Portanto, $\mathscr{I}_i^{(1)}$ é uma forma quadrática dos coeficientes $a_1, a_2, ..., a_6$ de u_i e pode ser escrita como $a^T\tilde{S}_ia$, em que

$$\tilde{S}_i = A_i\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 4\nu \\ 0 & 0 & 0 & 0 & 2(1-\nu) & 0 \\ 0 & 0 & 0 & 4\nu & 0 & 4 \end{bmatrix} \tag{2.22}$$

é uma matriz simétrica que, exceto pelo fator A_i, não depende do triângulo T_i e, portanto, não depende da geometria da malha. Para escrever $\mathscr{I}_i^{(1)}$ como forma quadrática em função do vetor de variáveis nodais $\vec{u_i} = \left(u^1, u^2, u^3, (\partial_{n_1}u)^4, (\partial_{n_2}u)^5, (\partial_{n_3}u)^6\right)$, usaremos a relação que existe entre este vetor e o vetor de coeficientes de u_i : $(a_1, ...a_6)$

$$\vec{u}_i = C_i\vec{a} \; ,$$

como descrito no item (ii) da seção anterior. A relação $\vec{u}_i = C_i\vec{a}$ é construída escrevendo os valores de $u_i(x, y)$ e de $\partial_n u_i(x, y)$ em cada um dos nós (vértices e pontos médios, respectivamente) do elemento finito:

$$
\begin{aligned}
u_1 =& u_i(x_1, y_1) = a_1 + a_2x_1 + a_3y_1 + a_4{x_1}^2 + a_5x_1y_1 + a_6{y_1}^2 \\
u_2 =& u_i(x_2, y_2) = a_1 + a_2x_2 + a_3y_2 + a_4{x_2}^2 + a_5x_2y_2 + a_6{y_2}^2 \\
u_3 =& u_i(x_3, y_3) = a_1 + a_2x_3 + a_3y_3 + a_4{x_3}^2 + a_5x_3y_3 + a_6{y_3}^2 \qquad (2.23)\\
(\partial_{n_1}u_i)_4 =& A_1u_{i_x}(x_4, y_4) + B_1u_{i_y}(x_4, y_4) = A_1(a_2 + 2a_4x_4 + y_4) + B_1(a_3 + x_4a_5 + 2a_6y_4) \\
(\partial_{n_2}u_i)_5 =& A_2u_{i_x}(x_5, y_5) + B_2u_{i_y}(x_5, y_5) = A_2(a_2 + 2a_4x_5 + y_5) + B_2(a_3 + x_5a_5 + 2a_6y_5) \\
(\partial_{n_3}u_i)_6 =& A_3u_{i_x}(x_6, y_6) + B_3u_{i_y}(x_6, y_6) = A_3(a_2 + 2a_4x_4 + y_4) + B_3(a_3 + x_4a_5 + 2a_6y_4)\ .
\end{aligned}
$$

Os coeficientes A_j e B_j, $j = 1, 2, 3$ são coeficientes do vetor normal, quando escrito como combinação linear das derivadas de u nas direções canônicas. i.e., $\left(\partial_{n_j}u_i\right)_{j+3} = A_ju_{i_x} + B_ju_{i_y}$, onde as derivadas parciais são calculadas nos nós (x_{j+3}, y_{j+3}), para $j = 1, 2, 3$.

Assim, a matriz C_i depende da geometria da malha e é dada por

$$
C_i = \begin{bmatrix}
1 & x_1 & y_1 & {x_1}^2 & x_1y_1 & {y_1}^2 \\
1 & x_2 & y_2 & {x_2}^2 & x_2y_2 & {y_2}^2 \\
1 & x_3 & y_3 & {x_3}^2 & x_3y_3 & {y_3}^2 \\
0 & A_1 & B_1 & 2A_1x_4 & (A_1y_4 + B_1x_4) & 2B_1y_4 \\
0 & A_2 & B_2 & 2A_2x_5 & (A_2y_5 + B_2x_5) & 2B_2y_5 \\
0 & A_3 & B_3 & 2A_3x_6 & (A_3y_6 + B_3x_6) & 2B_3y_6
\end{bmatrix}.
$$

Sua inversa ${C_i}^{-1}$ nos fornece a relação $a = C_i^{-1}u_i$ e assim temos que

$$
\mathscr{I}_i^{(1)} = ({C_i}^{-1}u_i)^T\tilde{S}_i{C_i}^{-1}u_i = {u_i}^T({C_i}^{-T}\tilde{S}_iC_i)u_i = {u_i}^TS_iu_i\ , \qquad (2.24)
$$

em que $S_i = (C_i^{-1})^T\tilde{S}_iC_i$ é a matriz de rigidez local.

Para a segunda parte (2.7) do funcional, a equação (2.18) foi substituída na equação (2.7), de onde obtivemos a integral

$$
\mathscr{I}_i^{(2)} = \iint\limits_{T_i} u_i^2\ dx\mathrm{d}y = \iint\limits_{T_i} [a_1 + a_2x + a_3y + a_4x^2 + a_5xy + a_6y^2]^2\ dx\mathrm{d}y\ .
$$

Neste caso, foi preciso aplicar uma mudança de variável na integral para facilitar o cálculo. Utilizamos a mudança de coordenadas $F : T_i \subset \mathbb{R}^2 \longrightarrow T_0 \subset \mathbb{R}^2$, descrita no item (iii) da seção anterior – que mapeia de forma biunívoca o triângulo T_i no triângulo padrão T_0. Considerando sua inversa $F^{-1} : T_0 \subset \mathbb{R}^2 \longrightarrow T_i \subset \mathbb{R}^2$ (2.10), a integral $\mathscr{I}_i^{(2)}$ fica assim:

$$
\begin{aligned}
\mathscr{I}_i^{(2)} &= \iint\limits_{T_i} u_i^2\ dx\mathrm{d}y = \iint\limits_{T_i} [a_1 + a_2x + a_3y + a_4x^2 + a_5xy + a_6y^2]^2\ dxdy = \\
&= \iint\limits_{T_0} [\alpha_1 + \alpha_2\xi + \alpha_3\eta + \alpha_4\xi^2 + \alpha_5\xi\eta + \alpha_6\eta^2]^2\ J_i\ d\xi d\eta \qquad (2.25)
\end{aligned}
$$

em que $dxdy = J_i\ d\xi d\eta$ e J_i (2.12) é o determinante da matriz Jacobiana de F^{-1} .

Como vimos no item (iii) da seção anterior, para cada T_i os coeficientes $\alpha_1, ..., \alpha_6$ se relacionam com os coeficientes $a_1, ..., a_6$ pela relação $\vec{\alpha} = D_i\vec{a}$. Nesse caso a matriz D_i é facilmente calculada:

$$D_i = \begin{bmatrix} 1 & \overline{x}_1 & \overline{y}_1 & {\overline{x}_1}^2 & \overline{x}_1\overline{y}_1 & {\overline{y}_1}^2 \\ & \overline{x}_2 & \overline{y}_2 & 2\overline{x}_1\overline{x}_2 & \overline{x}_1\overline{y}_2 + \overline{x}_2 y_1 & 2\overline{y}_1\overline{y}_2 \\ & \overline{x}_3 & \overline{y}_3 & 2\overline{x}_1\overline{x}_3 & x_1\overline{y}_3 + \overline{x}_3\overline{y}_1 & 2\overline{y}_1\overline{y}_3 \\ & & & {\overline{x}_2}^2 & \overline{y}_2\overline{x}_2 & {\overline{y}_2}^2 \\ & & & \overline{x}_2\overline{x}_3 & \overline{y}_2\overline{x}_3 + \overline{y}_3\overline{x}_2 & \overline{y}_2\overline{y}_3 \\ & & & {\overline{x}_3}^2 & \overline{y}_3\overline{x}_3 & {\overline{y}_3}^2 \end{bmatrix} .$$

Continuando a expansão da integral $\mathscr{I}^{(2)}$, temos:

$$\mathscr{I}_i^{(2)} = \iint_{T_0} [\alpha_1 + \alpha_2\xi + \alpha_3\eta + \alpha_4\xi^2 + \alpha_5\xi\eta + \alpha_6\eta^2]^2 \; J_i \; d\xi d\eta =$$

$$J_i \iint_{T_0} [{\alpha_1}^2 + 2\alpha_1\alpha_2\xi + 2\alpha_1\alpha_3\eta + (2\alpha_1\alpha_4 + {\alpha_2}^2)\xi^2 + 2(\alpha_1\alpha_5 + \alpha_2\alpha_3)\xi\eta \; d\xi d\eta +$$

$$J_i \iint_{T_0} ({\alpha_3}^2 + 2\alpha_1\alpha_6)\eta^2 + 2\alpha_2\alpha_4\xi^3 + 2(\alpha_2\alpha_5 + \alpha_3\alpha_4)\xi^2\eta + 2(\alpha_2\alpha_6 + \alpha_3\alpha_5)\xi\eta^2 \; d\xi d\eta +$$

$$J_i \iint_{T_0} 2\alpha_3\alpha_6\eta^3 + {\alpha_4}^2\xi^4 + 2\alpha_4\alpha_5\eta\xi^3 + (2\alpha_4\alpha_6 + {\alpha_5}^2)\xi^2\eta^2 + 2\alpha_5\alpha_6\xi\eta^3 + \alpha_6^2\eta^4] \; d\xi d\eta \; . \tag{2.26}$$

As integrais de cada parcela são da forma $\iint_{T_0} \xi^p\eta^q \; d\xi d\eta$ e foram calculadas usando o teorema (2.1.1). Daí, obtemos de (2.26) que

$$\begin{aligned} \mathscr{I}_i^{(2)} &= J_i \Bigg(\frac{1}{2}{\alpha_1}^2 + 2\alpha_1\alpha_2\frac{1}{6} + 2\alpha_1\alpha_3\frac{1}{6} + (2\alpha_1\alpha_4 + {\alpha_2}^2)\frac{1}{12} + 2(\alpha_1\alpha_5 + \alpha_2\alpha_3)\frac{1}{24} + \\ &+ ({\alpha_3}^2 + 2\alpha_1\alpha_6)\frac{1}{12} + 2\alpha_2\alpha_4\frac{1}{20} + 2(\alpha_2\alpha_5 + \alpha_3\alpha_4)\frac{1}{60} + 2(\alpha_2\alpha_6 + \alpha_3\alpha_5)\frac{1}{60} + \\ &+ 2\alpha_3\alpha_6\frac{1}{20} + {\alpha_4}^2\frac{1}{30} + 2\alpha_4\alpha_5\frac{1}{120} + (2\alpha_4\alpha_6 + {\alpha_5}^2)\frac{1}{180} + 2\alpha_5\alpha_6\frac{1}{120} + \alpha_6^2\frac{1}{30} \Bigg) = \\ &= J_i\vec{\alpha}^T P\vec{\alpha} \; , \end{aligned} \tag{2.27}$$

em que P é a matriz simétrica e definida positiva

$$P = \begin{bmatrix} 1/2 & 1/6 & 1/6 & 1/12 & 1/24 & 1/12 \\ & 1/12 & 1/24 & 1/20 & 1/60 & 1/60 \\ & & 1/12 & 1/60 & 1/60 & 1/20 \\ & \textit{simétrica} & & 1/30 & 1/120 & 1/180 \\ & & & & 1/180 & 1/120 \\ & & & & & 1/30 \end{bmatrix} .$$

Portanto,$\mathscr{I}_i^{(2)}$ é forma quadrática e pode ser escrita da seguinte forma:

$$\begin{aligned} \mathscr{I}_i^{(2)} &= J_i(\vec{\alpha}^T P\vec{\alpha}) = J_i((D_i\vec{a})^T P D_i\vec{a}) = J_i(\vec{a}^T D_i^T P D_i\vec{a}) = \\ &= J_i((C_i^{-1}\vec{u}_i)^T D_i^T P D_i\vec{C}_i^{-1}\vec{u}_i) = J_i(\vec{u}_i^T(C_i^{-T} D_i^T P D_i C_i^{-1}) \; \vec{u}_i). \end{aligned} \tag{2.28}$$

A matriz $M_i = C_i^{-T} D_i^T P D_i C_i^{-1}$ é a *matriz de massa* local relativa ao problema de autovalor. A dimensão das matrizes de massa e rigidez globais depende do número de nós da malha e do número de variáveis incidentes em cada um desses nós.

Nesta aproximação, temos apenas uma variável nodal incidente em cada vértice. Logo, a dimensão da matriz será igual ao número total de nós da malha. A montagem das matrizes globais é feita da seguinte forma: para cada triângulo T_i da malha a matriz global recebe na posição (i_m, i_n) o valor que está na posição (m, n) da matriz local, para $m, n \in \{1, 2, 3, 4, 5, 6\}$. Ou seja: $S_{i_m,i_n} = S_{i_m,i_n} + S^i{}_{mn}$ no caso da matriz de rigidez e $M_{i_m,i_n} = M_{i_m,i_n} + M^i{}_{mn}$ no caso da matriz de massa.

Depois da montagem das matrizes globais a partir das matrizes locais, o último passo para finalizar esse processo é contabilizar as condições de contorno do problema. Estamos considerando a placa com bordo grampeado, em que os valores de u e de suas derivadas normais se anulam no bordo $\partial\Omega$ do domínio Ω. Sendo assim, os valores de u e $\partial_n u$ estão prescritos e são ambos iguais a zero. Assim, para todos os nós da fronteira de Ω, o valor de u e $\partial_n u$ já são conhecidos e portanto o problema de autovalor não precisa ser resolvido para eles. Aplicar essas condições de contorno, consiste, então, em eliminar das matrizes globais as linhas e colunas relativas àqueles nós que pertencem à fronteira do domínio.

2.3 Aproximação de Quinto Grau

O objetivo desta seção é apresentar a implementação numérica do método de elementos finitos para o problema de autovalor (2.1) usando uma aproximação polinomial de quinto grau. Assim como na seção anterior, tal implementação consiste na montagem das matrizes de massa e rigidez, que compõem as formas quadráticas relativas às partes $\mathscr{I}^{(1)}$ (2.6) e $\mathscr{I}^{(2)}$ (2.7) do funcional $\mathscr{I}$ (2.3) do problema de autovalor.

Novamente, como na seção anterior, apresentamos as características do elemento finito triangular que usamos neste caso, a aproximação polinomial que será usada localmente, as matrizes de massa e rigidez locais e a montagem das matrizes globais a partir delas.

O elemento finito que utilizamos para a aproximação de grau 5 possui 21 variáveis nodais, sendo elas: o valor da função u e de suas derivadas de até segunda ordem: $u, u_x, u_y, u_{xx}, u_{xy}, u_{yy}$ nos vértices do triângulo, e o valor das derivadas normais $\dfrac{\partial u}{\partial \vec{n}}$, nos pontos médios das arestas do triângulo, como mostra a Figura 2.4. A derivada normal será denotada por $\partial_n u$.

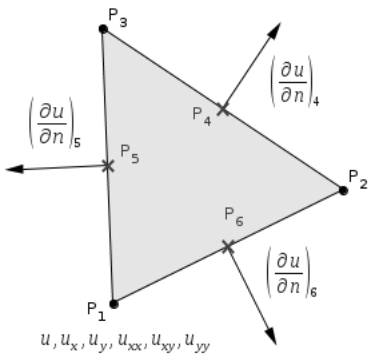


Figura 2.4: *Elemento finito da aproximação de quinto grau.*

O vetor de variáveis nodais foi organizado por tipo de variável:

$$\vec{u_i} = (u^1, u^2, u^3, u^1_x, u^2_x, u^3_x, u^1_y, u^2_y, u^3_y, u^1_{xx}, u^2_{xx}, u^3_{xx}, u^1_{xy}, u^2_{xy}, u^3_{xy}, u^1_{yy}, u^2_{yy}, u^3_{yy}, \partial_n u^4, \partial_n u^5, \partial_n u^6) \ ,$$

em que o índice acima de cada variável indica o nó (vértice ou ponto médio) do triângulo T_i em que seus valores foram calculados. Aproximamos localmente (i.e., em cada elemento triangular) a função $u(x, y)$ por um polinômio de quinto grau u_i nas variáveis x e y

$$\begin{aligned} u_i(x,y) = a_1 + a_2x + a_3y + a_4x^2 + a_5xy + a_6y^2 + a_7x^3 + a_8x^2y + a_9xy^2 + a_{10}y^3 + a_{11}x^4 + a_{12}x^3y + \\ + a_{13}x^2y^2 + a_{14}xy^3 + a_{15}y^4 + a_{16}x^5 + a_{17}x^4y + a_{18}x^3y^2 + a_{19}x^2y^3 + a_{20}xy^4 + a_{21}y^5. \end{aligned} \tag{2.29}$$

Optamos por essa implementação de grau 5, por dois motivos: esse elemento finito possui variáveis presentes nas condições de contorno do problema de vibração de uma placa com o *bordo livre* ou com o bordo *simplesmente suportado*. Saber trabalhar com essa implementação e com essas variáveis prepara o caminho para implementação do problema com essas diferentes condições de contorno.

Além disso, essa aproximação tem a característica de ser *conforme*. Tendo em mente o Teorema 2.1.1, isso significa que o polinômio aproximador u_i e sua derivada normal $\partial_n u_i$ são funções contínuas entre os elementos, o que pode ser concluído mostrando que as funções u_i e $\partial_n u_i$ são unicamente determinadas, ao longo de uma aresta (arbitrária) que liga dois elementos, pelas variáveis nodais presentes nesta aresta. Ao longo da aresta, u_i é um polinômio de grau 5:

$$u_i(x(s), y(s)) = u_i(s) = e_1 + e_2s + e_3s^2 + e_4s^3 + e_5s^4 + e_6s^5$$

e precisa de seis parâmetros para ser determinado.

Usaremos os valores de $\partial_s u_i$ e de $\partial_s(\partial_n u_i)$ nos pontos inicial e final da aresta, uma vez que esses valores são unicamente determinados pelas variáveis nodais presentes nos vértices do elemento triangular:

$$\begin{aligned} u_i(x_I, y_I) &= u_i(s=0) = e_1 & u_i(x_F, x_F) &= u_i(s=1) = e_1 + e_2 + e_3 + e_4 + e_5 + e_6 \\ \partial_s u_i(x_I, y_I) &= \partial_s u_i(s=0) = e_2 & \partial_s u_i(x_F, y_F) &= \partial_s u_i(s=1) = e_2 + 2e_3s + 3e_4s^2 + 4e_5s^3 + 5e_6s^4 \\ \partial_s(\partial_s u_i)(x_I, y_I) &= \partial_s(\partial_s u_i)(s=0) = e_3 & \partial_s(\partial_s u_i)(x_F, y_F) &= \partial_s(\partial_s u_i)(s=1) = 2e_3 + 6e_4s + 12e_5s^2 + 20e_6s^3 \end{aligned} \tag{2.30}$$

Essas seis equações são suficientes para determinar os coeficientes $e_1, ..., e_6$ do polinômio u_i. Portanto, u_i é contínuo entre os elementos. A derivada normal $\partial_n(s)$ é um polinômio de grau 4 ao longo da aresta: $\partial_n u_i(s) = d_1 + d_2s + d_3s^2 + d_4s^3 + d_5s^4$, que pode ser determinado pelas cinco relações abaixo

$\partial_n u_i(x_M, y_M) = \partial_n u_i(s = 1/2) = d_1 + \frac{1}{2}d_2 + \frac{1}{4}d_3 + \frac{1}{8}d_4 + \frac{1}{16}d_5$

$$\begin{aligned} \partial_n u_i(x_I, y_I) &= \partial_n u_i(s=0) = d_1 & \partial_n u_i(x_F, y_F) &= \partial_n u_i(s=1) = d_1 + d_2 + d_3 + d_4 \\ \partial_s(\partial_n u_i)(x_I, y_I) &= \partial_s(\partial_n u_i)(s=0) = d_2 & \partial_s(\partial_n u_i)(x_F, y_F) &= \partial_s(\partial_n u_i)(s=1) = d_2 + 2d_3 + 3d_4 + 4d_5 \end{aligned} \tag{2.31}$$

de onde concluímos que a derivada normal também é contínua ao longo da aresta.

Nessa aproximação, as matrizes de rigidez e massa locais têm dimensão 21. Denotaremos por $\vec{a_i} = (a_1, ..., a_{21})$ o vetor formado pelos coeficientes $a_1, ..., a_{21}$ do polinômio de quinto grau. Tais coeficientes são unicamente determinados pelas variáveis nodais pela relação $\vec{u_i} = C_i\vec{a}$, onde C_i é uma matriz que é obtida de maneira análoga àquela da aproximação quadrática, tem dimensão 21×21 e depende da geometria da malha – isto é, para cada elemento triangular, temos uma matriz C_i . Sua inversa ${C_i}^{-1}$ nos fornece a relação $\vec{a} = {C_i}^{-1}\vec{u}$ que será utilizada posteriormente.

Comecemos pela primeira parte do funcional $\mathscr{I}$: $\mathscr{I}^{(1)}$, calculada em cada triângulo T_i da malha. Calculamos suas parcelas separadamente:

$\iint_{T_i} {u_{xx}}^2 \, dxdy$ (1)

$\iint_{T_i} {u_{yy}}^2 \, dxdy$ (2)

$\iint_{T_i} {u_{xy}}^2 \, dxdy$ (3)

$\iint_{T_i} u_{xx}u_{yy} \, dxdy$ (4)

O polinômio u_i (2.29) foi substituído em cada uma dessas quatro expressões para o cálculo das integrais. Como $u(x, y)$ é polinômio de grau 5, os integrandos serão compostos por produtos de dois polinômios de grau 3. Descreveremos o procedimento detalhado para uma delas.

Para a integral (1):

$$u_{xx} = 2a_4 + 6a_7x + 2a_8y + 12a_{11}x^2 + 6a_{12}xy + 2a_{13}y^2 + 20a_{16}x^3 + 12a_{17}x^2y + 6a_{18}xy^2 + 2a_{19}y^3 \ .$$

Considere o operador derivação $D_{xx} : \mathbb{R}^{21} \longrightarrow \mathbb{R}^{10}$, como descrito em no item (i) da seção anterior (2.8). D_{xx} leva os 21 coeficientes $a_1, ..., a_{21}$, que determinam o polinômio de grau 5, em 10 coeficientes que determinam um polinômio de grau 3. No caso de $u_{i_{xx}}$, $(a_1, ..., a_{21}) \longmapsto (2a_4, 6a_7, 2a_8, 12a_{11}, 6a_{12}, 2a_{13}, 20a_{16}, 12a_{17}, 6a_{18}, 2a_{19})$. A matriz associada a esse operador, $[D_{xx}]$, considerando bases canônicas do $\mathbb{R}^{21}$ e $\mathbb{R}^{10}$, tem dimensão 10×21 e é dada por

$$[D_{xx}] = \begin{bmatrix}
0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0
\end{bmatrix}$$

Denotemos o vetor $[D_{xx}]\vec{a} = (2a_4, 6a_7, 2a_8, 12a_{11}, 6a_{12}, 2a_{13}, 20a_{16}, 12a_{17}, 6a_{18}, 2a_{19})$ por

$$\vec{b} = (b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}) \ .$$

Assim, a integral $\iint_{T_i} {u_{i_{xx}}}^2 \, dxdy = \iint_{T_i} (D_{xx}u_i)^2 \, dxdy$ é a integral do quadrado de um polinômio cúbico (o produto de dois polinômios cúbicos iguais), determinado pelos 10 coeficientes $b_1, ..., b_{10}$.

Diferentemente do caso da aproximação quadrática, as expressões (1), (2), (3) e (4) não apresentam apenas constantes, o que torna os cálculos de tais integrais mais complicados, pois ficam dependentes de cada triângulo T_i. Para simplificar os cálculo, foi feita uma mudança de coordenadas na integral, através da transformação F (2.9), para que a integral pudesse ser calculada no triângulo padrão T_0, utilizando o teorema (2.1.1).

A transformação F leva o polinômio cúbico $u_{i_{xx}}$, definido no plano xy e com coeficientes $\vec{b} = (b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10})$ em um polinômio cúbico nas variáveis $\xi\eta$ e com vetor de coeficientes $\vec{\beta} = (\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10})$.

Como vimos no início do capítulo, a relação entre as famílias de coeficientes $\beta_1, ..., \beta_{10}$, do polinômio no plano xy, e $b_1, ..., b_{10}$, é dada por $\vec{\beta} = D_i\vec{b}$. Assim, de (2.11) e utilizando o teorema (2.1.1) temos que

$$\begin{aligned}
&\iint_{T_i} {u_{i_{xx}}}^2 \, dxdx = \\
&= \iint_{T_0} J_i[\beta_1 + \beta_2\xi + \beta_3\eta + \beta_4\xi^2 + \beta_5\xi\eta + \beta_6\eta^2 + \beta_7\xi^3 + \beta_8\xi^2\eta + \beta_9\xi\eta^2 + \beta_{10}\eta^3]^2 \, d\xi d\eta = \\
&= J_i(\vec{\beta}^T P \vec{\beta}) \; . \qquad (2.32)
\end{aligned}$$

Ou seja, a integral pode ser escrita como uma forma quadrática dos coeficientes $\beta_1, ..., \beta_{10}$, onde P é uma matriz quadrada, de dimensão 10×10, simétrica, positiva definida e cujas entradas são obtidas por integrais de monômios do tipo $\iint_{T_i} \xi^p\eta^q \, d\xi d\eta$.

Então podemos escrever:

$$\iint_{T_i} {u_{i_{xx}}}^2 \, dxdy = J_i\vec{\beta}^T P\vec{\beta} = J_i(D_i\vec{b})^T P(D_i\vec{b}) = J_i\vec{b}^T(D_i^T P D_i)\vec{b} =$$

$$J_i(D_{xx}\vec{a})^T D^T P D(D_{xx}\vec{a}) = J_i\vec{a}^T(D_{xx}^T D^T P D D_{xx})\vec{a} =$$

$$J_i(C_i{}^{-1}\vec{u})^T {D_{xx}}^T D_i^T P D_i D_{xx}(C_i{}^{-1}\vec{u}) = J_i\vec{u}((C_i^{-1})^T {D_{xx}}^T D_i^T P D_i D_{xx} C_i^{-1})\vec{u}.$$

Tal integral é uma forma quadrática, representada pela matriz $((C_i^{-1})^T {D_{xx}}^T D_i^T P D_i D_{xx} C_i^{-1})$. Os produtos entre matrizes são coerentes: P e D são matrizes 10×10, logo, $D_i^T P D_i$ é matriz 10×10. D_{xx} é matriz 10×21 o que faz com que ${D_{xx}}^T D_i^T P D_i D_{xx}$ seja matriz 21×21, por último, C_i^{-1} é 21×21 e, portanto, a matriz da forma quadrática (1) tem dimensão 21×21. Ainda, tais produtos preservam a simetria de P. Para o cálculo das integrais (2), (3) e (4), o procedimento é análogo.

De uma maneira mais geral, podemos dizer que a integral do produto de dois polinômios em duas variáveis pode ser vista como uma forma bilinear (simétrica) dos coeficientes de cada um desses polinômios. No caso da integral (1) os dois fatores de tal produto foram o mesmo polinômio e por isso obtivemos diretamente uma forma quadrática. Já a integral (4), por exemplo, apresenta um produto de dois polinômios distintos. Nesse caso, passamos primeiramente por uma forma bilinear, mas em seguida, obtemos também uma forma quadrática, como será mostrado a seguir. Aqui, usaremos que *"Uma forma quadrática pode ser escrita como um polinômio homogêneo de grau 2 em suas variáveis. Ela pode ser ser representada matricialmente e tal representação é única se a matriz for simétrica."*

Notemos que apesar de uma matriz quadrada e simétrica definir uma única forma quadrática,

essa não é a única maneira de definir uma forma quadrática. A representação matricial não é única quando olhamos para todo o espaço das matrizes quadradas (é única apenas quando nos restringimos ao subespaço das matrizes quadradas e simétricas).

As matrizes P, D_i e C_i são as mesmas para todas as parcelas de (1) a (4), as únicas mudanças se dão na matriz da transformação derivação e nos vetores de coeficientes do plano $\xi\eta$ relativo à mudança de variáveis F.

Considerando $\vec{c} = D_{yy}\vec{a}$; $\vec{d} = D_{xy}\vec{a}$; $\vec{\gamma} = D_i\vec{c}$ e $\vec{\delta} = D_i\vec{d}$, temos:

Para (2) :

$$\iint\limits_{T_i} {u_{i_{yy}}}^2 \, dxdx = J_i\vec{\gamma}^T P\vec{\gamma} = J_i(D_i\vec{c})^T P(D_i\vec{c}) = J_i\vec{c}^T(D_i^T P D_i)\vec{c} = J_i(D_{yy}\vec{a})^T D_i^T P D_i(D_{yy}\vec{a}) =$$

$$= J_i\vec{a}^T(D_{yy}^T D_i^T P D_i D_{yy})\vec{a} = J_i(C_i^{-1}\vec{u})^T {D_{yy}}^T D_i^T P D_i D_{yy}(C_i^{-1}\vec{u}) = J_i\vec{u}({C_i^{-1}}^T {D_{yy}}^T D_i^T P D_i D_{yy} C_i^{-1})\vec{u} \tag{2.33}$$

Para (3):

$$\iint\limits_{T_i} {u_{i_{xy}}}^2 \, dxdx = J_i\vec{\delta}^T P\vec{\delta} = J_i(D_i\vec{d})^T P(D_i\vec{d}) = J_i\vec{d}^T(D_i^T P D_i)\vec{d} = J_i(D_{xy}\vec{a})^T D_i^T P D_i(D_{xy}\vec{a}) =$$

$$= J_i\vec{a}^T(D_{xy}^T D_i^T P D_i D_{xy})\vec{a} = J_i(C_i^{-1}\vec{u})^T {D_{xy}}^T D_i^T P D_i D_{xy}(C_i^{-1}\vec{u}) = J_i\vec{u}({C_i^{-1}}^T {D_{xy}}^T D_i^T P D_i D_{xy} C_i^{-1})\vec{u} \tag{2.34}$$

Para a integral (4) que apresenta o produto de dois polinômios cúbicos distintos, obtemos:

$$\iint\limits_{T_i} u_{i_{xx}} u_{i_{yy}} \, dxdx = J_i\vec{\beta}^T P\vec{\gamma} = J_i(D_i\vec{b})^T P(D_i\vec{c}) = J_i\vec{b}^T(D_i^T P D_i)\vec{c} \, .$$

Até esse ponto, temos uma forma bilinear simétrica, representada pela matriz $(D_i^T P D_i)$. Tal matriz é simétrica, pois P é simétrica e D_i é não singular. Ao substituirmos os vetores $\vec{b}$ e $\vec{c}$ pela relação que os define: $D_{xx}\vec{a}$ e $D_{yy}\vec{a}$, respectivamente, obtemos uma forma quadrática

$$J_i D_{xx}\vec{a}^T(D_i^T P D_i)D_{yy}\vec{a} = J_i\vec{a}{D_{xx}}^T D_i^T P D_i D_{yy}\vec{a} \, .$$

Notemos que a matriz que representa tal forma quadrática não é simétrica. A simetria foi perdida pois as matrizes ${D_{xx}}^T$ e D_{yy}, que foram multiplicadas à esquerda e a direita de $(D_i^T P D_i)$, respectivamente, são distintas. Entretanto, continuamos obtendo um polinômio homogêneo de grau dois nos coeficientes $(a_1, ..., a_{21})$ do polinômio de grau cinco e, portanto, temos ainda uma forma quadrática. Continuando, obtemos

$$\iint\limits_{T_i} u_{i_{xx}} u_{i_{yy}} \, dxdx = J_i\vec{\beta}^T P\vec{\gamma} = J_i(D_i\vec{b})^T P(D_i\vec{c}) = J_i\vec{b}^T(D_i^T P D_i)\vec{c} = J_i(D_{xx}\vec{a})^T D_i^T P D_i(D_{yy}\vec{a}) =$$

$$= J_i\vec{a}^T(D_{xx}^T D_i^T P D_i D_{yy})\vec{a} = J_i C_i^{-1}\vec{u}^T(D_{xx}^T D_i^T P D_i D_{yy})C_i^{-1}\vec{u} = J_i\vec{u}^T(C_i^{-1})^T D_{xx}^T D_i^T P D_i D_{yy} C_i^{-1}\vec{u} \tag{2.35}$$

Dessa forma, a integral (4) consiste numa forma quadrática, apesar de sua representação matricial (natural, i.e. intrínseca da montagem) $J_i[(C_i^{-1})^T D_{xx}^T D_i^T P D_i D_{yy} C_i^{-1}]$ não ser simétrica.

Como essa etapa é parte de um processo que visa construir uma matriz simétrica, vamos utilizar o fato de que

Teorema 2.3.1. *Dada uma forma quadrática, existe uma única matriz simétrica que a define. E dada uma matriz simétrica, existe uma única forma quadrática definida por ela, ou seja, existe um isomorfismo entre o espaço das formas quadráticas e das matrizes quadradas simétricas.*

E o fato de que

Teorema 2.3.2. *Dada uma matriz quadrada A, a matriz $B = \frac{1}{2}(A + A^T)$ é simétrica e define a mesma forma quadrática que A.*

Substituímos então a matriz não simétrica $({C^{-1}}^T D_{xx}^T D_i^T P D_i D_{yy} C^{-1})$, por sua simetrização:

$$\frac{1}{2}\{({C^{-1}}^T D_{xx}^T D_i^T P D_i D_{yy} C^{-1}) + ({C^{-1}}^T D_{xx}^T D_i^T P D_i D_{yy} C^{-1})^T\} = \frac{1}{2}\{({C^{-1}}^T D_{xx}^T D_i^T P D_i D_{yy} C^{-1}) + ({C^{-1}}^T D_{yy}^T D_i^T P D_i D_{xx} C^{-1})\}. \tag{2.36}$$

E então a forma quadrática $\iint\limits_{T_i} u_{i_{xx}} u_{i_{yy}}\, dxdx$ pode ser escrita como

$$\iint\limits_{T_i} u_{i_{xx}} u_{i_{yy}}\, dxdx = J_i \vec{u}^T \{\frac{1}{2}({C_i^{-1}}^T D_{xx}^T D_i^T P D_i D_{yy} C_i^{-1} + {C_i^{-1}}^T D_{yy}^T D_i^T P D_i D_{xx} C_i^{-1})\}\vec{u}.$$

Combinando as parcelas (1), (2), (3) e (4) com os devidos coeficientes a integral $\mathscr{I}^{(1)}$ fica assim:

$$\mathscr{I}^{(1)} = \iint\limits_{T_i} u_{i_{xx}}^2 + u_{i_{yy}}^2 + 2\nu u_{i_{xx}} u_{i_{yy}} + 2(1-\nu)u_{i_{xy}}^2\, dxdy = Ji\{\ [\ (1) + (2) + 2(1-\nu)(3) + 2\nu(4)\ ]\ \} =$$

$$J_i\{\vec{u}\ [\ (\ {C_i^{-1}}^T D_{xx}^T D_i^T P D_i D_{xx} C_i^{-1}\ ) + ({C_i^{-1}}^T {D_{yy}}^T D_i^T P D_i D_{yy} C_i^{-1}) +$$
$$2(1-\nu)({C_i^{-1}}^T {D_{xy}}^T D_i^T P D_i D_{xy} C_i^{-1}) + 2\nu({C_i^{-1}}^T {D_{xx}}^T D_i^T P D_i D_{yy} C_i^{-1})\ ]\ \vec{u}\}\ .$$

E, assim, fica definida a matriz de rigidez local:

$$S_i = J_i[(\ {C_i^{-1}}^T D_{xx}^T D^T P D D_{xx} C_i^{-1}\ ) + ({C_i^{-1}}^T {D_{yy}}^T D^T P D D_{yy} C_i^{-1}) + 2(1-\nu)({C_i^{-1}}^T {D_{xy}}^T D^T P D D_{xy} C_i^{-1}) + 2\nu({C_i^{-1}}^T {D_{xx}}^T D^T P D D_{yy} C_i^{-1})\ ] \tag{2.37}$$

Quanto à integral $\mathscr{I}^{(2)} = \iint\limits_{T_i} u_i^2\, dxdy$, estamos integrando o quadrado de um polinômio de grau 5, nas variáveis x e y. Para isso, usaremos novamente a mudança de coordenadas F usada na montagem das parcelas de $\mathscr{I}^{(1)}$. Com isso, a integral passa a ser calculada no triângulo padrão T_0 do plano $\xi\eta$. Usaremos as "mesmas" matrizes (em termos de estrutura e finalidade) D_i, P e C_i^{-1} usadas anteriormente, mas agora, com P e D_i de dimensão 21. Sendo $\vec{\alpha} = D_i\vec{a}$ e $\vec{a} = C_i^{-1}\vec{u_i}$, ficamos

com o seguinte:

$$\begin{aligned}\mathscr{I}^{(2)} &= \iint\limits_{T_i} u_i^2 \; dxdy = J_i \iint\limits_{T_0} \left(u_i(F^{-1}(\xi,\eta)\right)^2 \; d\xi d\eta = \\ &= J_i \; [\alpha_1 + \alpha_2\xi + \alpha_3\eta + \alpha_4\xi^2 + \alpha_5\xi\eta + \alpha_6\eta^2 + \alpha_7\xi^3 + \alpha_8\xi^2\eta + \alpha_9\xi\eta^2 + \alpha_{10}\eta^3 + \alpha_{11}\xi^4 + \alpha_{12}\xi^3\eta + \\ &\quad + \alpha_{13}\xi^2\eta^2 + \alpha_{14}\xi\eta^3 + \alpha_{15}\eta^4 + \alpha_{16}\xi^5 + \alpha_{17}\xi^4\eta + \alpha_{18}\xi^3\eta^2 + \alpha_{19}\xi^2\eta^3 + \alpha_{20}\xi\eta^4 + \alpha_{21}\eta^5 \;]^2 \; d\xi d\eta = \\ &= J_i\vec{\alpha}^T P\vec{\alpha} = J_i(D_i\vec{a})^T P D_i\vec{a} = J_i\vec{a}(D_i^T P D_i)\vec{a} = J_i(C_i^{-1}\vec{u})^T D_i^T P D_i C_i^{-1}\vec{u} = J_i\vec{u}(C_i^{-T} D_i^T P D_i C_i^{-1})\vec{u}.\end{aligned} \tag{2.38}$$

Assim, definimos $M_i = J_i(C_i^{-T} D_i^T P D_i C_i^{-1})$, que é a matriz de massa local. Esse produto de matrizes é coerente, pois todas as matrizes envolvidas têm dimensão 21. A simetria de P é mantida nesse produto e, se as matrizes D_i e C_i são não singulares, M_i é positiva definida como P. Isso é devido ao

Teorema 2.3.3. *Seja P uma matriz quadrada, de dimensão n, simétrica e positiva definida. Se Q é matriz de dimensão n, não singular, então o produto Q^TPQ é matriz positiva definida.*

Demonstração: Como P é positiva definida, então $\langle u, Pu\rangle > 0$, $\forall\ u \in \mathbb{R}^n,\ u \neq 0$.
Seja $v \in \mathbb{R}^n,\ v \neq 0$. Como Q é não singular, $Qv \neq 0$ e, portanto, $\langle Qv, PQv\rangle > 0$. Mas

$$0 < \langle Qv, PQv\rangle = \langle v, Q^TPQv\rangle,\ \forall\ v \neq 0$$

de onde concluímos que Q^TPQ é matriz positiva definida. □

Agora, utilizamos as matrizes de massa e rigidez locais para a montagem das matrizes de massa e rigidez globais. Novamente como na aproximação quadrática, para cada triângulo T_i da malha, a matriz global recebe na posição (i_m, i_n) o valor que está na posição (m, n) da matriz local, para $m, n \in \{1, 2, 3, 4, 5, 6\}$. Ou seja: $S_{i_m,i_n} = S_{i_m,i_n} + S^i{}_{mn}$ no caso da matriz de rigidez e $M_{i_m,i_n} = M_{i_m,i_n} + M^i{}_{mn}$ no caso da matriz de massa, onde i_m indica a numeração global do nó local m do triângulo T_i. Assim, cada triângulo local dá a sua contribuição na malha e portanto, cada nó incorpora as informações das variáveis de cada triângulo que nele incide.

Depois da montagem das matrizes globais a partir das matrizes locais, o último passo para finalizar o processo de montagem das matrizes que serão usadas no problema de autovalor, é contabilizar as condições de contorno do problema. A condição de *bordo grampeado* $u = \partial_n u = 0$, no problema de vibração de uma placa fina de metal tem como consequência o anulamento dos valores de u e de sua derivada normal $\partial_n u$ na fronteira $\partial\Omega$ do domínio Ω. Assim, para todos os nós da fronteira de Ω, o valor de u e $\partial_n u$ já são conhecidos e portanto o problema de autovalor não precisa ser resolvido para eles.

Nesse caso da aproximação de quinto grau, devemos considerar que, além dos valores de u e de $\partial_n u$, temos como variáveis nodais nos vértices do triângulo as derivadas de primeira e segunda ordem de u: u_x,u_y, u_{xx},u_{xy}, u_{yy}. Como o valor da função u deve se anular na fronteira, isso implica que a derivada de u na direção tangente é zero. Como $\partial_n u$ pode ser escrito em função dos valores de u_x e u_y e $\partial_n u = 0$ na fronteira, segue que $u_x = u_y = 0$ em $\partial\Omega$. Aplicar essas condições de contorno, consiste em eliminar das matrizes globais, as linhas e colunas relativas às variáveis u, u_x, u_y e $\partial_n u$ daqueles nós que pertencem à fronteira do domínio. Foi necessário tomar um certo cuidado

na implementação desta etapa, uma vez que num mesmo nó que é vértice de triângulo incidem seis variáveis nodais (e não apenas uma como no caso da aproximação quadrática) e dessas seis, apenas três foram eliminadas dos vértices de fronteira.

Observações sobre a matriz C_i

A matriz C_i, como na seção anterior, depende da geometria da malha e terá que ser invertida tantas vezes quanto o número de triângulos que a malha contém. Nesse caso da aproximação de quinto grau, em que temos muito mais variáveis, se comparada com a aproximação quadrática, essa inversão nos causou alguns problemas computacionais, já que trabalhamos com a inversão de matrizes muito grandes. Falaremos adiante, no Capítulo 3 sobre duas alternativas encontradas para tentar contornar esse problema.

Essa primeira abordagem descrita até aqui não nos forneceu resultados satisfatórios. O programa é "instável" sob a mudança de malhas, no sentido de que para certas malhas a matriz de massa numérica perde sua característica de ser positiva definida, condição necessária para que o problema de autovalores generalizado possa ser resolvido numericamente.

Capítulo 3

Tratamentos alternativos para a aproximação de quinto grau

3.1 Evitando as várias inversões da matriz C_i

Vimos que, na implementação do elemento conforme de quinto grau, a matriz C_i depende da geometria da malha, o que a torna diferente para cada triângulo. Com isso, ela deve ser montada e invertida tantas vezes quanto for o número de elementos da malha. Foi observado que numa mesma malha, a matriz C_i foi encontrada singular, e portanto não inversível, para vários elementos T_i's da malha, impossibilitando a solução numérica do problema. Acreditamos que isso possa ter acontecido devido ao refinamento da malha, que torna cada elemento triangular cada vez menor.

De modo geral, por construção, a matriz C_i fica dependente da malha quando a aproximação da função deslocamento $u(x, y)$ pelo polinômio u_i é feita num triângulo arbitrário no plano xy. Se essa aproximação é feita no triângulo padrão T_0 do no plano $\xi\eta$, a matriz C_i passa a ser independente da geometria da malha e portanto não precisa ser montada e invertida para cada triângulo. A matriz C_i só precisa ser calculada uma vez, o que nos permite chamar C_i apenas de C.

Uma ideia, então, foi estabelecer uma maneira de fazer a aproximação polinomial de quinto grau diretamente no triângulo padrão. Para isso, aplicaríamos a mudança de coordenadas F ao elemento finito T_i, de grau 5, levando-o, juntamente com suas variáveis nodais, para o triângulo padrão. A dificuldade de se fazer essa transformação é controlar o mapeamento das direções normais de T_i para T_0, uma vez que a direção normal fica distorcida por essa transformação, e cada distorção depende da geometria de cada triângulo T_i.

A fim de melhorar essa situação desenvolvemos uma maneira de realizar a aproximação polinomial no triângulo padrão, através da mudança de algumas das variáveis nodais que compõem o elemento finito triangular. Optamos por trocar a derivada normal do ponto médio das arestas por outras variáveis que independem da direção normal, de modo a não haver preocupação com o controle da direção do vetor normal sob a transformação F.

Queremos deixar claro aqui que a ideia não foi começar uma implementação totalmente nova, definindo um novo tipo de elemento finito, mas, foi aproveitar o elemento conforme de quinto grau e usar a nova organização de variáveis (o novo elemento) como mais uma etapa da implementação, uma etapa "facilitadora". Nessa etapa, o objetivo é reorganizar as novas variáveis a partir das originais, relacionando os dois conjuntos de variáveis nodais. Essa relação pôde ser estabelecida

através de uma matriz, que denotaremos por NC.

3.2 Reorganizando o elemento triangular conforme

Nesta seção definiremos o elemento triangular que foi obtido da substituição de algumas variáveis nodais, com o objetivo de fazer a aproximação no triângulo padrão T_0 do plano $\xi\eta$. Para diferenciar o novo conjunto de variáveis nodais escolhido, nos referenciaremos ao conjunto de variáveis nodais originais do elemento de quinto grau por *variáveis clássicas*.

As variáveis escolhidas para compor esse novo elemento foram as seguintes: nos vértices do triângulo, os valores da função u e de suas derivadas de segunda ordem u_{xx}, u_{yy} e u_{xy}, e, nos pontos médios das arestas os valores de u e suas derivadas de primeira ordem u_x e u_y. Os valores de u e suas derivadas de segunda ordem nos vértices também aparecem no conjunto das variáveis *clássicas*. A novidade são as derivadas de primeira ordem que passaram dos vértices para os pontos médios, substituindo as derivadas normais. Definimos uma notação para cada um desses conjuntos de variáveis que leva em conta o número de variáveis nodais nos vértices do triângulo e no ponto médio das suas arestas. O conjunto de variáveis clássico será do tipo 6-1, devido às seis variáveis nodais nos vértices e a única variável nodal nos pontos médios. O conjunto de variáveis novas, será do tipo 4-3 (quatro variáveis nodais nos vértices e três variáveis nodais nos pontos médios). Os vetores para cada conjunto de variáveis serão denotado por $\vec{u}_S$ e $\vec{u}_Q$, respectivamente ("S" de seis e "Q" de quatro).

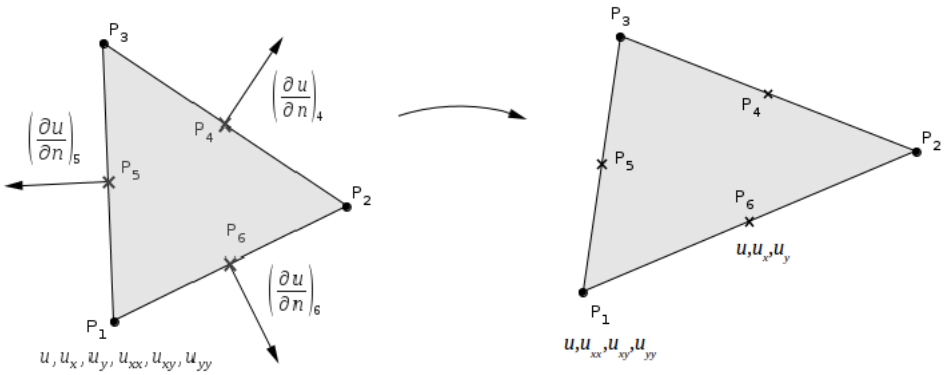


Figura 3.1: *Elemento finito de quinto grau. Organização dos conjunto de variáveis clássica: tipo 6-1 e conjunto de variáveis novas: tipo 4-3.*

Sendo assim, temos:

$$\vec{u}_Q = \left(u^1, u^2, u^3, u^4, u^5, u^6, u_x^4, u_x^5, u_x^6, u_y^4, u_y^5, u_y^6, u_{xx}^1, u_{xx}^2, u_{xx}^3, u_{xy}^1, u_{xy}^2, u_{xy}^3, u_{yy}^1, u_{yy}^2, u_{yy}^3\right)$$

e

$$\vec{u}_S = (u^1, u^2, u^3, u_x^1, u_x^2, u_x^3, u_y^1, u_y^2, u_y^3, u_{xx}^1, u_{xx}^2, u_{xx}^3, u_{xy}^1, u_{xy}^2, u_{xy}^3, u_{yy}^1, u_{yy}^2, u_{yy}^3, \partial_n u^4, \partial_n u^5, \partial_n {}^6) \ .$$

O índice acima de cada variável indica a numeração do nó do elemento onde esse valor será calculado. Para relacionar esses dois vetores, foi necessário escrever cada uma das novas variáveis nodais em função daquelas que aparecem no vetor de variáveis nodais clássicas. Mas como os valores

de u nos vértices do triângulo e as derivadas de segunda ordem aparecem em ambos vetores, essa variáveis se relacionam de forma trivial. O maior trabalho se dá na construção dessa relação para os valores de u nos pontos médios das arestas e para o valor das derivadas de primeira ordem u_x e u_y nesses pontos. Para isso utilizamos a técnica de interpolação com diferenças divididas.

A seguir apresentamos como foi estabelecida tal relação entre as *variáveis novas* e *clássicas*, bem como o processo de interpolação. Mais adiante, apresentamos como ficam as expressões dos funcionais $\mathscr{I}^{(1)}$ e $\mathscr{I}^2$ nessa nova formulação.

3.2.1 Procedimento para relacionar as variáveis *novas* e as *clássicas*:

O objetivo é escrever o valor de $u(x, y)$ e de suas derivadas de primeira ordem u_x e u_y, nos pontos médios das arestas, como combinação das *variáveis nodais clássicas*. Para isso, consideramos a função polinomial $u(x, y)$ restrita a uma aresta do triângulo, onde ela se torna um polinômio $p(s)$ na variável s.

Consideramos a parametrização de uma aresta com um parâmetro de comprimento de arco s:

$$t(s) = P_I + \frac{s}{\ell}(P_F - P_I) = (x(s), y(s)),$$

em que ℓ é o comprimento da aresta, $P_I = (x_I, y_I)$ e $P_F = (x_F, y_F)$ são seus pontos inicial e final, respectivamente. Assim, em cada aresta j, o vetor (unitário) na direção da aresta j é dado por:

$$\vec{s}_j = (a_j, b_j) = \left(\frac{x_{F_j} - x_{I_j}}{\ell}, \frac{y_{F_j} - y_{I_j}}{\ell} \right) = \frac{dt(s)}{ds} \ ,$$

em que $\ell = ||\vec{s}_j|| = \sqrt{(x_{F_j} - x_{I_j})^2 + (y_{F_j} - y_{I_j})^2}$.

Os pontos inicial e final das arestas locais são fixados de forma que a direção da aresta, s_j, ao ser girada por 90 graus no sentido anti-horário, fique na direção normal apontada para fora do triângulo. Foi dessa forma que o vetor normal foi obtido: através da rotação do vetor $\vec{s}_j$ por 90 graus no sentido anti-horário. Dessa forma, então definimos a direção do vetor normal (unitário) como:

$$\vec{n}_j = (n_{x_j}, n_{y_j}) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \cdot (a_j, b_j) = (-b_j, a_j)$$

.

As coordenas cartesianas do vetor normal $\vec{n}$ (ou do vetor aresta $\vec{s}$) nos fornece uma relação entre as derivadas nas direções cartesianas e as derivadas nas direções normal e tangente.

$$\partial_{\vec{s}} u = \nabla u \cdot \vec{s} = a_j u_x + b_j u_y$$

e

$$\partial_{\vec{n}} u = \nabla u \cdot \vec{n} = (-b_j) u_x + a_j u_y$$

Podemos escrever essa relação em notação matricial.

$$\begin{bmatrix} \partial_{\vec{s}} u \\ \partial_{\vec{n}} u \end{bmatrix} = A \cdot \begin{bmatrix} u_x \\ u_y \end{bmatrix} \qquad e \qquad \begin{bmatrix} u_x \\ u_y \end{bmatrix} = A^{-1} \cdot \begin{bmatrix} \partial_{\vec{s}} u \\ \partial_{\vec{n}} u \end{bmatrix} \ ,$$

em que

$$A = \begin{bmatrix} a_j & b_j \\ -b_j & a_j \end{bmatrix} \qquad e \qquad A^{-1} = \begin{bmatrix} a_j & -b_j \\ b_j & a_j \end{bmatrix} .$$

Seja $p(s) = u(t(s)) = u(x(s), y(s))$ a restrição de $u(x, y)$ a uma aresta j. Vamos usar esse polinômio para descrever o valor de $u(x, y)$ nos extremos e nos pontos médios das arestas. Denotaremos os valores de u e de suas derivadas calculados nos pontos inicial, final ou médio das arestas por índices superiores I, F e M, respectivamente.

Assim, obtemos as relações:

$$\begin{aligned} p(0) =& u(x_{I_j}, y_{I_j}) = u^I \\ p(\ell) =& u(x_{F_j}, y_{F_j}) = u^F \\ p(\ell/2) =& u(x_{M_j}, y_{M_j}) = u^M \end{aligned} \tag{3.1}$$

$$p'(s) = \frac{d}{ds}(u \circ t(s)) = \nabla u(x(s), y(s)) \cdot \frac{dt}{ds} = \nabla u \cdot \vec{s} = \frac{\partial u}{\partial \vec{s}}(x(s), y(s)) .$$

$$\begin{aligned} p'(0) =& \frac{\partial u}{\partial s}(x(0), y(0)) = \frac{\partial u}{\partial \vec{s}}(x_{I_j}, y_{I_j}) = u^I_s = a_j u^I_x + b_j u^I_y \\ p'(\ell/2) =& \frac{\partial u}{\partial s}(x(\ell/2), y(\ell/2)) = \frac{\partial u}{\partial \vec{s}}(x_{M_j}, y_{M_j}) = u^M_s = a_j u^M_x + b_j u^M_y \end{aligned} \tag{3.2}$$

Também vamos precisar expressar o valor de $p''(s)$.

$$p''(s) = \frac{d}{ds}(\frac{\partial u}{\partial \vec{s}}(x(s), y(s)) = \frac{d}{ds}\left[a_j \frac{\partial u}{\partial x}(x(s), y(s)) + b_j \frac{\partial u}{\partial y}(x(s), y(s))\right] ,$$

em que a_j e b_j são as coordenadas do vetor aresta $\vec{s}_j$. Sendo

$$G(x(s), y(s)) = \left[a_j \frac{\partial u}{\partial x}(x(s), y(s)) + b_j \frac{\partial u}{\partial y}(x(s), y(s))\right] ,$$

então

$$p''(s) = \nabla G \cdot \vec{s} = \left(\frac{\partial G}{\partial x}, \frac{\partial G}{\partial y}\right) \cdot (a_j, b_j) =$$

$$\left(a_j \frac{\partial^2 u}{\partial x^2}(x(s), y(s)) + b_j \frac{\partial^2 u}{\partial x \partial y}(x(s), y(s)), a_j \frac{\partial^2 u}{\partial x \partial y}(x(s), y(s)) + b_j \frac{\partial^2 u}{\partial y^2}(x(s), y(s))\right) \cdot (a_j, b_j) =$$

$${a_j}^2 \frac{\partial^2 u}{\partial x^2}(x(s), y(s)) + 2 b_j a_j \frac{\partial^2 u}{\partial x \partial y}(x(s), y(s)) + {b_j}^2 \frac{\partial^2 u}{\partial y^2}(x(s), y(s)) .$$

Assim, para uma aresta j, $(j = 1, 2, 3)$ temos:

$$\begin{aligned} p''(0) &= {a_j}^2 \frac{\partial^2 u}{\partial x^2}(x_{I_j}, y_{I_j}) + 2 b_j a_j \frac{\partial^2 u}{\partial x \partial y}(x_{I_j}, y_{I_j}) + {b_j}^2 \frac{\partial^2 u}{\partial y^2}(x_{I_j}, y_{I_j}) = {a_j}^2 u^I_{xx} + 2 b_j a_j u^I_{xy} + {b_j}^2 u^I_{yy} \\ p''(\ell) &= {a_j}^2 \frac{\partial^2 u}{\partial x^2}(x_{F_j}, y_{F_j}) + 2 b_j a_j \frac{\partial^2 u}{\partial x \partial y}(x_{F_j}, y_{F_j}) + {b_j}^2 \frac{\partial^2 u}{\partial y^2}(x_{F_j}, y_{F_j}) = {a_j}^2 u^F_{xx} + 2 b_j a_j u^F_{xy} + {b_j}^2 u^F_{yy} . \end{aligned} \tag{3.3}$$

Com essas expressões em mãos, escrevemos as novas variáveis nodais em função das variáveis nodais originais. Os valores de $u(x, y)$ nos vértices do elemento triangular serão mantidos. Precisamos expressar $u(x, y)$, $u_x(x, y)$ e $u_y(x, y)$ nos pontos médios de cada aresta, i.e. temos que escrever $u^k = u(x_{m_j}, y_{m_j})$, ${u_x}^k(x_{m_j}, y_{m_j})$ e ${u_y}^k(x_{m_j}, y_{m_j})$, em função das variáveis nodais clássicas, para $k = 4, 5, 6$ (k varia segundo a numeração local das arestas) e $j = 1, 2, 3$ (j varia no número de arestas por elemento triangular):

$$\begin{aligned} u(x_{m_j}, y_{m_j}) &= p(l/2) \\ u_x(x_{m_j}, y_{m_j}) &= a_j \frac{\partial u}{\partial \vec{s}}(x_m, y_m) + (-b_j)\frac{\partial u}{\partial \vec{n}}(x_m, y_m) = a_j p'(l/2) - b_j \frac{\partial u}{\partial \vec{n}}(x_m, y_m) \\ u_y(x_{m_j}, y_{m_j}) &= b_j \frac{\partial u}{\partial \vec{s}}(x_m, y_m) + a_j \frac{\partial u}{\partial \vec{n}}(x_m, y_m) = b_j p'(l/2) + a_j \frac{\partial u}{\partial \vec{n}} \end{aligned} \tag{3.4}$$

em que os valores de p e p', serão obtidas por interpolação, como veremos na Subseção 3.2.2.

3.2.2 Interpolação - uma expressão para $p(s)$ em função dos valores de p, p' e p'' nas extremidades das arestas

Nessa relação entre as variáveis modais novas e clássicas (originais), teremos que escrever variáveis nos pontos médios em função dos valores de u e suas primeiras derivadas nos vértices e vice-versa. Para isso, estamos usando o valor de $p(s)$ como um "passo intermediário" . Mas, segundo o que a variáveis nodais clássicas nos fornece, só é conhecido o valor de $p(s)$ nos extremos das arestas. Para que seja possível expressar $p(s)$ em qualquer ponto da aresta, faremos uso da interpolação polinomial, escrevendo o polinômio $p(s)$ em função dos seus valores nos pontos inicial e final de cada aresta. Esses pontos foram tomados como base, pois é neles que se encontram as variáveis nodais do problema.

A função $u(x, y)$ é polinômio de grau cinco em cada elemento triangular. Restrito a uma aresta j, tal polinômio se torna um polinômio também de grau 5, mas apenas na variável s. Assim, escrevemos $p(s)$ em função das expressões de $p(0)$, $p(\ell)$, $p'(0)$, $p'(\ell)$, $p''(0)$ e $p''(\ell)$, usando diferenças divididas.

$$p(s) = p(0) + p'(0)s + 1/2p''(0)s^2 + As^3 + B(s-l)s^3 + C(s-l)^2 s^3 \ ,$$

em que

$$\begin{aligned} A &= \frac{p(\ell) - p(0) - lp'(0) - \frac{1}{2}\ell^2 p''(0)}{l^3} \\ B &= \frac{3p(0) - 3p(\ell) + 2\ell p'(0) + \ell p'(\ell) + \frac{1}{2}\ell^2 p''(0)}{\ell^4} \\ C &= \frac{\frac{1}{2}p''(\ell)\ell^2 - \frac{1}{2}p''(0)\ell^2 - 3\ell p'(\ell) - 3\ell p'(0) + 6p(\ell) - 6p(0)}{\ell^5} \end{aligned} \tag{3.5}$$

Substituindo s por $l/2$ na equação acima obtemos:

$$p(l/2) = \frac{1}{2}p(0) + \frac{1}{2}p(\ell) + \frac{5\ell}{32}p'(0) + \frac{-5\ell}{32}p'(l) + \frac{\ell^2}{64}p''(0) + \frac{\ell^2}{64}p''(\ell).$$

Usando os valores de p, p' e p'' em 0 e em ℓ encontrados em (3.1), (3.2) e (3.3), obtemos:

$$\begin{aligned} p(l/2) &= \frac{1}{2}u^I + \frac{1}{2}u^F + \frac{5\ell}{32}(a_j u_x^I + b_j u_y^I) + \frac{-5\ell}{32}(a_j u_x^F + b_j u_y^F) + \\ &= \frac{\ell^2}{64}(u_{xx}^I {a_j}^2 + 2u_{xy}^I)b_j a_j + u_{yy}^I {b_j}^2) + \frac{\ell^2}{64}u_{xx}^F {a_j}^2 + 2u_{xy}^F b_j a_j + u_{yy}^F {b_j}^2). \end{aligned} \tag{3.6}$$

Vamos precisar escrever o valor de $p'(l/2)$ em função dos valores de p, p' e p'' nos pontos inicial e final, e portanto, em função dos valores de $u(x, y)$ nos vértices das arestas; $p'(l/2)$ é obtida derivando o polinômio interpolador $p(s)$ e calculando-o em $\ell/2$:

$$\begin{aligned} p(s) &= p(0) + p'(0)s + 1/2p''(0)s^2 + As^3 + B(s-l)s^3 + C(s-l)^2 s^3 \\ p'(s) &= p'(0) + p''(0)s + 3As^2 + B(s^3 + (s-l)3s^2) + C(2(s-l)s^3 + (s-l)^2 3s^2) \end{aligned} \tag{3.7}$$

de onde

$$\begin{aligned} p'(\ell/2) &= \frac{-30}{16\ell}p(0) + \frac{30}{16\ell}p(\ell) + \frac{-7}{16}p'(0) + \frac{-7}{16}p'(\ell) + \frac{-1}{32}p''(0) + \frac{1}{32}p''(\ell) = \\ &= \frac{-30}{16\ell}u^I + \frac{30}{16\ell}u^F + \frac{-7}{16}(a_j u_x^I + b_j u_y^I) + \frac{-7}{16}(a_j u_x^F + b_j u_y^F) + \\ &+ \frac{-1}{32}(u_{xx}^I {a_j}^2 + 2u_{xy}^I b_j a_j + u_{yy}^I {b_j}^2) + \frac{1}{32}(u_{xx}^F {a_j}^2 + 2u_{xy}^F b_j a_j + u_{yy}^F {b_j}^2)\ . \end{aligned} \tag{3.8}$$

Agora, escrevemos os valores de $u(x_M, y_M)$, $u_x(x_M, y_M)$ e $u_y(x_M, y_M)$ em função de $p(\ell/2)$ e de $p'(\ell/2)$:

$$\begin{aligned} u^M &= u(x_M, y_M) = p(l/2) = \frac{1}{2}p(0) + \frac{1}{2}p(\ell) + \frac{5\ell}{32}p'(0) + \frac{-5\ell}{32}p'(l) + \frac{\ell^2}{64}p''(0) + \frac{\ell^2}{64}p''(\ell) = \\ &= \frac{1}{2}u^I + \frac{1}{2}u^F + \frac{5\ell}{32}(a_j u_x^I + b_j u_y^I) + \frac{-5\ell}{32}(a_j u_x^F + b_j u_y^F) + \\ &+ \frac{\ell^2}{64}(u_{xx}^I {a_j}^2 + 2u_{xy}^I)b_j a_j + u_{yy}^I {b_j}^2) + \frac{\ell^2}{64}u_{xx}^F {a_j}^2 + 2u_{xy}^F b_j a_j + u_{yy}^F {b_j}^2)\ . \end{aligned} \tag{3.9}$$

$$\begin{aligned} u_x^M &= u_x(x_M, y_M) = a_j u_s^M + (-b_j)u_n^M = a_j p'(l/2) - b_j u_n^M = \\ &= a_j\left(\frac{-30}{16\ell}p(0) + \frac{30}{16\ell}p(\ell) + \frac{-7}{16}p'(0) + \frac{-7}{16}p'(\ell) + \frac{-1}{32}p''(0) + \frac{1}{32}p''(\ell)\right) - b_j u_n^M = \\ &= a_j\left(\frac{-30}{16\ell}u^I + \frac{30}{16\ell}u^F + \frac{-7}{16}(a_j u_x^I + b_j u_y^I) + \frac{-7}{16}(a_j u_x^F + b_j u_y^F) + \right. \\ &+ \left.\frac{-1}{32}(u_{xx}^I {a_j}^2 + 2u_{xy}^I b_j a_j + u_{yy}^I {b_j}^2) + \frac{1}{32}(u_{xx}^F {a_j}^2 + 2u_{xy}^F b_j a_j + u_{yy}^F {b_j}^2)\right) - b_j u_n^M. \end{aligned} \tag{3.10}$$

$$\begin{aligned} u_y^M &= u_y(x_M, y_M) = b_j u_s^M + a_j u_n^M = b_j p'(l/2) + a_j u_n^M = \\ &= b_j\left(\frac{-30}{16\ell}p(0) + \frac{30}{16\ell}p(\ell) + \frac{-7}{16}p'(0) + \frac{-7}{16}p'(\ell) + \frac{-1}{32}p''(0) + \frac{1}{32}p''(\ell)\right) + a_j u_n^M = \end{aligned} \tag{3.11}$$

$$\begin{aligned} &= b_j\left(\frac{-30}{16\ell}u^I + \frac{30}{16\ell}u^F + \frac{-7}{16}(a_j u_x^I + a_j u_y^I) + \frac{-7}{16}(a_j u_x^F + b_j u_y^F) + \right. \\ &+ \left.\frac{-1}{32}(u_{xx}^I {a_j}^2 + 2u_{xy}^I b_j a_j + u_{yy}^I {b_j}^2) + \frac{1}{32}(u_{xx}^F {a_j}^2 + 2u_{xy}^F b_j a_j + u_{yy}^F {b_j}^2)\right) + a_j u_n^M \end{aligned} \tag{3.12}$$

Agora já temos tudo o que precisamos para relacionar as variáveis novas com as originais. Essa relação será feita através da matriz NC, cuja estrutura descreveremos a seguir, da seguinte forma:

$$\vec{u}_Q = (NC)\vec{u}_S \ .$$

3.2.3 Estrutura da matriz NC

A matriz NC tem estrutura segundo o esquema abaixo. Possui muitas entradas nulas e dois blocos de identidade. Em algumas entradas, o sinal poderá mudar conforme o elemento triangular, por isso elas foram indicadas com o sinal $\pm$. O sinal será definido conforme for definido qual vértice de cada aresta assume o papel de vértice inicial e vértice final. É importante construir um método para fazer tal distinção.

Neste esquema, letras iguais representam valores (teoricamente/estruturalmente) iguais. O que eu quero dizer com isso é que, mesmo já tendo fixado um triângulo i, os valores de x_I, x_F, y_I e y_F ainda dependem de cada aresta j, $j = 1, 2, 3$ deste triângulo.

Os valores a seguir são que compõe a matriz NC, com $j = 1, 2, 3$, são:

$$a = \frac{5\ell}{32}a_j = \frac{5}{32}(x_{F_j} - x_{I_j}) \qquad b = \frac{5\ell}{32}b_j = \frac{5}{32}(y_{F_j} - y_{I_j})$$

$$c = \frac{\ell^2}{64}a_j^2 = \frac{l}{64}(x_{F_j} - x_{I_j})^2 \qquad d = \frac{\ell^2}{32}a_j b_j = \frac{l}{32}(x_{F_j} - x_{I_j})(y_{F_j} - y_{I_j})$$

$$e = \frac{\ell^2}{64}b_j^2 = \frac{l}{64}(y_{F_j} - y_{I_j})^2 \qquad f = \frac{30}{16\ell}a_j = \frac{30}{16\ell^2}(x_{F_j} - x_{I_j})$$

$$g = \frac{-7}{16}a_j^2 = \frac{-7}{16\ell^2}(x_{F_j} - x_{I_j})^2 \qquad h = \frac{-7}{16}a_j b_j = \frac{-7}{16\ell^2}(x_{F_j} - x_{I_j})(y_{F_j} - y_{I_j})$$

$$i = \frac{\ell}{32}a_j^3 = \frac{1}{32\ell^2}(x_{F_j} - x_{I_j})^3 \qquad j = \frac{\ell}{16}a_j^2 b_j = \frac{1}{16\ell^2}(x_{F_j} - x_{I_j})^2(y_{F_j} - y_{I_j})$$

$$k = \frac{\ell}{32}a_j b_j^2 = \frac{1}{32\ell^2}(x_{F_j} - x_{I_j})(y_{F_j} - y_{I_j})^2 \qquad l = \frac{30}{16\ell}b_j = \frac{30}{16\ell^2}(y_{F_j} - y_{I_j})$$

$$m = \frac{-7}{16}a_j b_j = \frac{-7}{16\ell^2}(x_{F_j} - x_{I_j})(y_{F_j} - y_{I_j}) \qquad n = \frac{-7}{16}b_j^2 = \frac{-7}{16\ell^2}(y_{F_j} - y_{I_j})^2$$

$$o = \frac{\ell}{32}b_j a_j^2 = \frac{1}{32\ell^2}(y_{F_j} - y_{I_j})(x_{F_j} - x_{I_j})^2 \qquad p = \frac{\ell}{16}a_j b_j^2 = \frac{1}{16\ell^2}(x_{F_j} - x_{I_j})(y_{F_j} - y_{I_j})^2$$

$$q = \frac{\ell}{32}b_j^3 = \frac{1}{32\ell^2}(y_{F_j} - y_{I_j})^3 \qquad r = -b_j = \frac{-(y_{F_j} - y_{I_j})}{\ell}$$

$$s = a_j = \frac{(x_{F_j} - x_{I_j})}{\ell}$$

Estrutura da matriz NC:

Id_{3x3}			0						0								0		
0	$\frac{1}{2}$	$\frac{1}{2}$	0	$\pm a$	$\pm a$	0	$\pm b$	$\pm b$	0	c	c	0	d	d	0	e	e			
$\frac{1}{2}$	0	$\frac{1}{2}$	$\pm a$	0	$\pm a$	$\pm b$	0	$\pm b$	c	0	c	d	0	d	e	0	e		0	
$\frac{1}{2}$	$\frac{1}{2}$	0	$\pm a$	$\pm a$	0	b	b	0	c	c	0	d	d	0	e	e	0			
0	$\pm f$	$\pm f$	0	g	g	0	h	h	0	$\pm i$	$\pm i$	0	$\pm j$	$\pm j$	0	$\pm k$	$\pm k$	r	0	0
$\pm f$	0	$\pm f$	g	0	g	h	0	h	$\pm i$	0	$\pm i$	$\pm j$	0	$\pm j$	$\pm k$	0	$\pm k$	0	r	0
$\pm f$	$\pm f$	0	g	g	0	h	h	0	$\pm i$	$\pm i$	0	$\pm j$	$\pm j$	0	$\pm k$	$\pm k$	0	0	0	r
0	$\pm l$	$\pm l$	0	m	m	0	n	n	0	$\pm o$	$\pm o$	0	$\pm p$	$\pm p$	0	$\pm q$	$\pm q$	s	0	0
$\pm l$	0	$\pm l$	m	0	m	n	0	n	$\pm o$	0	$\pm o$	$\pm p$	0	$\pm p$	$\pm q$	0	$\pm q$	0	s	0
$\pm l$	$\pm l$	0	m	m	0	n	n	0	$\pm o$	$\pm o$	0	$\pm p$	$\pm p$	0	$\pm q$	$\pm q$	0	0	0	s
0			0						Id_{9x9}								0		

3.2.4 Implementação do elemento triangular reorganizado

Depois de estabelecida a relação entre as variáveis do tipo 4-3 e as do tipo 6-1, aplicamos a mudança de variáveis F que leva o triângulo T_i no triângulo padrão T_0. A aproximação da função u por um polinômio de quinto grau u_i foi feita então no plano $\xi\eta$. Ou seja, a função $u(\xi,\eta)$ foi aproximada por um polinômio

$$u_i(\xi,\eta) = \alpha_1 + \alpha_2\xi + \alpha_3\eta + \alpha_4\xi^2 + \alpha_5\xi\eta + \alpha_6\eta^2 + \alpha_7\xi^3 + \alpha_8\xi^2\eta + \alpha_9\xi\eta^2 + \alpha_{10}\eta^3 + \alpha_{11}\xi^4 + \alpha_{12}\xi^3\eta + \\ + \alpha_{13}\xi^2\eta^2 + \alpha_{14}\xi\eta^3 + \alpha_{15}\eta^4 + \alpha_{16}\xi^5 + \alpha_{17}\xi^4\eta + \alpha_{18}\xi^3\eta^2 + \alpha_{19}\xi^2\eta^3 + \alpha_{20}\xi\eta^4 + \alpha_{21}\eta^5 \ . \tag{3.13}$$

Calculamos a partir daí as derivadas de segunda ordem que compõem o funcional, levando em conta que as variáveis x e y são dependentes de ξ e η e, portanto, a regra da cadeia deve ser utilizada. A partir daqui, suprimimos o índice inferior i do polinômio aproximador u_i, para não carregar a notação.

$$\begin{aligned} u_x &= u_\xi\xi_x + u_\eta\eta_x \\ u_y &= u_\xi\xi_y + u_\eta\eta_y \end{aligned} \tag{3.14}$$

$$
\begin{aligned}
u_{xx} &= (u_{\xi\xi}\xi_x + u_{\eta\xi}\eta_x)\xi_x + (u_{\xi\eta}\xi_x + u_{\eta\eta}\eta_x)\eta_x = u_{\xi\xi}{\xi_x}^2 + 2u_{\xi\eta}\xi_x\eta_x + u_{\xi\xi}{\eta_x}^2 \\
u_{yy} &= (u_{\xi\xi}\xi_y + u_{\eta\xi}\xi_y)\xi_y + (u_{\xi\eta}\xi_y + u_{\eta\eta}\eta_y)\eta_y = u_{\xi\xi}{\xi_y}^2 + 2u_{\xi\eta}\xi_y\eta_y + u_{\eta\eta}{\eta_y}^2 \\
u_{xy} &= (u_{\xi\xi}\xi_x + u_{\eta\xi}\eta_x)\xi_y + (u_{\xi\eta}\xi_x + u_{\eta\eta}\eta_x)\eta_y = u_{\xi\xi}\xi_y\xi_y + u_{\xi\eta}(\eta_x\xi_y + \eta_y\xi_x) + u_{\eta\eta}\eta_y\eta_x
\end{aligned} \tag{3.15}
$$

Então,

$$
\begin{aligned}
(u_{xx})^2 = (u_{\xi\xi}{\xi_x}^2 + 2u_{\xi\eta}\xi_x\eta_x + u_{\xi\xi}{\eta_x}^2)^2 = {u_{\xi\xi}}^2{\xi_x}^4 + {u_{\eta\eta}}^2{\eta_x}^4 + \\
2u_{\xi\xi}u_{\eta\eta}{\xi_x}^2{\eta_x}^2 + 4u_{\xi\xi}u_{\eta\eta}{\xi_x}^3\eta_x + 4u_{\xi\eta}u_{\eta\eta}{\eta_x}^3\xi_x + 4u_{\xi\eta}^2{\xi_x}^2{\eta_x}^2
\end{aligned}
$$

$$
\begin{aligned}
(u_{yy})^2 = (u_{\xi\xi}{\xi_y}^2 + 2u_{\xi\eta}\xi_y\eta_y + u_{\eta\eta}{\eta_y}^2)^2 = {u_{\xi\xi}}^2{\xi_y}^4 + {u_{\eta\eta}}^2{\eta_y}^4 + \\
2u_{\xi\xi}u_{\eta\eta}{\xi_y}^2{\eta_y}^2 + 4u_{\xi\xi}u_{\eta\eta}{\xi_y}^3\eta_y + 4u_{\xi\eta}u_{\eta\eta}{\eta_y}^3\xi_y + 4u_{\xi\eta}^2{\xi_y}^2{\eta_y}^2
\end{aligned}
$$

$$
\begin{aligned}
(u_{xy})^2 = (u_{\xi\xi}\xi_y\xi_y + u_{\xi\eta}(\eta_x\xi_y + \eta_y\xi_x) + u_{\eta\eta}\eta_y\eta_x)^2 = {u_{\xi\xi}}^2{\xi_x}^2{\xi_y}^2 + {u_{\eta\eta}}^2{\eta_y}^2{\eta_y}^2 + 2u_{\xi\xi}u_{\eta\eta}{\xi_y}^2\xi_x\xi_y\eta_y\eta_x + \\
2u_{\xi\xi}u_{\eta\eta}\xi_x\xi_y(\xi_y\eta_x + \xi_x\eta_y) + 2u_{\xi\eta}u_{\eta\eta}\eta_x\eta_y(\xi_y\eta_x + \eta_y\xi_x) + u_{\xi\eta}^2(\xi_y\eta_x + \eta_y\xi_x)^2
\end{aligned} \tag{3.16}
$$

$$
u_{xx}u_{yy} = (u_{\xi\xi}{\xi_x}^2 + 2u_{\xi\eta}\xi_x\eta_x + u_{\xi\xi}{\eta_x}^2)(u_{\xi\xi}{\xi_y}^2 + 2u_{\xi\eta}\xi_y\eta_y + u_{\eta\eta}{\eta_y}^2) = \tag{3.17}
$$

$$
{u_{\xi\xi}}^2{\xi_x}^2{\xi_y}^2 + {u_{\eta\eta}}^2{\eta_x}^2{\eta_y}^2 + u_{\xi\xi}u_{\eta\eta}({\xi_y}^2{\eta_x}^2 + {\eta_y}^2{\xi_x}^2) + 2u_{\xi\xi}u_{\xi\eta}({\xi_x}^2\xi_y\eta_y + {\xi_y}^2\xi_x\eta_x) + \tag{3.18}
$$

$$
2u_{\xi\eta}u_{\eta\eta}(\eta_x\xi_x{\eta_y}^2 + \eta_y{\eta_x}^2\xi_y) + 4u_{\xi\eta}^2\xi_y\xi_x\eta_x\eta_y \tag{3.19}
$$

Assim, substituindo as expressões de (3.15) no funcional e agrupando os fatores $u_{\xi\xi}^2$, $u_{\xi\eta}^2$, $u_{\eta\eta}^2$, $u_{\xi\xi}u_{\eta\eta}$, $u_{\xi\xi}u_{\xi\eta}$, $u_{\xi\eta}u_{\eta\eta}$, obtemos:

$$
\begin{aligned}
\mathscr{J}^{(1)} = \iint_{T_i} u_{xx}^2 + u_{yy}^2 + 2\nu u_{xx}u_{yy} + 2(1-\nu)u_{xy}^2 \; dxdy = \\
\iint_{T_0} J_i\Big\{{u_{\xi\xi}}^2\Big[{\xi_x}^4 + {\xi_y}^4 + 2\nu(\xi_x^2\xi_y^2) + 2(1-\nu)(\xi_x^2\xi_y^2)\Big] + u_{\eta\eta}^2\Big[{\eta_x}^4 + {\eta_y}^4 + 2(1-\nu)\eta_x^2\eta_y^2 + 2\nu\eta_x^2\eta_y^2\Big] + \\
u_{\xi\xi}u_{\eta\eta}\Big[2\xi_x^2\eta_x^2 + 2\xi_y^2\eta_y^2 + 4(1-\nu)2\xi_x\xi_y\eta_x\eta_y + 2\nu(\xi_y\eta_x\xi_x\eta_y)\Big] + \\
u_{\xi\xi}u_{\xi\eta}\Big[4\xi_x^3\eta_x + 4\xi_y^3\eta_y + 4(1-\nu)\xi_x\xi_y(\eta_x\xi_y + \eta_y\xi_x) + 4\nu(\xi_x^2\xi_y\eta_y + \xi_y^2\xi_x\eta_x)\Big] + \\
u_{\eta\eta}u_{\xi\eta}\Big[4\eta_x^3\xi_x + 4\eta_y^3\xi_y + 4(1-\nu)\eta_x\eta_y(\eta_x\xi_y + \eta_y\xi_x) + 4\nu(\eta_y^2\xi_x\eta_x + \eta_x^2\xi_y\eta_y)\Big] + \\
u_{\xi\eta}^2\Big[4\eta_x^2\xi_x^2 + 4\eta_y^2\xi_y^2 + 2(1-\nu)(\eta_x\xi_y + \eta_y\xi_x)^2 + 8\nu\eta_x\eta_y\xi_x\xi_y\Big]\Big\} \; d\xi d\eta = \\
\iint_{T_0} f_1u_{\xi\xi}^2 + f_2u_{\xi\xi}u_{\xi\eta} + f_3u_{\xi\xi}u_{\eta\eta} + f_4u_{\xi\eta}^2 + f_5u_{\xi\eta}u_{\eta\eta} + f_6u_{\eta\eta}^2 \; d\xi d\eta.
\end{aligned} \tag{3.20}
$$

Em que $f_1, ..., f_6$, são dados por:

$$
\begin{aligned}
f_1 &= J_i(\xi_x^2 + \xi_y^2)^2 & f_4 &= J_i[4(\xi_x\eta_x + \xi_y\eta_y)^2 + 2(1-\nu)(\xi_y\eta_x - \xi_x\eta_y)^2 \\
f_2 &= 4J_i(\xi_x^2 + \xi_y^2)(\xi_x\eta_x + \xi_y\eta_y) & f_5 &= 4J_i(\xi_x^2 + \xi_y^2)(\xi_x\eta_x + \xi_y\eta_y) \\
f_3 &= 2J_i(\xi_x\eta_x + \xi_y\eta_y)^2 + \nu(\xi_y\eta_x - \xi_x\eta_y)^2 & f_6 &= J_i(\eta_x^2 + \eta_y^2)^2
\end{aligned}
$$

em que $J_i = \overline{x}_2\overline{y}_3 - \overline{x}_3\overline{y}_2$ é o determinante da matriz jacobiana da mudança de variáveis F^{-1} (2.10) e $\overline{x}_i = x_i - x_1$ para $j = 2, 3$.

Considerando a transformação $F : T_0 \longrightarrow T_i$, obtemos as expressões para $f_1, f_2, .., f_6$ em função das coordenadas do triângulo T_i.

$$f_1 = \frac{\left(\overline{y}_3^2 + \overline{x}_3^2\right)^2}{J_i^3} \qquad f_4 = \frac{4[(\overline{y}_3\overline{y}_2 + \overline{x}_3\overline{x}_2)^2] + 2(1-\nu)(\overline{x}_3\overline{y}_2 - \overline{y}_3\overline{x}_2)^2}{J_i^3}$$

$$f_2 = \frac{-4(\overline{y}_3^2 + \overline{x}_3^2)(\overline{y}_3\overline{y}_2 + \overline{x}_3\overline{x}_2)}{J_i^3} \qquad f_5 = \frac{-4(\overline{y}_2^2 + \overline{x}_2^2)(\overline{y}_3\overline{y}_2 + \overline{x}_3\overline{x}_2)}{J_i^3}$$

$$f_3 = \frac{2(\overline{y}_3\overline{y}_2 + \overline{x}_3\overline{x}_2)^2 + \nu(\overline{x}_3\overline{y}_2 - \overline{y}_3\overline{x}_2)^2}{J_i^3} \qquad f_6 = \frac{\left(\overline{y}_2^2 + \overline{x}_2^2\right)^2}{J_i^3}$$

Notemos que devido à mudança de coordenadas que nos levou ao triângulo padrão T_0, o número de parcelas do funcional aumentou (devido à regra da cadeia). A integração foi, portanto, feita diretamente em T_0, o que dispensou a mudança de variáveis na integral, como nos casos anteriores. Assim, não há duas famílias de coeficientes (uma do plano xy e outra do $\xi\eta$) a serem relacionadas, o que tornou a matriz D_i desnecessária nesse processo. Entretanto, é necessário relacionar o novo (i.e., depois da reorganização do elemento) conjunto de variáveis nodais do triângulo padrão T_0 : $\tilde{u}, \tilde{u}_\xi, \tilde{u}_\eta, \tilde{u}_{\xi\xi}, \tilde{u}_{\xi\eta}, \tilde{u}_{\eta\eta}$ com aquele do triângulo arbitrário $T_i : u, u_x, u_y, u_{xx}, u_{xy}, u_{yy}$.

Consideramos a definição $\tilde{u}(\xi, \eta) := u \circ F(\xi, \eta)$, em que F é a mudança de coordenadas (2.9) que leva T_i em T_0. A partir daí, estabelecemos a relação entre as variáveis nodais do plano xy e as do plano $\xi\eta$.

A relação entre as derivadas são obtidas das equações de (3.14) e (3.15). Ela pode ser escrita matricialmente: $\vec{\tilde{u}} = \tilde{D}_i\vec{u}$, em que

$$\vec{\tilde{u}} = (\tilde{u}^1, \tilde{u}^2, \tilde{u}^3, \tilde{u}^4, \tilde{u}^5, \tilde{u}^6, \tilde{u}^4_\xi, \tilde{u}^5_\xi, \tilde{u}^6_\xi, \tilde{u}^4_\eta, \tilde{u}^5_\eta, \tilde{u}^6_\eta, \tilde{u}^1_{\xi\xi}, \tilde{u}^2_{\xi\xi}, \tilde{u}^3_{\xi\xi}, \tilde{u}^1_{\xi\eta}, \tilde{u}^2_{\xi\eta}, \tilde{u}^3_{\xi\eta}, \tilde{u}^1_{\eta\eta}, \tilde{u}^2_{\eta\eta}, \tilde{u}^3_{\eta\eta}) \ ,$$

$$\vec{u} = (u^1, u^2, u^3, u^4, u^5, u^6, u^4_x, u^5_x, u^6_x, u^4_y, u^5_y, u^6_y, u^1_{xx}, u^2_{xx}, u^3_{xx}, u^1_{xy}, u^2_{xy}, u^3_{xy}, u^1_{yy}, u^2_{yy} u^3_{yy}) \ ,$$

e $\tilde{D}_{21x21}$ é

$$
\left[\begin{array}{c|c|c}
\begin{array}{cccccc}
1 & & & & & \\
& 1 & & & & \\
& & 1 & & & \\
& & & 1 & & \\
& & & & 1 & \\
& & & & & 1
\end{array} & & \\
\hline
 & \begin{array}{cccccc}
\overline{x}_2 & & & \overline{y}_2 & & \\
& \overline{x}_2 & & & \overline{y}_2 & \\
& & \overline{x}_2 & & & \overline{y}_2 \\
\overline{x}_3 & & & \overline{y}_3 & & \\
& \overline{x}_3 & & & \overline{y}_3 & \\
& & \overline{x}_3 & & & \overline{y}_3
\end{array} & \\
\hline
 & & \begin{array}{cccccccccc}
{\overline{x}_2}^2 & & & 2\overline{y}_2\overline{x}_2 & & & {\overline{y}_2}^2 & & & \\
& {\overline{x}_2}^2 & & & 2\overline{y}_2\overline{x}_2 & & & {\overline{y}_2}^2 & & \\
& & {\overline{x}_2}^2 & & & 2\overline{y}_2\overline{x}_2 & & & {\overline{y}_2}^2 & \\
2\overline{x}_2\overline{x}_3 & & & \overline{y}_2\overline{x}_3+\overline{y}_3\overline{x}_2 & & & 2\overline{y}_2\overline{y}_3 & & & \\
& 2\overline{x}_2\overline{x}_3 & & & \overline{y}_2\overline{x}_3+\overline{y}_3\overline{x}_2 & & & 2\overline{y}_2\overline{y}_3 & & \\
& & 2\overline{x}_2\overline{x}_3 & & & \overline{y}_2\overline{x}_3+\overline{y}_3\overline{x}_2 & & & 2\overline{y}_2\overline{y}_3 & \\
{\overline{x}_3}^2 & & & 2\overline{y}_3\overline{x}_3 & & & {\overline{y}_3}^2 & & & \\
& {\overline{x}_3}^2 & & & 2\overline{y}_3\overline{x}_3 & & & {\overline{y}_3}^2 & & \\
& & {\overline{x}_3}^2 & & & 2\overline{y}_3\overline{x}_3 & & & {\overline{y}_3}^2 &
\end{array}
\end{array}\right].
$$

Para concluir, vamos considerar as relações $\vec{b} = D_{\xi\xi}\vec{\alpha}$, $\vec{c} = D_{\eta\eta}\vec{\alpha}$, $\vec{d} = D_{\xi\eta}\vec{\alpha}$, que envolvem matrizes de derivação, e as relações entre os vetores de variáveis de cada etapa:

$$\vec{\alpha} = C^{-1}\vec{\tilde{u}}_Q \ , \ \vec{\tilde{u}}_Q = \tilde{D}\vec{u}_Q \text{ e } \vec{u}_Q = NC\vec{u}_S \ .$$

Olhemos para cada parcela da integral $\mathscr{I}^{(1)}$ (3.20) separadamente:

$$
\begin{aligned}
\iint\limits_{T_0} u_{\xi\xi}^2 \, d\xi d\eta = \iint\limits_{T_0} J_i[\ b_1 + b_2\xi + b_3\eta + b_4\xi^2 + b_5\xi\eta + b_6\eta^2 + b_7\xi^3 + b_8\xi^2\eta + b_9\xi\eta^2 + b_{10}\eta^3\ ]^2 \, d\xi d\eta = J_i \ \vec{b}^T P\vec{b} \\
= J_i \ [(D_{\xi\xi}\vec{\alpha})^T P(D_{\xi\xi}\vec{\alpha})] = J_i \ [\vec{\alpha}^T {D_{\xi\xi}}^T P D_{\xi\xi}\vec{\alpha}] = J_i \ [((C^{-1})\vec{\tilde{u}}_Q)^T {D_{\xi\xi}}^T P D_{\xi\xi}(C^{-1}\vec{\tilde{u}}_Q)] \\
= J_i \ [\vec{\tilde{u}}_Q^T (C^{-1})^T {D_{\xi\xi}}^T P D_{\xi\xi}(C^{-1}\vec{\tilde{u}}_Q)] = J_i \ [(\tilde{D}\vec{u}_Q)^T (C^{-1})^T {D_{\xi\xi}}^T P D_{\xi\xi}C^{-1}(\tilde{D}\vec{u}_Q)] \\
= J_i \ [\vec{u}_Q^T \tilde{D}^T (C^{-1})^T {D_{\xi\xi}}^T P D_{\xi\xi}C^{-1}\tilde{D}\vec{u}_Q] = J_i \ [((NC)\vec{u}_S)^T \tilde{D}^T (C^{-1})^T {D_{\xi\xi}}^T P D_{\xi\xi}C^{-1}\tilde{D}(NC)\vec{u}_S] \\
= J_i \ [\vec{u}_S^T NC^T \tilde{D}^T (C^{-1})^T {D_{\xi\xi}}^T P D_{\xi\xi}C^{-1}\tilde{D}NC\vec{u}_S].
\end{aligned}
\tag{3.21}
$$

$$\iint_{T_0} u_{\eta\eta}^2 \, d\xi d\eta = \iint_{T_0} J_i[\ c_1 + c_2\xi + c_3\eta + c_4\xi^2 + c_5\xi\eta + c_6\eta^2 + c_7\xi^3 + c_8\xi^2\eta + c_9\xi\eta^2 + c_{10}\eta^3\ ]^2 \, d\xi d\eta = J_i \ \vec{c}^T P \vec{c} =$$

$$= J_i \ [(D_{\eta\eta}\vec{\alpha})^T P (D_{\eta\eta}\vec{\alpha})] = J_i \ [\vec{\alpha}^T {D_{\eta\eta}}^T P D_{\eta\eta}\vec{\alpha}] = J_i \ [((C^{-1})\vec{\tilde{u}}_Q)^T {D_{\eta\eta}}^T P D_{\eta\eta}(C^{-1}\vec{\tilde{u}}_Q)]$$

$$= J_i \ [\vec{\tilde{u}}_Q^T (C^{-1})^T {D_{\eta\eta}}^T P D_{\eta\eta}(C^{-1}\vec{\tilde{u}}_Q)] = J_i \ [(\tilde{D}\vec{u}_Q)^T (C^{-1})^T {D_{\eta\eta}}^T P D_{\eta\eta} C^{-1}(\tilde{D}\vec{u}_Q)] =$$

$$= J_i \ [\vec{u}_Q^T \tilde{D}^T (C^{-1})^T {D_{\eta\eta}}^T P D_{\eta\eta} C^{-1} \tilde{D}\vec{u}_Q] = J_i \ [((NC)\vec{u}_S)^T \tilde{D}^T (C^{-1})^T {D_{\eta\eta}}^T P D_{\eta\eta} C^{-1} \tilde{D}(NC)\vec{u}_S]$$

$$= J_i \ [\vec{u}_S^T NC^T \tilde{D}^T (C^{-1})^T {D_{\eta\eta}}^T P D_{\eta\eta} C^{-1} \tilde{D} NC \vec{u}_S]. \tag{3.22}$$

$$\iint_{T_0} u_{\xi\eta}^2 \, d\xi d\eta = \iint_{T_0} J_i[\ d_1 + d_2\xi + d_3\eta + d_4\xi^2 + d_5\xi\eta + d_6\eta^2 + d_7\xi^3 + d_8\xi^2\eta + d_9\xi\eta^2 + d_{10}\eta^3\ ]^2 \, d\xi d\eta = J_i \ \vec{d}^T P \vec{d} =$$

$$= J_i \ [(D_{\xi\eta}\vec{\alpha})^T P (D_{\xi\eta}\vec{\alpha})] = J_i \ [\vec{\alpha}^T {D_{\xi\eta}}^T P D_{\xi\eta}\vec{\alpha}] = J_i \ [((C^{-1})\vec{\tilde{u}}_Q)^T {D_{\xi\eta}}^T P D_{\xi\eta}(C^{-1}\vec{\tilde{u}}_Q)]$$

$$= J_i \ [\vec{\tilde{u}}_Q^T (C^{-1})^T {D_{\xi\eta}}^T P D_{\xi\eta}(C^{-1}\vec{\tilde{u}}_Q)] = J_i \ [(\tilde{D}\vec{u}_Q)^T (C^{-1})^T {D_{\xi\eta}}^T P D_{\xi\eta} C^{-1}(\tilde{D}\vec{u}_Q)] =$$

$$= J_i \ [\vec{u}_Q^T \tilde{D}^T (C^{-1})^T {D_{\xi\eta}}^T P D_{\xi\eta} C^{-1} \tilde{D}\vec{u}_Q] = J_i \ [((NC)\vec{u}_S)^T \tilde{D}^T (C^{-1})^T {D_{\xi\eta}}^T P D_{\xi\eta} C^{-1} \tilde{D}(NC)\vec{u}_S]$$

$$= J_i \ [\vec{u}_S^T NC^T \tilde{D}^T (C^{-1})^T {D_{\xi\eta}}^T P D_{\xi\eta} C^{-1} \tilde{D} NC \vec{u}_S]. \tag{3.23}$$

Para as parcelas que contêm produtos de polinômios cúbicos, o processo é análogo ao realizado na seção (2.3). Obteremos formas quadráticas, representadas por uma matriz G não simétrica e que será substituída por sua simetrização $\frac{1}{2}(G^T + G)$.

$$\iint_{T_0} u_{\xi\xi} u_{\xi\eta} \, d\xi d\eta = \iint_{T_0} J_i[\ b_1 + b_2\xi + b_3\eta + b_4\xi^2 + b_5\xi\eta + b_6\eta^2 + b_7\xi^3 + b_8\xi^2\eta + b_9\xi\eta^2 + b_{10}\eta^3\ ] \cdot$$

$$[\ d_1 + d_2\xi + d_3\eta + d_4\xi^2 + d_5\xi\eta + d_6\eta^2 + d_7\xi^3 + d_8\xi^2\eta + d_9\xi\eta^2 + d_{10}\eta^3\ ] \, d\xi d\eta =$$

$$= J_i \ \vec{b}^T P \vec{d} = J_i \ [(D_{\xi\xi}\vec{\alpha})^T P (D_{\xi\eta}\vec{\alpha})] = J_i \ [\vec{\alpha}^T {D_{\xi\xi}}^T P D_{\xi\eta}\vec{\alpha}\ ] =$$

$$= J_i \ [((C^{-1})\vec{\tilde{u}}_Q)^T {D_{\xi\xi}}^T P D_{\xi\eta}(C^{-1}\vec{\tilde{u}}_Q)] = J_i \ [\vec{\tilde{u}}_Q^T (C^{-1})^T {D_{\xi\xi}}^T P D_{\xi\eta}(C^{-1})\vec{\tilde{u}}_Q] =$$

$$= J_i \ [(\tilde{D}\vec{u}_Q)^T (C^{-1})^T {D_{\xi\xi}}^T P D_{\xi\eta} C^{-1}(\tilde{D}\vec{u}_Q)] = J_i \ [\vec{u}_Q^T \tilde{D}^T (C^{-1})^T {D_{\xi\xi}}^T P D_{\xi\eta} C^{-1} \tilde{D}\vec{u}_Q] =$$

$$= J_i \ [((NC)\vec{u}_S)^T \tilde{D}^T (C^{-1})^T {D_{\xi\xi}}^T P D_{\xi\eta} C^{-1} \tilde{D}(NC)\vec{u}_S]$$

$$= J_i \ [\vec{u}_S^T \underbrace{[NC^T \tilde{D}^T (C^{-1})^T {D_{\xi\xi}}^T P D_{\xi\eta} C^{-1} \tilde{D} NC]}_{G} \vec{u}_S]. \tag{3.24}$$

$$\iint_{T_0} u_{\eta\eta}u_{\xi\eta}\ d\xi d\eta = \iint_{T_0} J_i[\ c_1 + c_2\xi + c_3\eta + c_4\xi^2 + c_5\xi\eta + c_6\eta^2 + c_7\xi^3 + c_8\xi^2\eta + c_9\xi\eta^2 + c_{10}\eta^3\ ]\cdot$$
$$[\ d_1 + d_2\xi + d_3\eta + d_4\xi^2 + d_5\xi\eta + d_6\eta^2 + d_7\xi^3 + d_8\xi^2\eta + d_9\xi\eta^2 + d_{10}\eta^3\ ]\ d\xi d\eta =$$
$$= J_i\ \vec{c}^T P\vec{d} = J_i\ [(D_{\eta\eta}\vec{\alpha})^T P(D_{\xi\eta}\vec{\alpha})] = J_i\ [\vec{\alpha}^T {D_{\eta\eta}}^T P D_{\xi\eta}\vec{\alpha}\ ] =$$
$$J_i\ [((C^{-1})\vec{\tilde{u}}_Q)^T {D_{\eta\eta}}^T P D_{\xi\eta}(C^{-1}\vec{\tilde{u}}_Q)] = J_i\ [\vec{\tilde{u}}_Q^T (C^{-1})^T {D_{\eta\eta}}^T P D_{\xi\eta}(C^{-1})\vec{\tilde{u}}_Q] =$$
$$= J_i\ [(\tilde{D}\vec{u}_Q)^T (C^{-1})^T {D_{\eta\eta}}^T P D_{\xi\eta} C^{-1}(\tilde{D}\vec{u}_Q)] = J_i\ [\vec{u}_Q^T \tilde{D}^T (C^{-1})^T {D_{\eta\eta}}^T P D_{\xi\eta} C^{-1}\tilde{D}\vec{u}_Q] =$$
$$= J_i\ [((NC)\vec{u}_S)^T \tilde{D}^T (C^{-1})^T {D_{\eta\eta}}^T P D_{\xi\eta} C^{-1}\tilde{D}(NC)\vec{u}_S]$$
$$= J_i\ [\vec{u}_S^T \underbrace{[NC^T\tilde{D}^T(C^{-1})^T {D_{\eta\eta}}^T P D_{\xi\eta}C^{-1}\tilde{D}NC]}_{G}\vec{u}_S]. \quad (3.25)$$

$$\iint_{T_0} u_{\xi\xi}u_{\eta\eta}\ d\xi d\eta = \iint_{T_0} J_i[\ b_1 + b_2\xi + b_3\eta + b_4\xi^2 + b_5\xi\eta + b_6\eta^2 + b_7\xi^3 + b_8\xi^2\eta + b_9\xi\eta^2 + b_{10}\eta^3\ ]\cdot$$
$$[\ c_1 + c_2\xi + c_3\eta + c_4\xi^2 + c_5\xi\eta + c_6\eta^2 + c_7\xi^3 + c_8\xi^2\eta + c_9\xi\eta^2 + c_{10}\eta^3\ ]\ d\xi d\eta =$$
$$= J_i\ \vec{b}^T P\vec{c} = J_i\ [(D_{\xi\xi}\vec{\alpha})^T P(D_{\eta\eta}\vec{\alpha})] = J_i\ [\vec{\alpha}^T {D_{\xi\xi}}^T P D_{\eta\eta}\vec{\alpha}\ ] =$$
$$J_i\ [((C^{-1})\vec{\tilde{u}}_Q)^T {D_{\xi\xi}}^T P D_{\eta\eta}(C^{-1}\vec{\tilde{u}}_Q)] = J_i\ [\vec{\tilde{u}}_Q^T (C^{-1})^T {D_{\xi\xi}}^T P D_{\eta\eta}(C^{-1})\vec{\tilde{u}}_Q] =$$
$$= J_i\ [(\tilde{D}\vec{u}_Q)^T (C^{-1})^T {D_{\xi\xi}}^T P D_{\eta\eta} C^{-1}(\tilde{D}\vec{u}_Q)] = J_i\ [\vec{u}_Q^T \tilde{D}^T (C^{-1})^T {D_{\xi\xi}}^T P D_{\eta\eta} C^{-1}\tilde{D}\vec{u}_Q] =$$
$$= J_i\ [((NC)\vec{u}_S)^T \tilde{D}^T (C^{-1})^T {D_{\xi\xi}}^T P D_{\eta\eta} C^{-1}\tilde{D}(NC)\vec{u}_S]$$
$$= J_i\ [\vec{u}_S^T \underbrace{[NC^T\tilde{D}^T(C^{-1})^T {D_{\xi\xi}}^T P D_{\eta\eta}C^{-1}\tilde{D}NC]}_{G}\vec{u}_S]. \quad (3.26)$$

Combinando as seis parcelas (3.21) a (3.26) com os devidos coeficientes $f_1, ..., f_6$, obtemos a matriz de rigidez S_i, relativa à forma quadrática de

$$\mathscr{I}^{(1)} = \iint_{T_i} u_{xx}^2 + u_{yy}^2 + 2\nu u_{xx}u_{yy} + 2(1-\nu)u_{xy}^2\ dxdy =$$
$$\iint_{T_i} f_1 u_{\xi\xi}^2 + f_2 u_{\xi\xi}u_{\xi\eta} + f_3 u_{\xi\xi}u_{\eta\eta} + f_4 u_{\xi\eta}^2 + f_5 u_{\xi\eta}u_{\eta\eta} + f_6 u_{\eta\eta}^2\ d\xi d\eta = ... = \ u^T S_i u. \quad (3.27)$$

em que S_i é a matriz de rigidez local.

Quanto à integral $\mathscr{I}^{(2)}$, temos que

$$\mathscr{I}^{(2)} = \iint_{T_i} u_i^2\ dxdy = J_i \iint_{T_0} \left(u_i(F^{-1}(\xi,\eta)\right)^2\ d\xi d\eta =$$
$$= \ J_i\ [\alpha_1 + \alpha_2\xi + \alpha_3\eta + \alpha_4\xi^2 + \alpha_5\xi\eta + \alpha_6\eta^2 + \alpha_7\xi^3 + \alpha_8\xi^2\eta + \alpha_9\xi\eta^2 + \alpha_{10}\eta^3 + \alpha_{11}\xi^4 + \alpha_{12}\xi^3\eta +$$
$$+\alpha_{13}\xi^2\eta^2 + \alpha_{14}\xi\eta^3 + \alpha_{15}\eta^4 + \alpha_{16}\xi^5 + \alpha_{17}\xi^4\eta + \alpha_{18}\xi^3\eta^2 + \alpha_{19}\xi^2\eta^3 + \alpha_{20}\xi\eta^4 + \alpha_{21}\eta^5\ ]^2\ d\xi d\eta =$$
$$= \ J_i\vec{\alpha}^T P\vec{\alpha} = J_i\vec{\tilde{u}}_Q^T (C^{-1})^T P(C^{-1})\vec{\tilde{u}}_Q = J_i\vec{u}_Q^T\tilde{D}^T(C^{-1})^T P(C^{-1})\tilde{D}\vec{u}_Q =$$
$$J_i\vec{u}_S^T(NC)^T\tilde{D}^T(C^{-1})^T P(C^{-1})\tilde{D}(NC)\vec{u}_S\ . \quad (3.28)$$

E assim definimos $M_i = (NC)^T\tilde{D}^T(C^{-1})^T P(C^{-1})\tilde{D}(NC)$, a matriz de massa local.

As matrizes de massa e rigidez globais são montadas com base nas matrizes globais, como

exposto na seção (2.3).

3.3 Aproximação de quinto grau com 18 variáveis nodais

Nesta seção, descreveremos uma formulação baseada na aproximação polinomial de quinto grau que apresentamos na seção (2.3). Essa formulação foi motivada pelo fato de o elemento de quinto grau, originalmente, apresentar uma quantidade diferente de variáveis nodais por nó (ele apresenta seis variáveis nodais nos vértices dos triângulos: valores da função, as primeiras e segundas derivadas; e uma variável nodal: a derivada normal, em cada ponto médio de aresta dos triângulos). A ideia é trabalhar com um elemento finito triangular que não apresente essa heterogeneidade na quantidade de variáveis nodais por nó. Trabalharemos com 18 variáveis nodais distribuídas igualmente entre os três vértices do triângulo, e isso será feito a partir do elemento triangular usado até agora na aproximação polinomial de quinto grau, eliminando dele a derivada normal dos pontos médios das arestas.

A maneira fazer isso, segundo [15] e [18], é *impor que a derivada normal seja um polinômio cúbico ao longo da aresta do triângulo.* Com os valores de $\frac{\partial u}{\partial \vec{n}}$ e $\frac{\partial^2 u}{\partial n \partial s}$ nos extremos de cada aresta é possível determinar unicamente[1] o polinômio cúbico $\frac{\partial u}{\partial n}$, de modo que o valor da derivada normal no ponto médio fica automaticamente determinado. Ainda, numa aresta comum a dois elementos triangulares o polinômio de grau 3 é único, o que garante a continuidade da derivada normal entre os elementos e, portanto, a conformidade desse elemento finito. Isso é possível no nosso elemento triangular pois $\frac{\partial u}{\partial n}$ e $\frac{\partial^2 u}{\partial n \partial s}$ são determinadas pelas seis variáveis nodais presentes nos vértices desse elemento: $u, u_x, u_y, u_{xx}, u_{xy}, u_{yy}$.

Desse modo, a derivada normal nos pontos médios das arestas deixa de ser variável nodal, e assim, passamos a trabalhar apenas com as 18 variáveis nodais, 6 por vértice, obtendo assim, uma homogeneidade no número de variáveis por vértice.

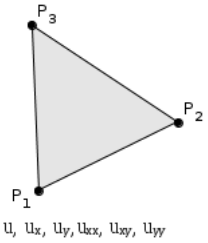


Figura 3.2: *Elemento finito de quinto grau, com 18 coeficientes.*

A implementação nesse caso segue de forma análoga ao que já foi feito nas seções anteriores: vamos montar as matrizes de massa e rigidez que compõem as formas quadráticas relativas às partes $\mathscr{I}^{(1)}$ (2.6) e $\mathscr{I}^{(2)}$ (2.7) do funcional $\mathscr{I}$ (2.3) do problema de autovalor.

Em cada elemento triangular T_i, consideraremos as 18 variáveis nodais

$$\vec{u_i} = \left(u^1, u^2, u^3, u^1_x, u^2_x, u^3_x, u^1_y, u^2_y, u^3_y, u^1_{xx}, u^2_{xx}, u^3_{xx}, u^1_{xy}, u^2_{xy}, u^3_{xy}, u^1_{yy}, u^2_{yy}, u^3_{yy}\right) .$$

A aproximação polinomial u_i continua sendo por um polinômio de grau 5, e essa será feita no

[1] Um polinômio cúbico precisa de 4 parâmetros para ser determinado. Segundo a interpolação de Hermite [15], os valores necessários para determinar um polinômio cúbico, ao longo de um segmento, são os valores do polinômio e de sua primeira derivada nos extremos do segmento considerado.

triângulo padrão T_0, do plano $\xi\eta$.

$$u_i(\xi,\eta) = \alpha_1 + \alpha_2\xi + \alpha_3\eta + \alpha_4\xi^2 + \alpha_5\xi\eta + \alpha_6\eta^2 + \alpha_7\xi^3 + \alpha_8\xi^2\eta + \alpha_9\xi\eta^2 + \alpha_{10}\eta^3 + \alpha_{11}\xi^4 + \alpha_{12}\xi^3\eta + \alpha_{13}\xi^2\eta^2 + \alpha_{14}\xi\eta^3 + \alpha_{15}\eta^4 + \alpha_{16}\xi^5 + \alpha_{17}\xi^4\eta + \alpha_{18}\xi^3\eta^2 + \alpha_{19}\xi^2\eta^3 + \alpha_{20}\xi\eta^4 + \alpha_{21}\eta^5. \quad (3.29)$$

Os 21 coeficientes $\alpha_1, \alpha_2, ..., \alpha_{21}$ deste polinômio serão unicamente determinados pelas variáveis nodais do triângulo padrão T_0

$$\vec{\tilde{u}}_i = \left(\tilde{u}^1, \tilde{u}^2, \tilde{u}^3, \tilde{u}^1_\xi, \tilde{u}^2_\xi, \tilde{u}^3_\xi, \tilde{u}^1_\eta, \tilde{u}^2_\eta, \tilde{u}^3_\eta, \tilde{u}^1_{\xi\xi}, \tilde{u}^2_{\xi\xi}, \tilde{u}^3_{\xi\xi}, \tilde{u}^1_{\xi\eta}, \tilde{u}^2_{\xi\eta}, \tilde{u}^3_{\xi\eta}, \tilde{u}^1_{\eta\eta}, \tilde{u}^2_{\eta\eta}, \tilde{u}^3_{\eta\eta}\right) .$$

pela relação $\vec{\tilde{u}} = C\vec{\alpha}$, feita através da matriz C, de dimensão 18×21. Com a "inversa C^{-1}" (e teremos que dar sentido à inversão), obtemos a relação $\alpha =$ "C^{-1}"$\vec{\tilde{u}}$, que usaremos posteriormente.

A pergunta natural que surge nesse ponto é como determinar essa matriz "C^{-1}", de dimensão 21×18, que relaciona os coeficientes da aproximação com as variáveis nodais?

Para responder a essa pergunta, primeiro vamos considerar as condições/restrições que surgem da imposição de que a derivada normal ao longo da aresta seja um polinômio cúbico. Essas restrições são dadas por equações que envolvem os coeficientes $\alpha_1, ..., \alpha_{21}$ da aproximação, e serão úteis para esse processo.

Para definir essas condições, considere, no triângulo padrão T_0, $\tilde{u}_i(\xi,\eta)$ o polinômio de quinto grau (3.29) nas variáveis ξ e η; $\tilde{u}_i(\xi,\eta)$ é a imagem de $u_i(x,y)$ no triângulo T_i.

Considere a mudança de coordenadas $F : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ que leva o triângulo T_i, de coordenadas $P_1 = (x_1, x_2)$, $P_1 = (x_1, x_2)$ e $P_1 = (x_1, x_2)$ no triângulo padrão T_0, de coordenadas $Q_1 = (0,0)$, $Q_2 = (1,0)$ e $Q_3 = (0,1)$.

Sejam n_1, n_2 e n_3 os vetores normais às arestas P_2P_3, P_2P_3 e P_2P_3, respectivamente (nesta notação, usada aqui e nas demais implementações, consideramos que o i-ésimo vetor normal está na aresta oposta ao i-ésimo vértice). As imagens, no plano $\xi\eta$, do vetores normais n_1, n_2 e n_3 do plano xy são os vetores v_1, v_2 e v_3. Cada v_i tem coordenadas (v_{i1}, v_{i2}) e $v_i = DFn_i$, $\forall\ i \in \{1,2,3\}$, em que $DF : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ é a derivada da transformação F.

A derivada normal $\partial_n\tilde{u}(x(s), y(s))$ em T_0 é dada por

$$\tilde{u}_n(x(s), y(s)) = \nabla\tilde{u}(F(x(s), y(s))) \cdot \vec{\mathbf{v}} = \nabla\tilde{u}(x(s), y(s)) \cdot DF\vec{n}$$

A partir de (3.29), no triângulo padrão T_0 temos que

$$u_\xi = a_2 + 2a_4\xi + a_5\eta + 3a_7\xi^2 + 2a_8\xi\eta + a_9\eta^2 + 4a_{11}\xi^3 + 3a_{12}\xi^2\eta + 2a_{13}\xi\eta^2 + a_{14}\eta^3 + 5a_{16}\xi^4 + 4a_{17}\xi^3\eta + 3a_{18}\xi^2\eta^2 + 2a_{19}\xi\eta^3 + a_{20}\eta^4.$$

$$u_\eta = a_3 + a_5\xi + 2a_6\xi + a_8\xi^2 + 2a_9\xi\eta + 3a_{10}\eta^2 + a_{12}\xi^2\eta + 3a_{14}\xi\eta^2 + 3a_{12}\xi^2\eta + 2a_{13}\xi\eta^2 + 3a_{14}\xi\eta^2 + 4a_{15}\eta^3 + a_{17}\xi^4 + 2a_{18}\xi^3\eta + 3a_{19}\xi^2\eta^2 + 4a_{20}\xi\eta^3 + 5a_{21}\eta^4 \quad (3.30)$$

Assim, nas arestas

$Q_1Q_2 : \tilde{u}_n(x(s), y(s)) = \tilde{u}_\xi(\xi, 0)v_{31} + \tilde{u}_\eta(\xi, 0)v_{32}$;

$Q_2Q_3 : \tilde{u}_n(x(s), y(s)) = \tilde{u}_\xi(\xi, 1-\xi)v_{11} + \tilde{u}_\eta(\xi, 1-\xi)v_{12}$;

$Q_3Q_1 : \tilde{u}_n(x(s), y(s)) = \tilde{u}_\xi(0, \eta)v_{21} + \tilde{u}_\eta(0, \eta)v_{22}$.

Impor que $\tilde{u}_n$ seja de grau 3, significa impor restrições às parcelas de grau 4 desses polinômios. Assim, considerando as expressões acima e as equações de (3.30) obtemos as seguintes restrições:

$$\begin{cases} 5v_{31}a_{16} + v_{32}a_{17} = 0 \\ 5v_{21}a_{20} + v_{22}a_{21} = 0 \\ 5v_{11}a_{16} + (-4v_{11} + v_{12})a_{17} + (3v_{11} - 2v_{12})a_{18} + (-2v_{11} + 3v_{12})a_{19} + (v_{11} - 4v_{12})a_{20} + 5v_{12}a_{21} = 0 \end{cases} \tag{3.31}$$

Essas três restrições podem ser organizadas matricialmente em função do vetor de coeficientes

$$L'\vec{\alpha} = 0$$

em que L' é uma matriz 3×21. Tais restrições serão usadas para completar a matriz C a partir de L' e obter a matriz $\tilde{L}$, de dimensão 21×21:

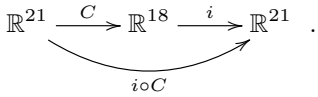
$$\tilde{L} = C \oplus L' \ .$$

Os espaço em branco na matriz C indicam a presença do número zero.

$$C = \begin{bmatrix}
1 & & & & & & & & & & & & & & & & & & & & \\
1 & & & & & & & & & & & & & & & & & & & & \\
1 & 1 & & 1 & & 1 & & & 1 & & & & 1 & & & & & & & & \\
1 & & 1 & & 1 & & & 1 & & & & 1 & & & & & & & 1 & & \\
 & 1 & & & & & & & & & & & & & & & & & & & \\
 & 1 & & 2 & & 2 & & & 4 & & & & 5 & & & & & & & & \\
 & 1 & & & 1 & & & 1 & & & & 1 & & & & & & 1 & & \\
 & & 1 & & & & & & & & & & & & & & & & & & \\
 & & 1 & & 1 & & & 1 & & & & 1 & & & & & & & & & \\
 & & 1 & & & 2 & & & 3 & & & & 4 & & & & & & & 5 & \\
 & & & 2 & & & & & & & & & & & & & & & & & \\
 & & & 2 & & & 6 & & & & 12 & & & & 20 & & & & & & \\
 & & & 2 & & & & 2 & & & & 2 & & & & & 2 & & & & \\
 & & & & 1 & & & & & & & & & & & & & & & & \\
 & & & & 1 & & & 2 & & & & 3 & & & & 4 & & & & & \\
 & & & & 1 & & & & 2 & & & & 3 & & & & 4 & & & & \\
 & & & & & 2 & & & & & & & & & & & & & & & \\
 & & & & & 2 & & & 2 & & & & 2 & & & & 2 & & & & \\
 & & & & & 2 & & & & 6 & & & & 12 & & & & & & 20 &
\end{bmatrix}_{18\times 21}$$

$$L' = \begin{bmatrix}
0 & \dots & 0 & 5v_{31} & v_{32} & 0 & 0 & 0 & 0 \\
\vdots & \ddots & \vdots & 0 & 0 & v_{21} & v_{22} & 0 & \\
0 & \dots & 0 & 5v_{11} & (-4v_{11} + v_{12}) & (3v_{11} - 2v_{12}) & (-2v_{11} + 3v_{12}) & (v_{11} - 4v_{12}) & 5v_{12}
\end{bmatrix}_{3\times 21}$$

Olhando para as transformações lineares definidas pelas matrizes a cima, podemos considerar a seguinte composição:



em que $i : \mathbb{R}^{18} \longrightarrow \mathbb{R}^{21}$ leva um vetor de $\mathbb{R}^{18}$ em $\mathbb{R}^{21}$ zerando as três últimas componentes. Assim, a composição $i \circ C$ leva vetores de $\mathbb{R}^{21}$ em vetores de $\mathbb{R}^{21}$ com as três últimas coordenadas nulas.

Por outro lado, $\tilde{L}$ também leva o $\mathrm{Ker}L'$, que tem dimensão 18[2], em $\mathbb{R}^{21}$ com as três últimas coordenadas nulas. Ou seja, $\tilde{L}$ e $i \circ C$ coincidem em $\mathrm{Ker}L'$. Portanto,

$$\tilde{L}^{-1} \circ i : \mathbb{R}^{18} \longrightarrow \mathrm{Ker}L'$$

é a inversa de C restrita a $\mathrm{Ker}L'$.

Em outras palavras: o núcleo de L' determina, no espaço de polinômios de grau 5 escritos na base canônica, um subespaço de dimensão 18. A restrição de C a $\mathrm{Ker}L'$ é um isomorfismo cuja inversa é dada pela composição $\tilde{L}^{-1} \circ i$.

As demais etapas desse processo, para a discretização das integrais $\mathscr{I}^{(1)}$ e $\mathscr{I}^{(2)}$ permanecem análogas ao que foi feito na Seção 3.2.4, já que a aproximação é feita em T_0 do plano $\xi\eta$. O que muda aqui é que as matrizes de rigidez e massa locais resultantes tem dimensão 18×18.

Essa característica é importante do ponto de vista numérico, pois isso faz com que as dimensões das matrizes globais, relativas a uma mesma malha, sejam menores.

[2]Na verdade, seria preciso demonstrar que as equações de (3.31) são LD.

Capítulo 4

Resultados

O objetivo deste capítulo é apresentar os resultados que obtivemos ao resolver o problema de autovalor $\Delta\Delta u = \lambda u$ (2.1), com condição de contorno *bordo grampeado* (2.2), numericamente. Faremos isso com base nas formulações quadrática *não conforme* e nas duas variações da formulação de quinto grau: aquela com interpolação, que chamaremos aqui de *conforme*, e a que utiliza apenas 18 variáveis nodais, que chamaremos de *conforme18*, e cujos detalhes já foram descritos nos Capítulos 2 e 3. Utilizamos o livro *"Vibration of plates"*, de Arthur W. Leissa [8], uma coleção de resultados (experimentais e numéricos) de vários autores, que apresenta tabelas e imagens relativas às primeiras frequências e modos de vibração de placas de diversos formatos, além de várias combinações de condições de contorno.

A escolha dos domínios poligonais utilizados para a obtenção dos resultados computacionais foi baseada, para efeito de comparação, nos formatos presentes nessa referência. Utilizamos principalmente formatos retangular e triangular para as placas, apesar de o programa implementado permitir o uso de qualquer formato poligonal, como veremos com alguns exemplos adicionais.

Alguns dos resultados foram selecionados e tomados como parâmetro para analisar nossas formulações e compará-las entre si. Essa comparação foi feita utilizando-se uma sequência de autovalores obtidos conforme o refinamento das malhas e, com base nela, elegemos a "melhor" formulação, e esta foi utilizada para a geração da maior parte das imagens onde visualizamos os modos de vibração. Analisamos também a variação do parâmetro ν (o coeficiente de Poisson), presente na EDP, as linhas nodais de placas retangulares de diferentes tamanhos e observamos o efeito de combinações lineares de autofunções relativas a um mesmo autovalor, fenômeno que aqui chamamos de *ressonância*.

As malhas triangulares, que usamos para a discretização do domínio e seus refinamentos, foram geradas pelo software *Triangle Mesh Generator* [16]. A implementação foi programada na linguagem FORTRAN 90 e, para a solução do problema de autovalor generalizado, utilizamos a rotina *DSYGV* do LAPACK (Liner Algebra Package), que fornece como solução uma lista dos autovalores e autovetores associados. As imagens das linhas nodais foram obtidas através de uma pequena rotina feita no software Scilab 5.3.3 (Scientific Software Package for Numerical Computations).

Os programas implementados (e cujos códigos podem ser vistos no Apêndice B) fornecem como resultado uma lista de autovalores λ – organizada em ordem crescente – e uma lista de autovetores $\vec{u}$ (que aproximam as autofunções) associados a cada um deles. O número total de autovalores e autovetores obtidos é a dimensão final das matrizes envolvidas no problema de autovalor generalizado. Escolhemos trabalhar com os 25 primeiros pares $(\lambda, \vec{u})$ obtidos – quantidade suficiente para as análises e considerações que fizemos.

4.1 Autovalores/Coeficientes de frequência

Neste capítulo, os autovalores λ obtidos serão, muitas vezes, representados por uma quantidade normalizada que chamamos de *coeficiente de frequência* $\kappa = \omega\sqrt{\frac{\rho}{D}}a^2$ (1.44), em que a representa uma medida linear da placa, escolhida para comparação com [8]: para retângulos, a é o comprimento do menor lado e, para triângulos isósceles, a é o comprimento de um dos lados congruentes, como descrito na subseção 1.3.1 do Capítulo 1. Esse parâmetro dependente apenas do formato placa e é invariante por homotetias. Como $\lambda = \omega^2\frac{\rho}{D}$, o valor de κ fica em função de λ: $\kappa = a^2\sqrt{\lambda}$. Comparamos os nossos valores de κ com o valor de referência (κ_{ref}) da tabela 4.44 de [8].

Um primeiro aspecto a observar é como a sequência de cada autovalor – ou de cada coeficiente de frequência – evolui com o refinamento da malha. Esperamos resultados melhores a cada refinamento da malha, isto é, com o aumento do número de elementos triangulares, e olhar para essa evolução foi uma maneira de comparar as diferentes formulações entre si.

Observamos essa evolução para o primeiro autovalor, representado aqui pelo coeficiente de frequência κ. As tabelas a seguir mostram essa evolução para placas quadrada (tabela 4.1), retangular (tabela 4.2) e triangular (um triângulo retângulo isósceles) (tabela 4.3) para cada uma das três formulações. Em cada coluna é possível observar como os valores obtidos se aproximam do valor de referência conforme a malha é refinada. As lacunas presentes nessas tabelas indicam resultados não obtidos para a malha correspondente. No caso da tabela 4.1 isso deve ao fato de as formulações de quinto grau serem mais "pesadas" (ou dispendiosas) – no sentido de consumo de tempo de processamento e espaço de memória – , o que não nos permitiu rodá-las em malhas tão finas quanto foi possível para a formulação quadrática. No caso da Tabela 4.2, isso se deve a problemas numéricos e limitações da própria implementação, que no meio da execução impossibilitaram a solução completa do problema.

Placa quadrada - coeficientes de frequência κ				
nº de elementos da malha	Quadrática	Quinto grau		Valor de referência κ_{ref} [8]
	não conforme	conforme	conforme18	
319	33.5841318	35.23495326	32.35117814	
632	34.67142199	35.34212194	32.94295204	
1227	35.33139219	35.39459758	32.9528213	
2538	35.65701631	35.33908992	33.12628812	35,9866 (tab. 4.22a)
3107	35.7269665	35.42744491	33.27039132	35,99 (tab. 4.22b)
3629	35.76559433	35.43789648	33.32938938	
3826	35.77696973	35.43782546	33.37521013	
5142	35.82776304	-	-	

Tabela 4.1: *Coeficientes de frequência $\kappa = a^2\sqrt{\lambda}$ por formulação. Placa quadrada de dimensão 1.0.*

Os valores apresentados nessas tabelas nos mostram que a sequência dos primeiros coeficientes de frequência é crescente e, no caso da formulação quadrática, se aproxima do valor de referência tabelado. No caso das formulações de quinto grau, o crescimento é mais lento e uma primeira impressão é que, para obter valores mais próximos daqueles esperados, precisaríamos trabalhar com malhas muito mais finas que essas. A formulação *conforme* apresenta valores maiores, com malhas mais grossas, mas não evolui tanto quanto a formulação quadrática. Já a formulação *conforme18* possui valores bem menores do que as duas anteriores. Na Seção 4.3 apresentaremos uma análise mais detalhada para comparação dessas três formulações.

4.2 Autofunções e modos de vibração de placas retangulares

Cada autofunção obtida na solução do problema, representa a função deslocamento transversal $u(x, y)$ da placa em função dos pontos (x, y) do plano. O gráfico da função u é, originalmente, tridimensional, mas estamos interessados na sua representação plana, em duas dimensões, em que o valor de $u(x, y)$ é representado em escala de cores. As autofunções estão sendo representadas (aproximadas) por vetores no $\mathbb{R}^n$ (n é o número de vértices de triângulos da malha) dos valores de u nesses vértices. Esse vetor é extraído do autovetor discreto $\vec{u}_i$ (o vetor de variáveis nodais que contém valores de u e de suas derivadas nos nós da malha) obtido na solução numérica e consiste de uma lista com o valor da função u nos nós da malha que são vértices de algum triângulo.

Utilizamos uma função do Scilab própria para a manipulação de elementos finitos que possuem esses valores como variáveis nodais. Essa função recebe informações sobre a malha triangular (as coordenadas dos nós que são vértices do triângulo) e os valores da função u que serão atribuídos a eles, e gera um gráfico em 2D, em escala de cores, automaticamente. Optamos por utilizar a escala de cinza de modo semelhante ao apresentado em [9], a fim de permitir a visualização das curvas no plano para as quais $u(x, y) = 0$, chamadas de *linhas nodais*. As linhas nodais são curvas ao longo das quais a placa permanece em repouso durante a vibração. Essas curvas caracterizam o *modo de vibração* da placa.

A construção desse gráfico foi feita tomando o valor absoluto das autofunções e organizando a escala de cinza de modo que a cor preta correspondesse às regiões do plano que possuem deslocamento nulo ($u(x, y) = 0$) e a cor branca correspondesse ao valor de deslocamento máximo (amplitude

Placa retangular - coeficientes de frequência κ				
nº de elementos da malha	Quadrática	Quinto grau		Valor de referência κ_{ref} [8]
	não conforme	conforme	conforme18	
296	22.0221654	23.98485366	-	
627	23.41443928	24.16683788	22.47220767	
767	23.62005658	24.17894531	22.68108988	24.09 (tab. 4.29)
994	23.82941881	24.16442392	-	24.56 (tab. 4.28)
1528	24.08008887	24.15969971	22.86074454	
3112	24.32646342	24.24799705	22.80703459	
3861	24.37815486	24.22625462	22.82143195	

Tabela 4.2: *Coeficientes de frequência $\kappa = a^2\sqrt{\lambda}$ por formulação. Placa retangular de dimensão* 2.0×1.0.

Placa triangular - coeficientes de frequência κ				
nº de elementos da malha	Quadrática	Quinto grau		Valor de referência κ_{ref} [8]
	não conforme	conforme	conforme18	
311	85.37310338	91.52395119	83.03166215	
519	88.73446013	91.96032881	84.58281455	
634	89.59648672	92.02291664	85.15543253	93.60 (tab. 7.2)
770	90.34319415	92.00013615	84.98697688	
1032	91.12934193	92.12988043	85.63462458	
1559	92.00662042	92.12988043	85.29307103	
2059	92.46434819	92.3429864	86.1435161	
3846	93.05849089	92.43861774	86.02196328	

Tabela 4.3: *Coeficientes de frequência $\kappa = a^2\sqrt{\lambda}$ por formulação. Placa em formato de triângulo retângulo isósceles. Lado congruente mede* $a = 2.0$.

máxima). Assim, as linhas nodais ficam evidentes e destacadas na cor preta. Ao analisar as imagens usaremos essas linhas e as regiões entre elas como referência.

Cada autofunção que representa um modo de vibração está associada a um autovalor λ e, portanto, a um coeficiente de frequência κ.

Cada figura foi organizada com a imagem dos 25 primeiros modos de vibração de uma placa. Veja os exemplos a seguir, gerados pela formulação quadrática, para placas de diferentes formatos. O número abaixo de cada modo é o autovalor λ correspondente (ou coeficiente de frequência, conforme o caso).

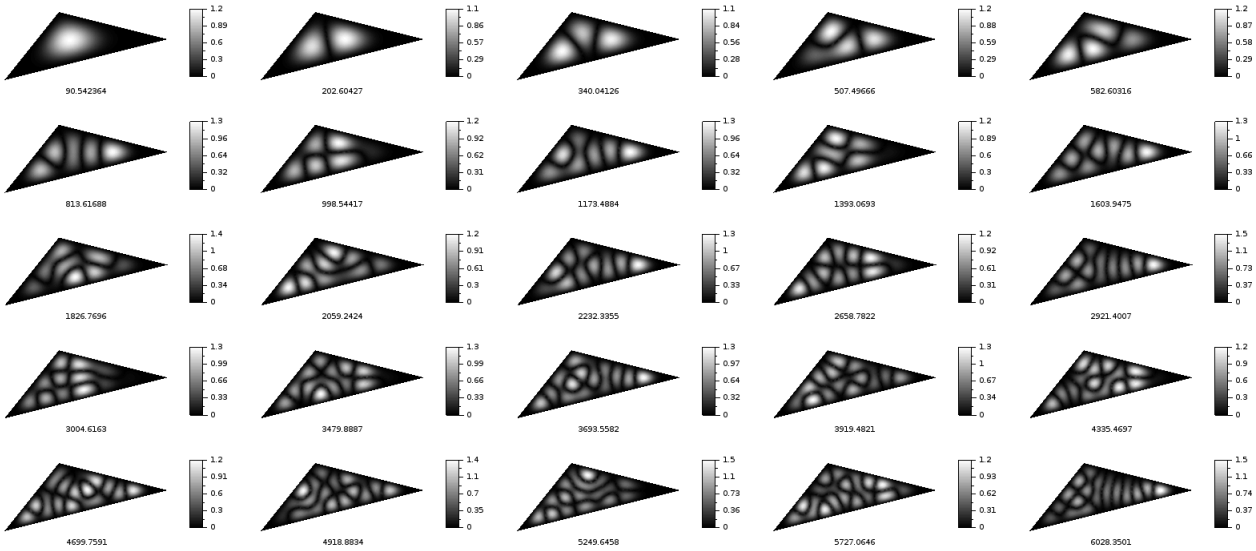


Figura 4.1: *Primeiros modos de vibração e autovalores correspondentes de uma placa no formato de um triângulo escaleno de coordenadas (0,0), (6,1.5) e (2,2.5).*

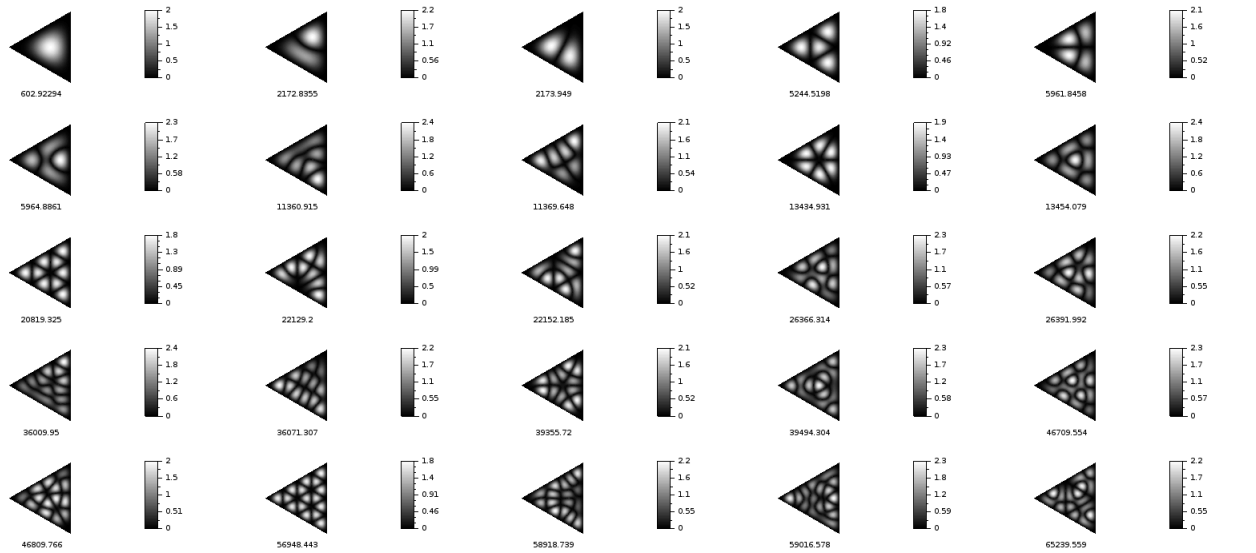


Figura 4.2: *Primeiros modos de vibração e autovalores correspondentes de uma placa no formato de um triângulo equilátero de lado $a = 2.0$.*

No caso de uma placa retangular, a notação utilizada para identificação dos modos de vibração foi dada pelo par ordenado (m, n), em que m, n são números naturais que indicam o número de subdivisões do maior e do menor lado do retângulo, respectivamente. Por exemplo, os quatro

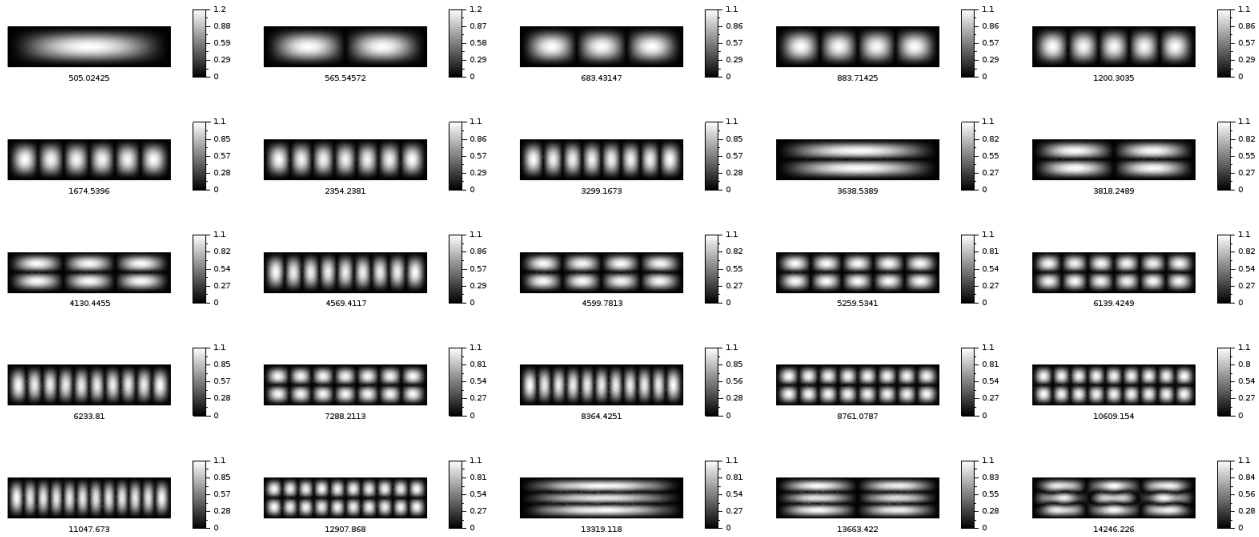


Figura 4.3: *Primeiros modos de vibração e autovalores correspondentes de uma placa retangular, de dimensão* 4.0×1.0.

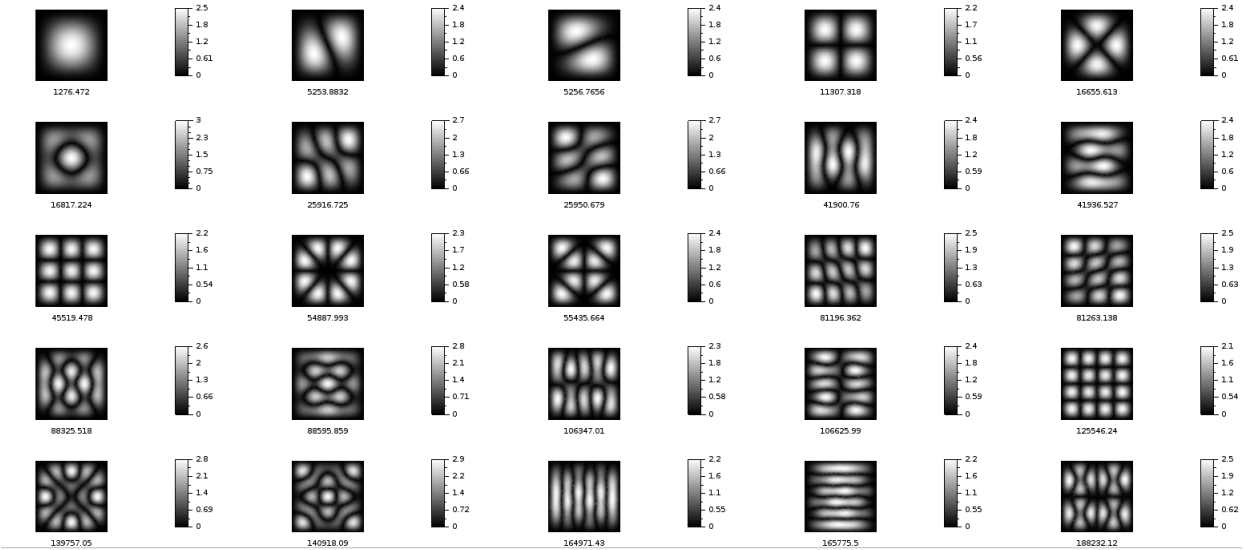


Figura 4.4: *Primeiros modos de vibração e autovalores correspondentes da placa quadrada de lado de lado* $a = 1.0$.

primeiros modos de vibração da placa retangular de dimensão 4.0×1.0, que aparecem na Figura 4.3 são, respectivamente, $(1,1)$, $(2,1)$, $(3,1)$ e $(4,1)$ e as imagens nº 9,10,11 e 13 correspondem aos modos $(1,2)$, $(2,2)$, $(3,2)$ e $(4,2)$, respectivamente. No caso da placa quadrada da Figura 4.4, identificamos claramente os modos (1,1) e (2,2), (3,3) (são o 1º, 4º e 11º modos, respectivamente). A placa quadrada também apresenta modos cujas linhas nodais não são exatamente as linhas dessa classificação. As curvas que aparecem podem ser resultados de combinações de diferentes modos de vibração. Veremos isso na Seção 4.5.

4.3 Comparação das implementações

O objetivo dessa seção é comparar as três formulações (quadrática *não conforme* e as de quinto grau *conforme* e *conforme18*) a partir dos resultados obtidos para o primeiro autovalor e coeficientes

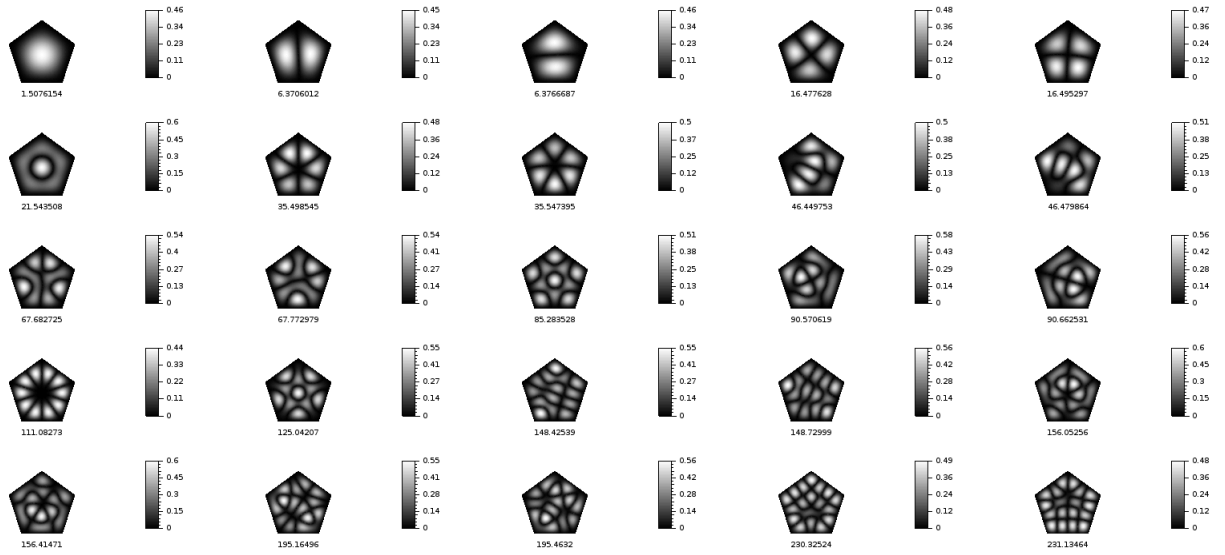


Figura 4.5: *Primeiros modos de vibração e autovalores correspondentes de uma placa pentagonal de lado de lado* $a = 2.0$.

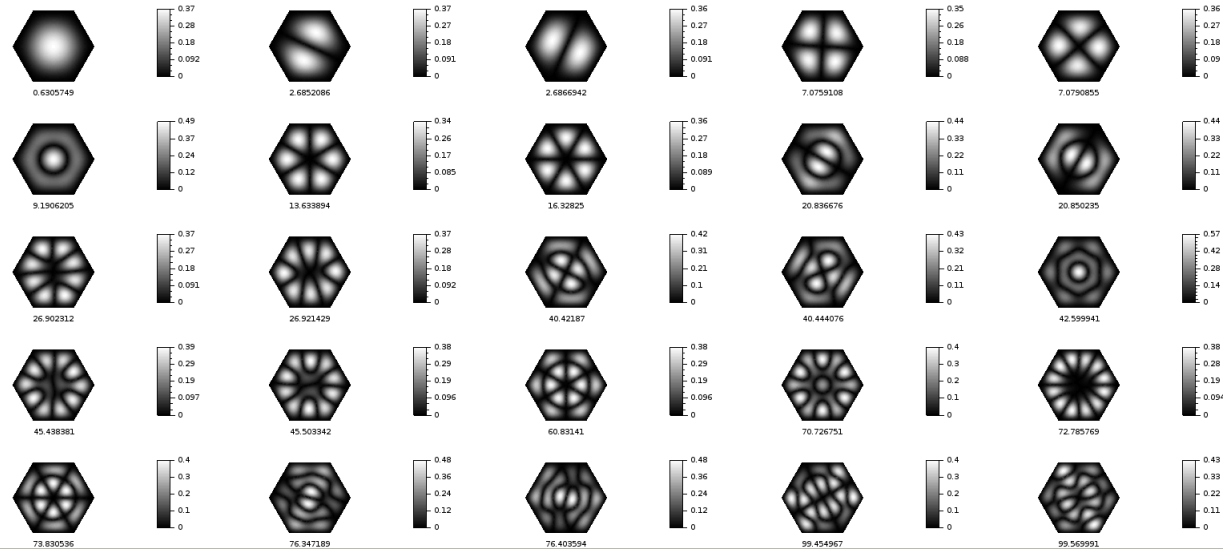


Figura 4.6: *Primeiros modos de vibração e autovalores correspondentes de uma placa hexagonal de lado de lado* $a = 2.0$.

de frequência. Analisamos a sequência dos primeiros coeficientes de frequência de uma placa com formato fixado e cuja malha foi refinada, isto é, fixado um formato para o domínio Ω, criamos uma lista de refinamentos da malha inicial, nas quais o programa foi executado, gerando uma lista dos primeiros coeficientes de frequência, ou seja, cada coeficiente de frequência da sequência formada corresponde a um refinamento da malha. Olhamos para essa sequência a fim de verificar se, e como, os valores se aproximam dos valores de referência de [8], à medida que a malha ia ficando mais fina. Fizemos isso para as três formulações, a fim de comparar a "convergência" do primeiro coeficiente de frequência para as três. Um pouco disto pôde ser visto nas primeiras tabelas 4.1, 4.2 e 4.3 que apresentamos no início do capítulo. Esse teste foi feito para os domínios quadrado, retangular e triangular (um triângulo retângulo isósceles) das tabelas acima citadas. Utilizamos as mesmas malhas para cada formulação.

No caso da formulação *não conforme*, observamos que a sequência dos primeiros coeficientes de

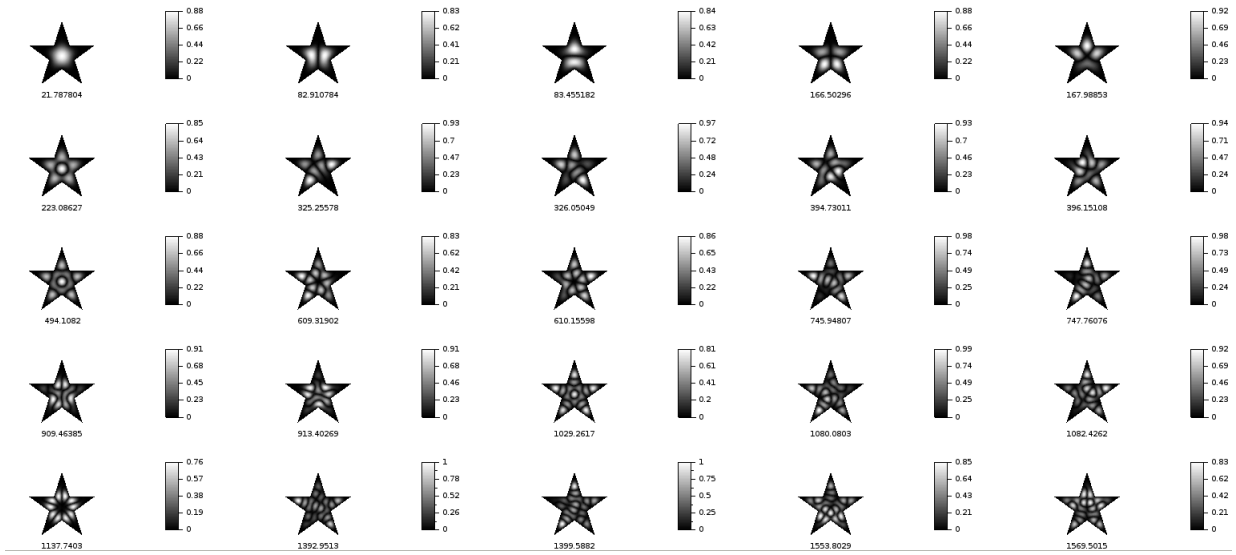


Figura 4.7: *Primeiros modos de vibração e autovalores correspondentes de uma placa no formato de estrela de cinco pontas.*

frequência é crescente e está se aproximando do valor esperado de [8] por valores menores que ele. Para cada formulação, é possível observar que a aproximação acontece, mas não da mesma forma. No caso da formulação de quinto grau as sequências não são sempre crescentes. A formulação *conforme* apresenta valores maiores – se comparada à formulação quadrática – logo de início, mas seu crescimento vai perdendo velocidade ao logo do refinamento das malhas. No caso da formulação *conforme18* a sequência como um todo possui valores bem menores do que aqueles esperados.

Para nos ajudar na análise desses dados (coeficiente de frequência para cada refinamento de malha) foi feito um ajuste dos mesmos, a uma função do tipo $\kappa = a - bN^c$, via Método dos Mínimos Quadrados (MMQ) linear, em que $\kappa = a^2\sqrt{\lambda}$ é o coeficiente de frequência, N é o número de elementos triangulares dos refinamentos da malha e a, b e c são os parâmetros que serão ajustados. O ajuste dos três parâmetros foi feito feito da seguinte forma: definimos um valor inicial a_0 para o parâmetro a e o fixamos. Para a_0 fixo, ajustamos $bN^c = \kappa - a_0$ usando a linearização com a função logarítmica

$$(\log b) + c(\log N) = \log(\kappa - a_0) \ .$$

Nesse ajuste, obtivemos valores b_1 e c_1 para os parâmetros b e c. Com b_1 e c_1 fixos, obtivemos o valor a_1 para a, pelo ajuste de $\kappa + b_0 N^{c_0} = a$. Com a_1 fixado voltamos ao passo inicial e ajustamos b e c novamente. Esse processo iterativo foi realizado 100 vezes para obtenção dos valores finais de a, b e c. Também calculamos o erro relativo médio e do ajuste dos pontos $(N_i, \kappa_i), 1 \leq i \leq n$, em que n é o total de pontos:

$$e = \frac{1}{n} \sum_{i=1}^{n} \frac{|\kappa_i - f(N_i)|}{f(N_i)} \ , \tag{4.1}$$

Para cada formato de placa, o valor inicial para o parâmetro a foi um valor aproximado do valor de referência κ_{ref} de [8], para o coeficiente de frequência. No caso da placa quadrada, $a_0 = 36.0$, para a placa retangular, $a_0 = 24.56$ e para a placa triangular $a_0 = 93.60$. Os dados ajustados são aqueles das tabelas 4.1, 4.2 e 4.3. Os resultados obtidos para cada formulação estão resumidos na tabela da Figura 4.8, na qual é possível comparar os valores das três formulações para cada formato. Os gráficos dos ajuste estão apresentados a seguir. Para a formulação quadrática, placa quadrada,

Parâmetros do ajuste MMQ e erro relativo médio

Quadrado			
	Formulações		
parâmetros	não conforme	conforme	conforme18
a	36.017	35.907	35.834
b	532.30	1.2834	6.6509
c	-0.93258	-0.11976	-0.11877
erro relativo médio	3.8968e-04	7.6278e-04	0.0021349
% ~	0.04	0.07	0.2

Retângulo			
	Formulações		
parâmetros	não conforme	conforme	conforme18
a	24.550	24.300	24.340
b	907.80	5.6869	3.1533
c	-1.0350	-0.54142	-0.091721
erro relativo médio	3.9276e-04	0.0011400	0.0032821
% ~	0.04	0.1	0.3

Triângulo			
	Formulações		
parâmetros	não conforme	conforme	conforme18
a	93.790	92.517	92.329
b	2178.9	40.149	19.025
c	-0.96885	-0.67218	-0.14103
erro relativo médio	2.9341e-04	9.4142e-04	0.0043728
% ~	0.03	0.09	0.4

Figura 4.8: *Parâmetros do ajuste MMQ para cada formulação e formato de placa.*

obtivemos $a = 36.017$, $b = 532.30$ e $c = -0.93258$, o que significa que aos pontos dados foi ajustada a função $\kappa = 36.017 + 532.30N^{-0.93258}$, cuja curva vemos na Figura 4.9. O erro relativo médio deste ajuste foi de 0.04%.

O gráfico nos mostra que, visualmente, essa função se ajustou bem aos nossos dados, de modo que podemos nos valer das suas características para entender e confirmar a tendência de convergência dos nossos dados para o valor esperado κ_{ref}, como já tínhamos percebido olhando para as tabelas. É interessante observar que o valor assintótico a é pouco maior que 36.0 e que o expoente c é próximo de -1.0. Isso significa que os dados tendem a se aproximar do valor de referência 36.0 a uma taxa aproximada de $1/N$. Quando o número de elementos N da malha cresce o "preço" de tempo e espaço pago pelo uso de uma malha mais fina é compensado pois atingimos valores cada vez mais próximos do esperado. Além disso, o erro relativo médio de 0.04% pode nos servir de parâmetro para comparar as demais formulações.

Uma análise e conclusão análoga pode ser obtida para o ajuste dos dados da placa retangular e triangular para essa mesma formulação. Para a placa retangular (Figura 4.10), obtivemos os parâmetros $a = 24.550$, $b = 907.80$ e $c = -1.0350$, com erro relativo médio de 0.04% e, para a placa triangular (Figura 4.11), obtivemos $a = 93.790$, $b = 2178.9$ e $c = -0.96885$, com erro relativo médio

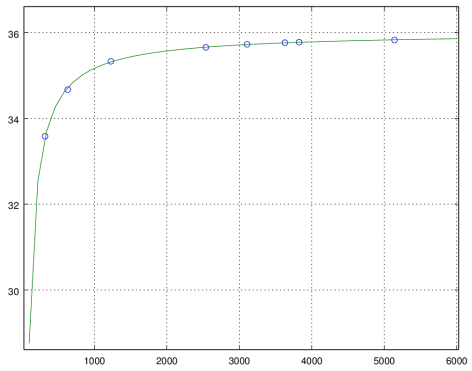


Figura 4.9: *Ajuste MMQ da convergência do primeiro coeficiente de frequência. Placa quadrada. Formulação não conforme.*

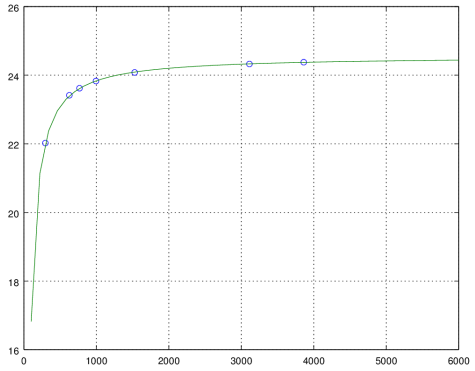


Figura 4.10: *Ajuste MMQ da convergência do primeiro coeficiente de frequência. Placa retangular. Formulação não conforme.*

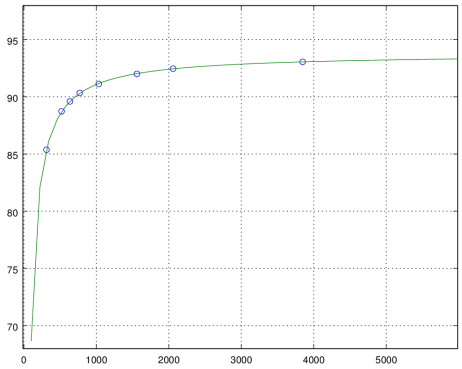


Figura 4.11: *Ajuste MMQ da convergência do primeiro coeficiente de frequência. Placa triangular.Formulação não conforme.*

0.03%.

Novamente como no caso do quadrado, nestes dois casos o valor assintótico a é próximo aos valores de referência $\kappa = 24.56$ e $\kappa = 93.60$, respectivamente e, o coeficiente c também é próximo a -1.0, o que nos indica uma tendência de convergência para o valor esperado a uma taxa próxima de $1/N$.

Vejamos agora o comportamento das formulações de quinto grau.

Para a formulação *conforme*, obtivemos, para a placa quadrada, os parâmetros $a = 35.907$, $b = 1.2834$ e $c = -0.11976$, cujo ajuste pode ser visto no gráfico da Figura 4.12. Visualmente, parece se ajustar bem aos dados, exceto pelo 4º ponto, que indica uma quebra no crescimento da sequência. O valor assintótico encontrado também é próximo do valor de referência. Entretanto, a constante c nos indica uma convergência bem mais lenta, como já observado nas tabelas, se comparada à formulação quadrática. Por outro lado, apesar de o valor assintótico encontrado a não estar tão longe do valor de referência $\kappa_{ref} = 36.0$, o ajuste não está bom o suficiente para que a interpretação dos seus parâmetros nos indique um comportamento real dos nossos dados à medida que a malha é refinada. Além disso, o erro relativo médio encontrado para esse caso foi de 0.07%, maior que o da formulação quadrática para esse formato de placa.

A mesma conclusão de qualidade do ajuste pode ser tirada para os ajustes dos dados das placas

retangular e triangular. Para a placa retangular obtivemos: $a = 24.300$, $b = 5.6869$ e $c = -0.54142$, com erro relativo médio de 0.1% e, para a placa triangular, obtivemos $a = 92.517$, $b = 40.149$ e $c = -0.67218$, com erro relativo médio 0.09%. Os ajustes nesses casos podem ser vistos nos gráficos das Figuras 4.13 e 4.14.

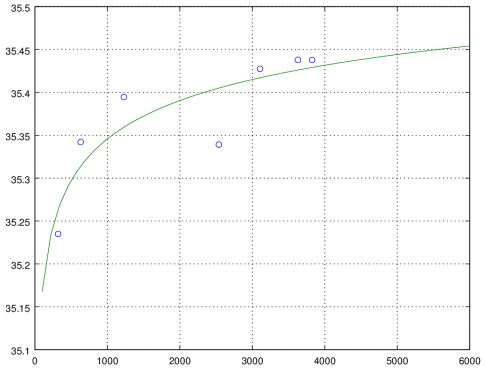


Figura 4.12: *Ajuste MMQ da convergência do primeiro coeficiente de frequência para a formulação conforme. Placa quadrada.*

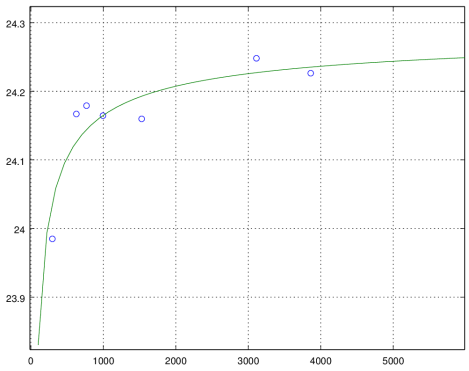


Figura 4.13: *Ajuste MMQ da convergência do primeiro coeficiente de frequência para a formulação conforme.Placa retangular.*

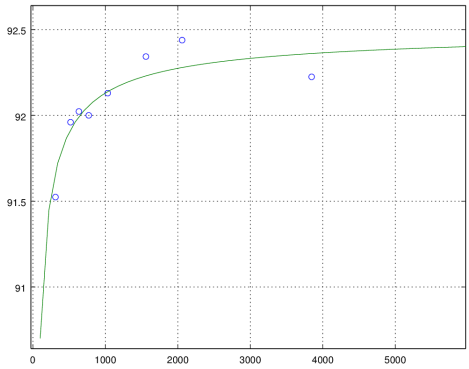


Figura 4.14: *Ajuste MMQ da convergência do primeiro coeficiente de frequência da formulação conforme. Placa triangular.*

Para a formulação *conforme18* temos uma situação parecida observável em todos os domínios. Esse é um caso que o conjunto de dados como um todo está longe do valor esperado – se comparado aos conjuntos de dados das formulações anteriores. Os gráficos das Figuras 4.15, 4.16 e 4.17 nos mostram os ajustes e por eles percebemos como os dados flutuam e se distanciam da função ajustada, principalmente no caso da placa retangular. O erro relativo médio obtido para cada formato nos ajuda a perceber a má qualidade do ajuste – principalmente quando o comparamos com a formulação quadrática. Os valores obtidos para os parâmetros a, b e c foram $a = 35.834$, $b = 6.6509$ e $c = -0.11877$, com erro relativo médio de 0.2% para a placa quadrada, $a = 24.340$, $b = 3.1533$ e $c = -0.091721$, com erro relativo médio 0.3% para a placa retangular e $a = 92.329$, $b = 1.025$ e $c = -0.14103$, com erro relativo médio de 0.4% para a placa triangular.

Nesse caso, o valor assintótico não fica tão perto do valor esperado e, acreditamos que isso se dá porque todo o conjunto de dados está bem abaixo do valor esperado. Vale lembrar que dispúnhamos de menos dados para o ajuste da placa retangular devido a uma limitação/erro computacional que

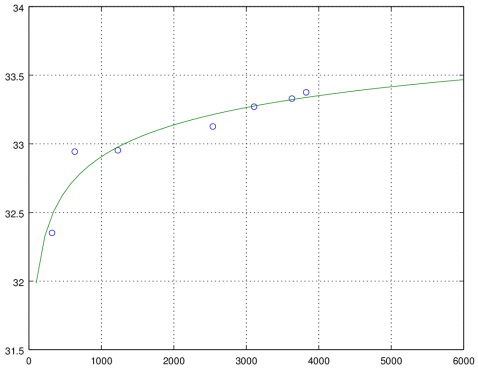


Figura 4.15: *Ajuste MMQ da convergência do primeiro coeficiente de frequência. Placa quadrada. Formulação conforme18.*

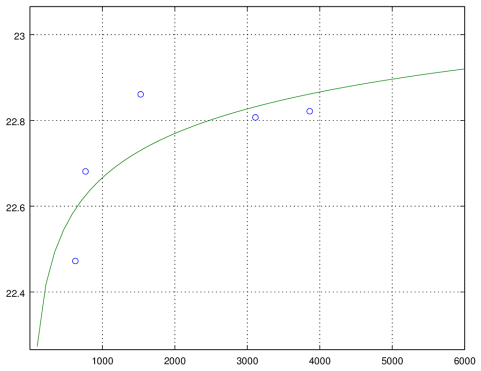


Figura 4.16: *Ajuste MMQ da convergência do primeiro coeficiente de frequência. Placa retangular. Formulação conforme 18.*

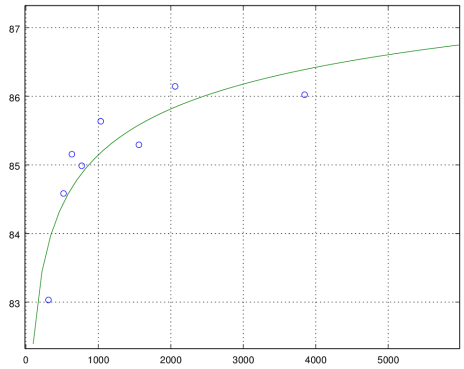


Figura 4.17: *Ajuste MMQ da convergência do primeiro coeficiente de frequência. Placa triangular. Formulação conforme18.*

não permitiu que o programa fosse inteiramente executado para algumas malhas.

Segundo o critério da análise de convergência do primeiro autovalor que foi escolhido aqui, e devido ao ajuste satisfatório dos dados da formulação quadrática, elegemos essa formulação como a que apresenta os melhores resultados e possui o melhor desempenho em termos de refinamento de malha. Desse modo, ela foi eleita para a obtenção dos demais resultados.

Entretanto, em termos das autofunções e imagens das linhas nodais obtidas, as formulações de quinto grau apresentam bons resultados, como podemos ver nas Figuras a seguir - 4.18, 4.19 e 4.20 para as placas quadradas; 4.21, 4.22 e 4.23 para as placas retangulares, 4.24, 4.25 e 4.26, para as placas triangulares – em que apresentamos as imagens das linhas nodais para as placas quadrada (malha com 3107 elementos) retangular (malha com 3112 elementos) e triangular (malha com 3846 elementos), que usamos nessa análise e onde é possível comparar as três formulações. Isso nos motiva a continuar com o estudo e o aprimoramento dessas formulações.

Escolhemos apresentar a maior parte das imagens deste trabalho obtidas pela formulação quadrática, pois foi com ela que conseguimos executar o programa para malhas mais finas com menos gasto de tempo de execução e espaço de memória.

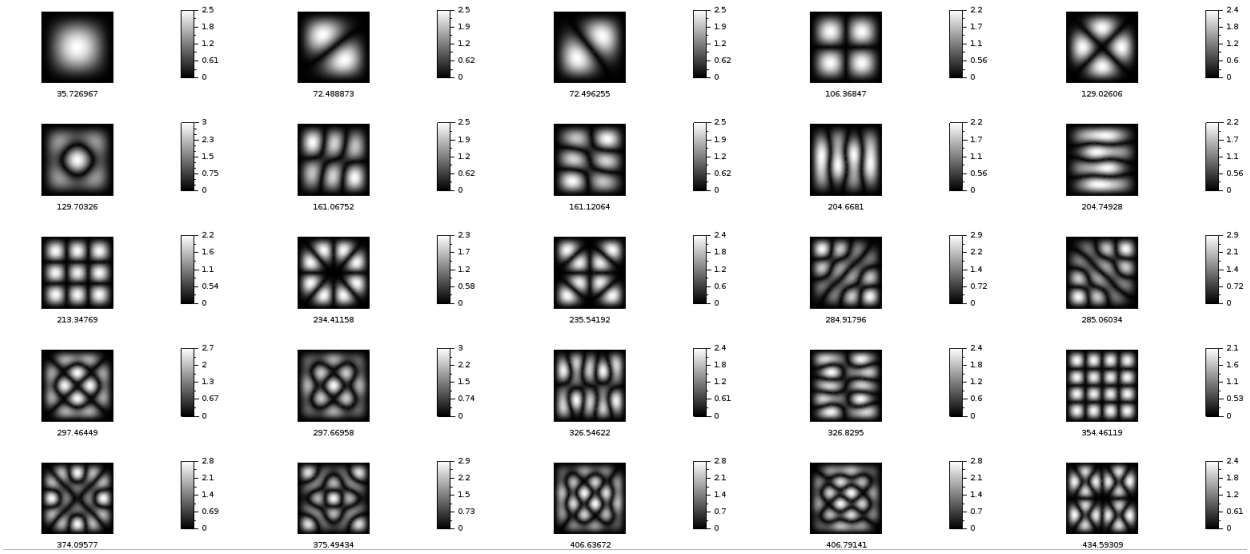


Figura 4.18: *Autofunções/linhas nodais. Placa quadrada de dimensão* 1.0*. Formulação não conforme.*

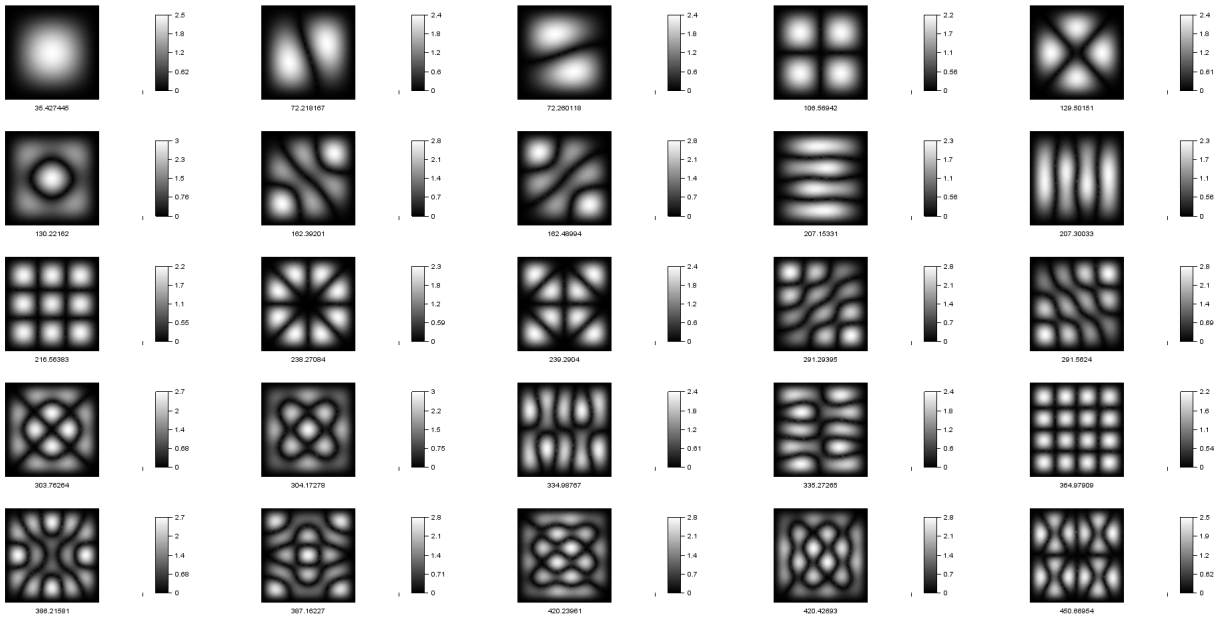


Figura 4.19: *Autofunções/linhas nodais. Placa quadrada de dimensão* 1.0*. Formulação conforme.*

4.4 Variação do coeficiente de Poisson ν

Vimos, no Capítulo 1, que o Coeficiente de Poisson ν, parâmetro presente na EDP, é a razão entre deformações horizontais e verticais da placa quando ela é excitada. Esse parâmetro depende, portanto, do material da placa. Para placas que não sofrem tais deformações quando excitadas, podemos considerar $\nu = 0.0$ [12] e, para placas de metal, ν é aproximadamente igual a 0.3. Esse parâmetro está presente na constante $D = \dfrac{Eh^3}{12(1-\nu^2)}$ da EDP (1.1) e, portanto, influencia diretamente a frequência de vibração $\omega = \sqrt{\frac{\lambda D}{\rho}}$. Entretanto, essa constante não precisa aparecer no problema de autovalor ($\Delta\Delta u = \lambda u$) se a condição de contorno considerada for a de *bordo grampeado*. Como discutimos na Subseção 1.4.2, ao menos teoricamente uma mudança no parâmetro ν não deve afetar a solução do problema de autovalor. Apesar de termos trabalhado com as condições de contorno do

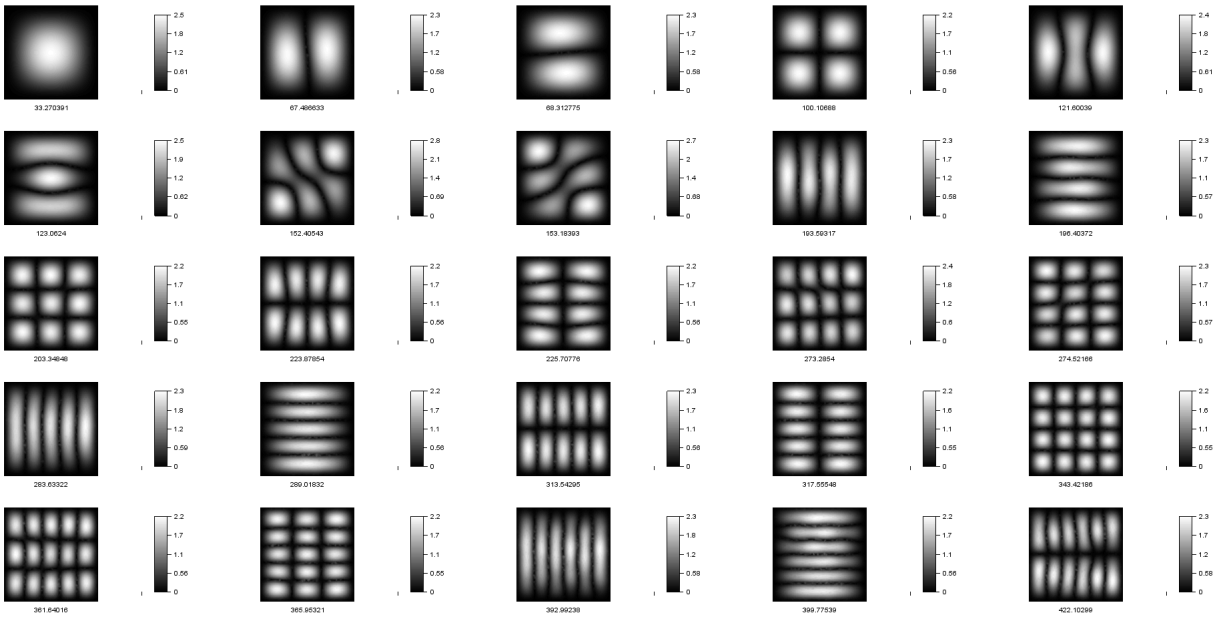


Figura 4.20: *Autofunções/linhas nodais. Placa quadrada de dimensão* 1.0*. Formulação conforme18.*

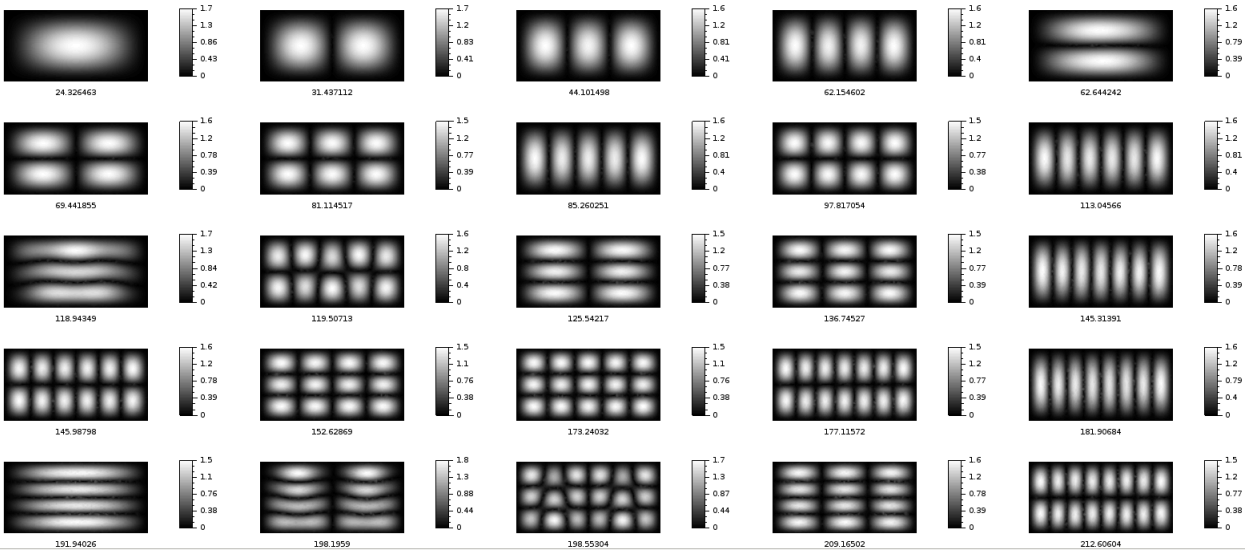


Figura 4.21: *Autofunções/linhas nodais. Placa retangular de dimensão* 2.0×1.0*. Formulação não conforme.*

bordo grampeado, fizemos testes, resolvendo o problema de autovalor para outros valores de ν para perceber possíveis variações nos resultados. A teoria nos diz que não devemos esperar variações, mas não esperamos valores exatamente iguais pois existem erros numéricos envolvidos. Os valores de ν foram escolhidos no intervalo $[0, 1)$, a placa utilizada foi uma placa quadrada de lado 1.0, com 3826 elementos triangulares. Escolhemos para ν os valores 0.0, 0.6 e 0.9 além do valor $\nu = 0.3$, valor padrão utilizado.

Observamos, na Tabela (4.27), um leve aumento dos autovalores com a diminuição do parâmetro ν, mas não é claro porque isso acontece. O teste com $\nu = 0.0$ apresentou valores mais próximos dos valores de referência disponíveis, mas entre um valor e outro de ν não há muita diferença entre os resultados obtidos. Sendo assim, não há porque considerar a formulação com $\nu = 0.0$ melhor que as demais pois é possível que, com malhas mais finas, esses valores sejam atingidos mesmo para $\nu = 0.3$. A maior diferença se dá para $\nu = 0.9$, que apresenta os valores mais baixos.

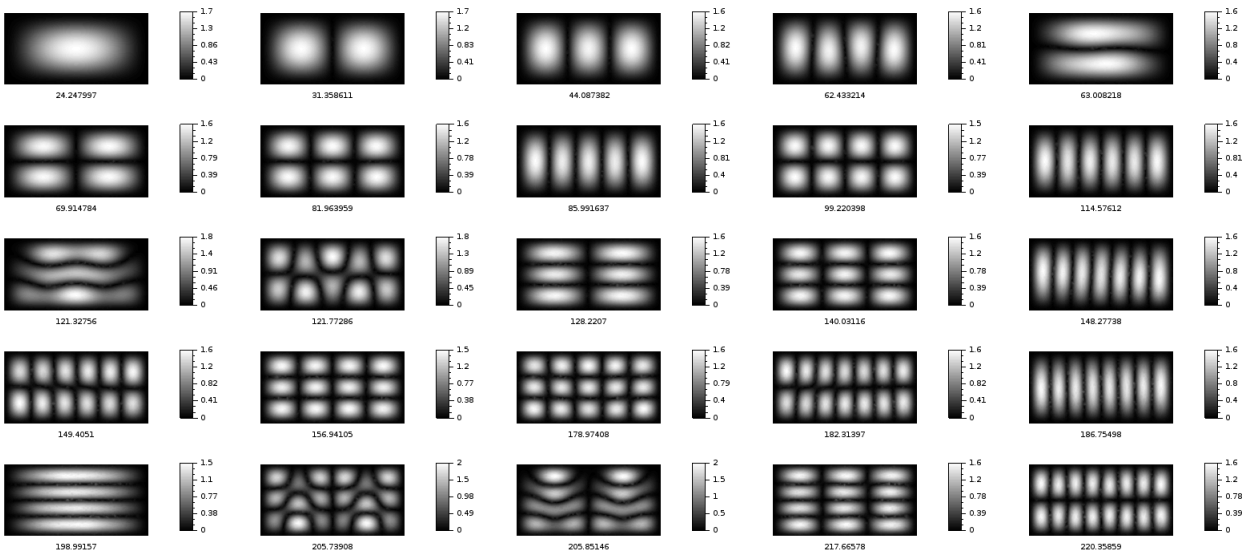


Figura 4.22: *Autofunções/linhas nodais. Placa retangular de dimensão* 2.0×1.0. *Formulação conforme.*

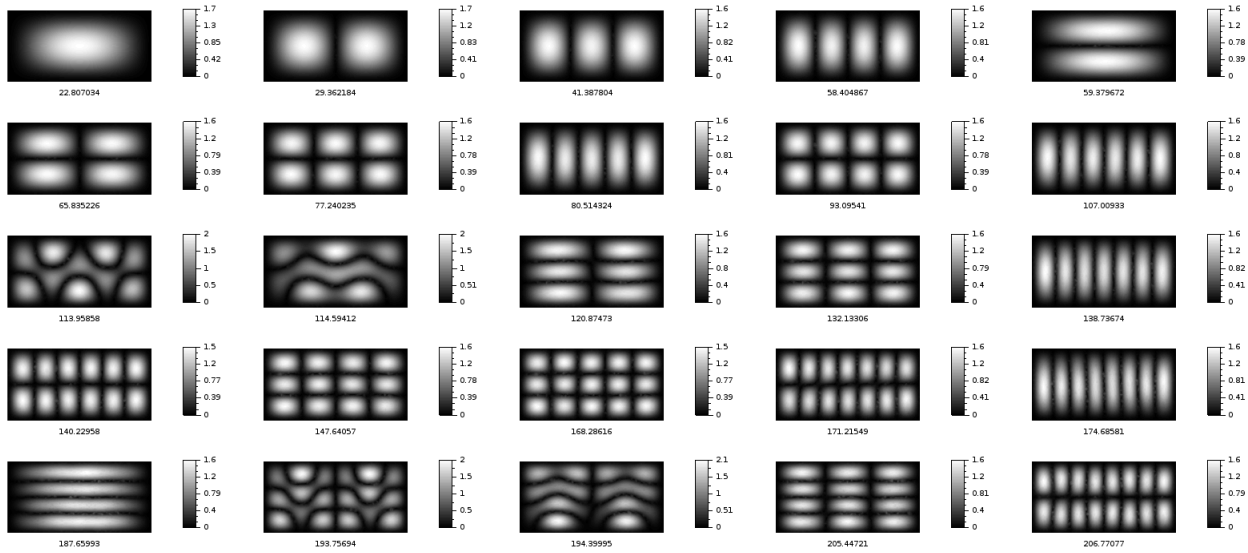


Figura 4.23: *Autofunções/linhas nodais. Placa retangular de dimensão* 2.0×1.0. *Formulação conforme18.*

As linhas nodais das autofunções relativas a cada valor de ν podem ser observadas nas Figuras 4.28, 4.29, e 4.30, onde percebemos um padrão dessas linhas para todos os casos. Observamos que possível perceber que até a 16º imagem as linhas nodais são essencialmente as mesmas, caracterizando os mesmos modos de vibração e, que as diferenças observadas nos modos 17º ao 24º, podem perfeitamente ser devidas ao fenômenos de *ressonância* que discutiremos na Seção 4.5.

4.5 Autovalores múltiplos - ressonâncias

Nesta seção falaremos de um fenômeno interessante que pode ser observado nas imagens das linhas nodais para placas de certos formatos. Esse fenômeno é o efeito da combinação linear de autofunções associadas a um mesmo autovalor ou frequência. Sempre que temos duas autofunções associadas a um mesmo autovalor, qualquer combinação linear das duas também é uma autofunção associada a esse mesmo autovalor e, por esse motivo, o número de autofunções associadas "cresce"

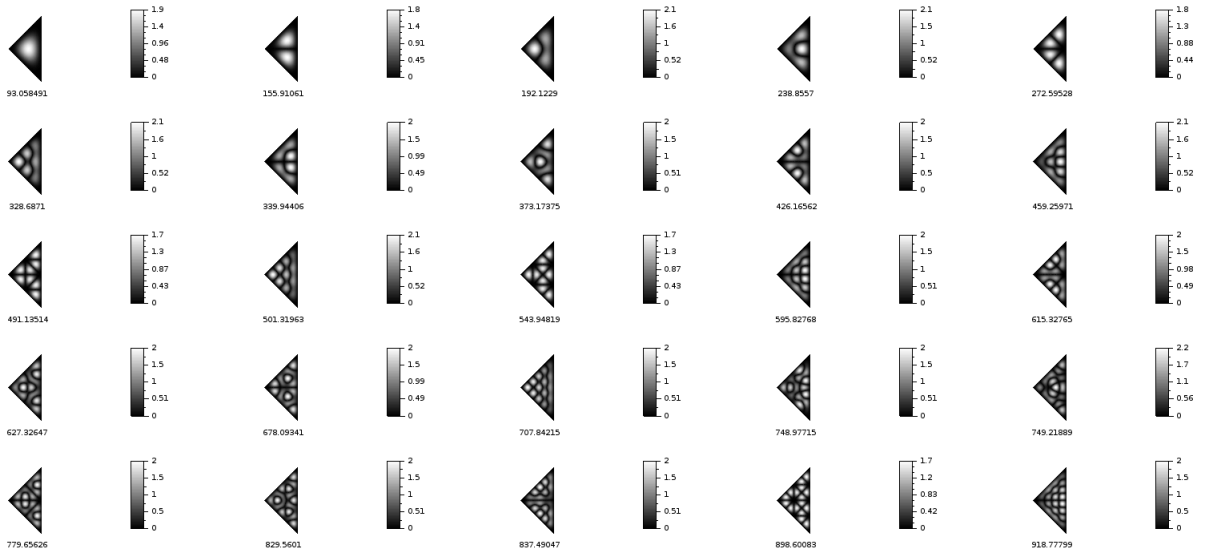


Figura 4.24: *Autofunções/linhas nodais. Placa triangular. Formulação não conforme.*

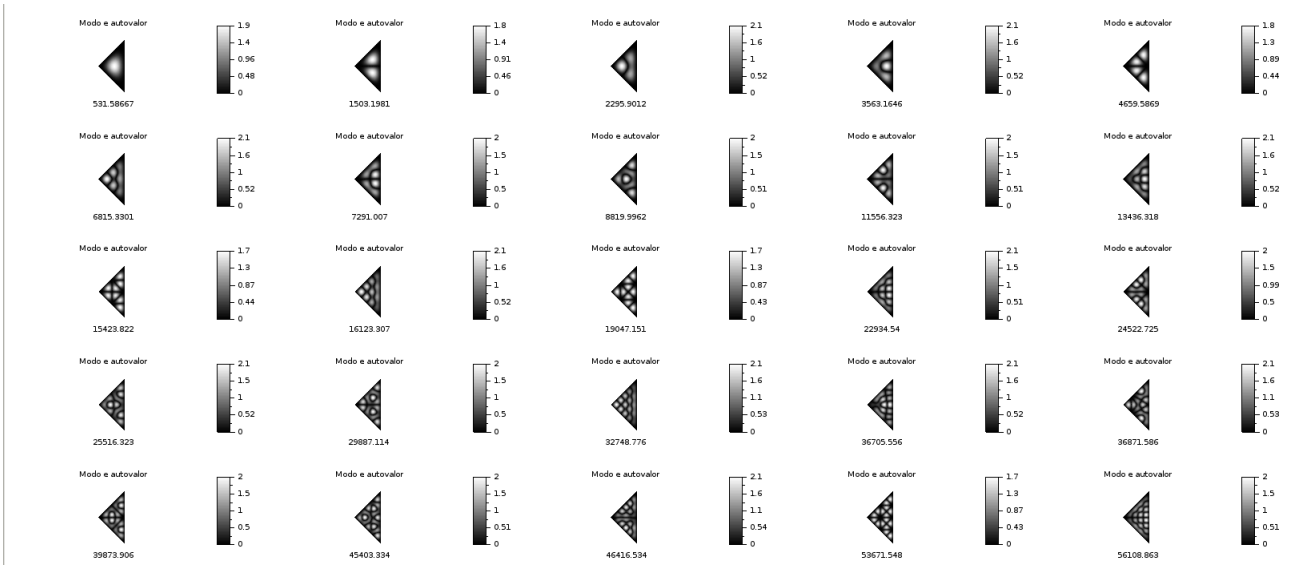


Figura 4.25: *Autofunções/linhas nodais. Placa triangular. Formulação conforme. Aqui, em particular, abaixo de cada figura temos o autovalor correspondente.*

muito. Chamaremos esse efeito de *ressonância*. Fisicamente, ressonância é usualmente definida como o estado de um sistema que vibra em frequência própria (natural) com amplitude acentuadamente maior, como resultado de estímulos externos de mesma frequência que a frequência natural deste sistema. Nos apropriamos desta terminologia no sentido de considerar a combinação de modos de vibração (nossas autofunções) associados a uma mesma frequência (autovalor).

Esse estudo será feito com base nos resultados numéricos obtidos para placas retangulares. Vale ressaltar que, por se tratar de uma aproximação, não obtivemos autovalores exatamente iguais. Usaremos aqui, como já temos feito em seções anteriores, os coeficiente de frequência como representantes dos autovalores. Além disso, nosso estudo será limitado à lista dos 25 primeiros modos de vibração. O objetivo aqui não é saber quantas ressonâncias ocorrem no total, mas identificar possíveis casos presentes nesse conjunto de dados e as possíveis combinações que ocorrem.

Esse estudo foi inspirado no fenômeno equivalente, observado nas linhas nodais provenientes da

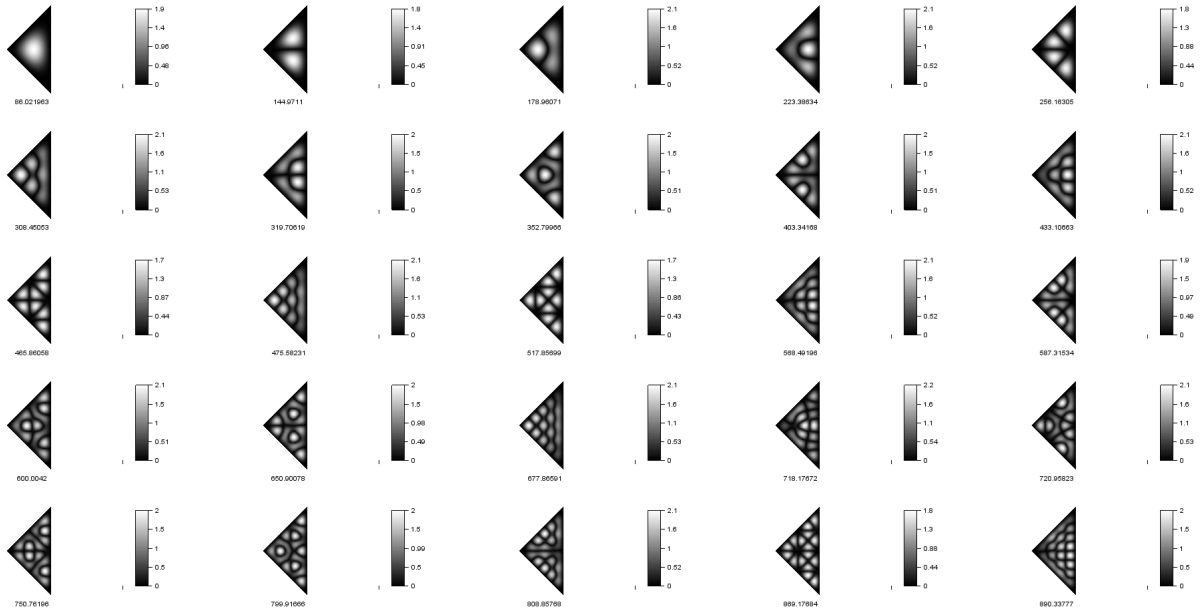


Figura 4.26: *Autofunções/linhas nodais. Placa triangular. Formulação conforme18.*

Malha quadrada de lado 1.0				
Coeficientes de frequência para os diferentes valores do parâmetro				Valores de referência
0.0	0.3	0.6	0.9	disponíveis - A. Leissa
35.84116667	35.77696973	35.60616422	34.45828801	**35.9866**
72.88870799	72.66790869	72.08198303	68.27718633	
72.89667935	72.67622477	72.09383394	68.31378482	**73.40 ou 73.41**
107.1844194	106.7276606	105.5353925	98.12652489	**108.22 ou 108.27**
130.1323503	129.5071733	127.8614712	117.6628926	**131.64**
130.7946493	130.1867144	128.5856537	118.6469847	**132.18 ou 132.25**
162.7902445	161.8279	159.3216542	144.4450138	**164.99 ou 165.15**
162.7983303	161.8360356	159.3546683	144.5945295	
207.206373	205.7967033	202.127136	180.6035243	
207.2465698	205.8538461	202.2192744	180.8283156	**210.35**
216.2510002	214.6218487	210.436163	186.7101328	**219.32**
237.7528247	235.8688665	231.033081	203.6539753	
238.8325656	236.9646418	232.1711255	205.0530468	
289.8726618	287.1359041	280.1637226	242.4062898	
289.9365838	287.1858737	280.2295801	242.6684576	**295.69**
302.3433137	299.5797654	292.4707833	253.0548159	
302.5911858	299.8166692	292.6788842	253.2267624	
332.5445306	329.1412083	320.4936327	274.1656064	
332.640544	329.2664381	320.6960191	274.6546329	
361.6812658	357.5619679	347.263031	294.1129842	**370.66**
382.038613	377.5581662	366.2758337	308.2171981	
383.429061	378.9877515	367.8473008	310.3701927	
415.3398363	410.4076017	397.8374918	332.5556377	
415.4085738	410.4634151	397.9265739	332.8482133	
444.4791244	438.7572739	424.3831634	351.6404372	

Figura 4.27: *Lista dos 25 primeiros coeficientes de frequência de uma placa quadrada, com 3826 elementos, para diferentes valores de ν. Comparação dos resultados obtidos com a variação de ν com os valores de referência das tabelas 4.22 e 4.24 de [8].*

vibração de uma membrana retangular de dimensão $a \times b$ com o *bordo preso* (condição de contorno $u = 0$). Esse é um problema de autovalor cuja solução analítica é possível e conhecida: as autofunções

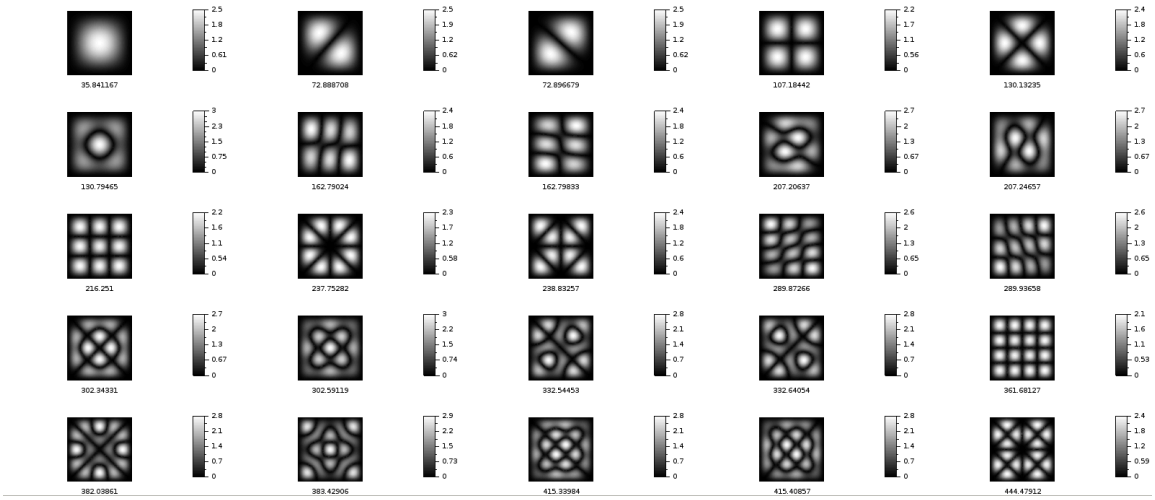


Figura 4.28: *Primeiras 25 autofunções/linhas nodais.* $\nu = 0.0$.

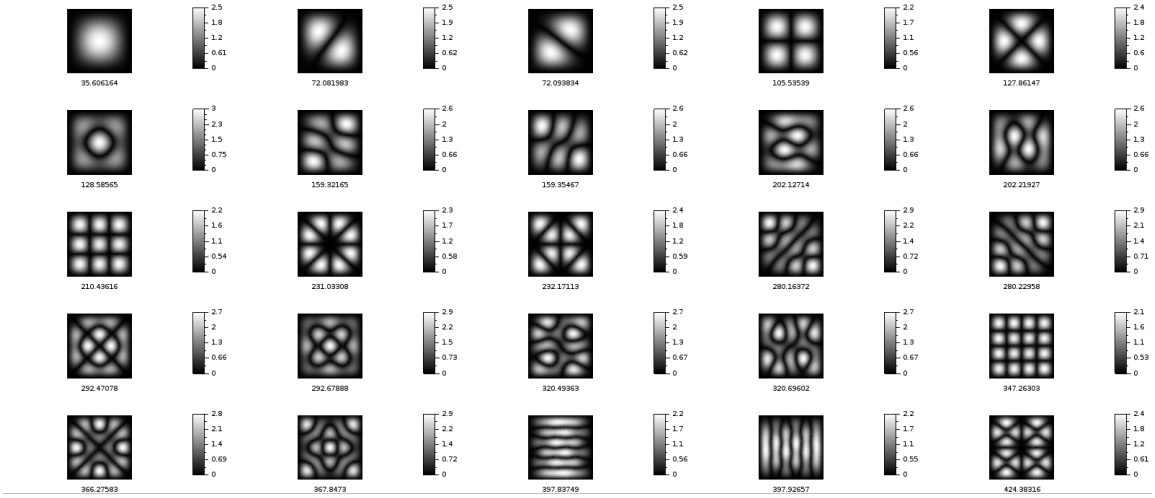


Figura 4.29: *Primeiras 25 autofunções/linhas nodais.* $\nu = 0.6$.

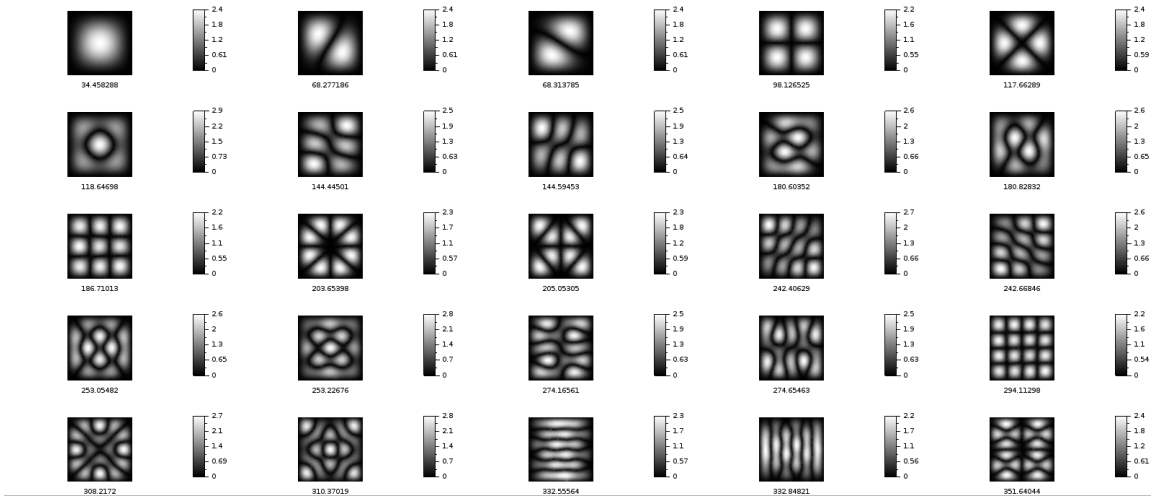


Figura 4.30: *Primeiras 25 autofunções/linhas nodais.* $\nu = 0.9$.

$u(x, y)$ são dadas pelo produto $\sin(\frac{n\pi x}{a})\sin(\frac{m\pi y}{b}), m, n \in \mathbb{Z}$ e, existe uma relação entre os autovalores λ, as dimensões a e b da membrana e os parâmetros m e n do modo de vibração (m, n): $\frac{\lambda}{\pi^2} =$

$\left(\frac{m^2}{a^2} + \frac{n^2}{b^2}\right)$. Além disso, é possível mostrar que as linhas nodais (curvas no plano para as quais $u(x, y) = 0$) são retas paralelas horizontas e verticais [2].

Autovalores múltiplos sempre ocorrem quando a razão entre os lados a e b do domínio retangular for um número racional, caso em que é possível mostrar que a equação $\frac{n^2}{a^2} + \frac{m^2}{b^2} = \frac{n'^2}{a^2} + \frac{m'^2}{b^2}$ tem soluções não triviais para m, n, m' e n' [2]. Se a placa for quadrada, i.e., se $a = b$, então a relação se simplifica de modo que o que se busca é saber de quantas maneiras é possível escrever um número k^2 como a soma de dois quadrados: $p^2 + q^2 = k^2$. Por exemplo, os pares $(m, n) = (1, 2)$ e $(m', n') = (2, 1)$ satisfazem essa equação para $k = \sqrt{5}$ e, portanto, são modos de vibração associados ao mesmo autovalor λ, que é tal que $k = \frac{\sqrt{\lambda}}{\pi}$. Entretanto, nem sempre esses dois modos ficam visíveis dessa forma pois quando há combinação de modos, outras linhas nodais – diferentes de linhas verticais e horizontais – podem ocorrer [2]. Assim, sempre que curvas diferentes daquelas esperadas ocorrem podemos dizer que elas são resultados de uma combinação linear. A Figura 4.31 ilustra isso. Ela contém alguns zeros da combinação linear $\alpha \sin(mx)\sin(nx) + \beta \sin(nx)\sin(mx),\ \alpha, \beta \in \mathbb{R}$ para uma placa quadrada.

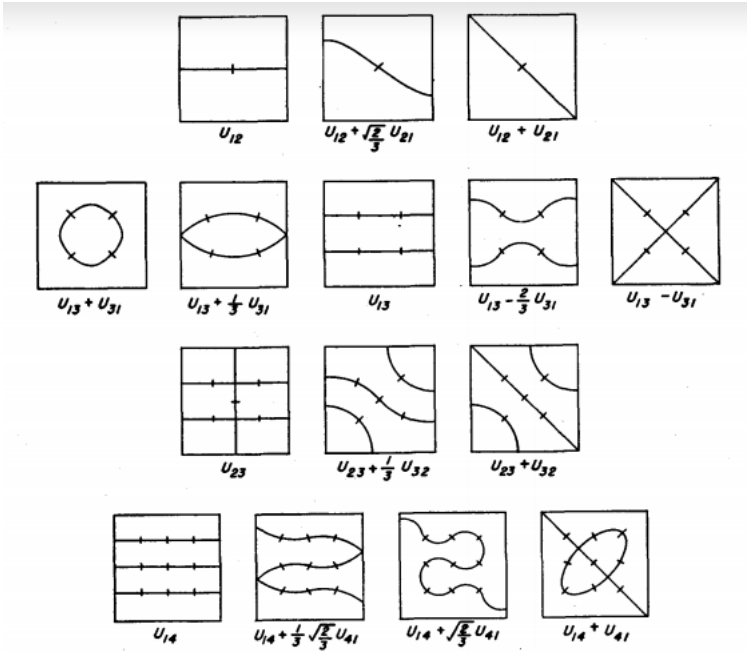


Figura 4.31: *Algumas linhas nodais não usais do problema de vibração da membrana quadrada de bordo preso. São resultado de combinações lineares de combinações lineares autofunções. u_{mn} indica o modo (m, n) e portanto a função $\sin(mx)\sin(nx)$. Fonte: livro Methods of Mathematical Physics R. Courant e D. Hilbert [2].*

No caso que estudamos nesse trabalho, da vibração de uma placa com o bordo grampeado, não temos posse da solução analítica e, portanto, não há como saber, previamente, as características das linhas nodais. Entretanto, nossos resultados para placas retangulares mostram um padrão de linhas horizontais e verticais paralelas como aqueles da membrana, o que nos motivou, apesar de o problema da membrana ser de outra natureza, a tomar essas linhas como base do que seria esperado para as linhas nodais da placa grampeada. Chamaremos os modos de vibração com essa característica de *modos puros*. Nosso programa fornece, como resultado, autovalores e suas autofunções associadas, mas ele não distingue ou seleciona uma autofunção específica associada a um mesmo autovalor, i.e., não temos controle sobre que tipo de autofunção (relativa a um modo puro ou uma combinação)

aparecerá como resultado. No entanto, sabendo que esse fenômeno de ressonância pode ocorrer, foi feita uma investigação visual a fim de identificar modos de vibração associados a um mesmo autovalor e combinações lineares de modos puros associados a um mesmo autovalor. A ideia aqui não é saber exatamente como os modos foram combinados, mas apenas identificá-los. Para isso, a Figura 4.31, apesar de ser relativa a um problema diferente, pode ajudar na interpretação das curvas que aparecem no problema da placa.

Uma forma de começar a investigação por ressonâncias é buscar, na lista dos parâmetros de frequência de cada placa, valores muito próximos. Como a nossa lista de coeficientes de frequência está organizada em ordem crescente, então coeficientes de frequência próximos são consecutivos. Por exemplo, a placa quadrada de dimensão 1.0 possui muitos coeficientes de frequência próximos que destacamos em azul na tabela 4.4.

25 primeiros coeficientes de frequência normalizados - placa quadrada					
1º	35.72774893	9º	204.696752	17º	297.6505649
2º	72.48367519	10º	204.7840985	18º	326.1088889
3º	72.50355581	11º	213.3529428	19º	326.53636
4º	106.3358756	12º	234.2818675	20º	354.3250514
5º	129.0566273	13º	235.4477953	21º	373.8409449
6º	129.6812386	14º	284.9497531	22º	375.390582
7º	160.9867224	15º	285.0669003	23º	406.1667518
8º	161.0921438	16º	297.1960939	24º	407.1553746
				25º	433.8572598

Tabela 4.4: *25 primeiros coeficientes de frequência normalizados da placa quadrada.*

Observamos nas imagens da Figura 4.32 as linhas nodais das autofunções relativas a esses coeficientes de frequência próximos e percebemos linhas curvas, diferente da esperadas para os modos puros. Veremos mais adiante que nem sempre valores próximos indicam linhas curvas e ressonâncias e que a placa quadrada pode ser considerada como um caso peculiar por apresentar tantos casos de ressonância. A busca por linhas curvas também pode iniciar uma busca por candidatos à ressonância. Lendo as figuras da esquerda para a direita e de cima para baixo, identificamos, pela ordem em que aparecem e, além daqueles indicados na tabela, os modos de vibração 7 e 8, 12 e 13, 21 e 22 e, 23 e 24, que podem indicar uma combinação linear de autofunções relativas a um mesmo coeficiente de frequência. Temos então 10 candidatos, obtidos por meios de observação diferentes.

O gráfico 4.33 pode usado para buscar autovalores próximos e sua vantagem é que ele contém informação de placas retangulares de diferentes proporções.

Esse gráfico foi construído com informações de 17 placas retangulares de dimensão $c \times 1.0$, com $1 \leq c \leq 3$, variando de 0.125 em 0.125. As imagens das linhas nodais para cada uma dessas placas estão disponíveis no Apêndice A.

Cada cor da Figura 4.33 representa uma posição na lista ordenada de autovalores. Fixada uma cor, cada ponto corresponde a um valor de c. Esses pontos foram ligados, formando linhas que nos ajudam a perceber valores próximos por meio do encontro de linhas de diferentes cores. Para a placa quadrada, verificamos os encontros na reta vertical c=1.0, onde as linhas que representam os coeficientes de frequência citados anteriormente se encontram: 2 e 3, 5 e 6, 7 e 8, 9 e 10.

Na reta vertical c =1.25 observamos o encontro das linhas verde escuro e roxa, representantes dos modos de número 4 e 5. Para c=1.5, temos o encontro das linhas amarela e verde escuro,

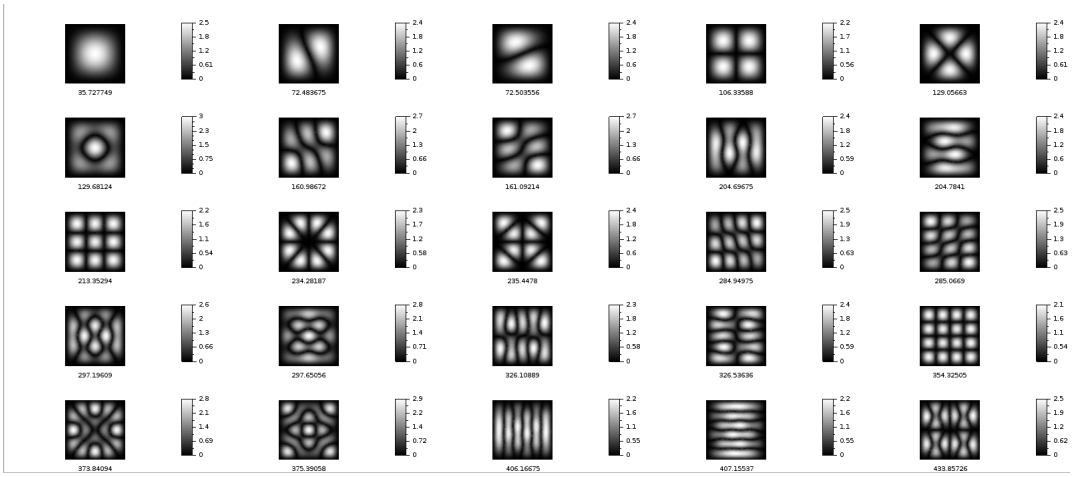


Figura 4.32: *Autofunções da placa quadrada de dimensão 1.0.*

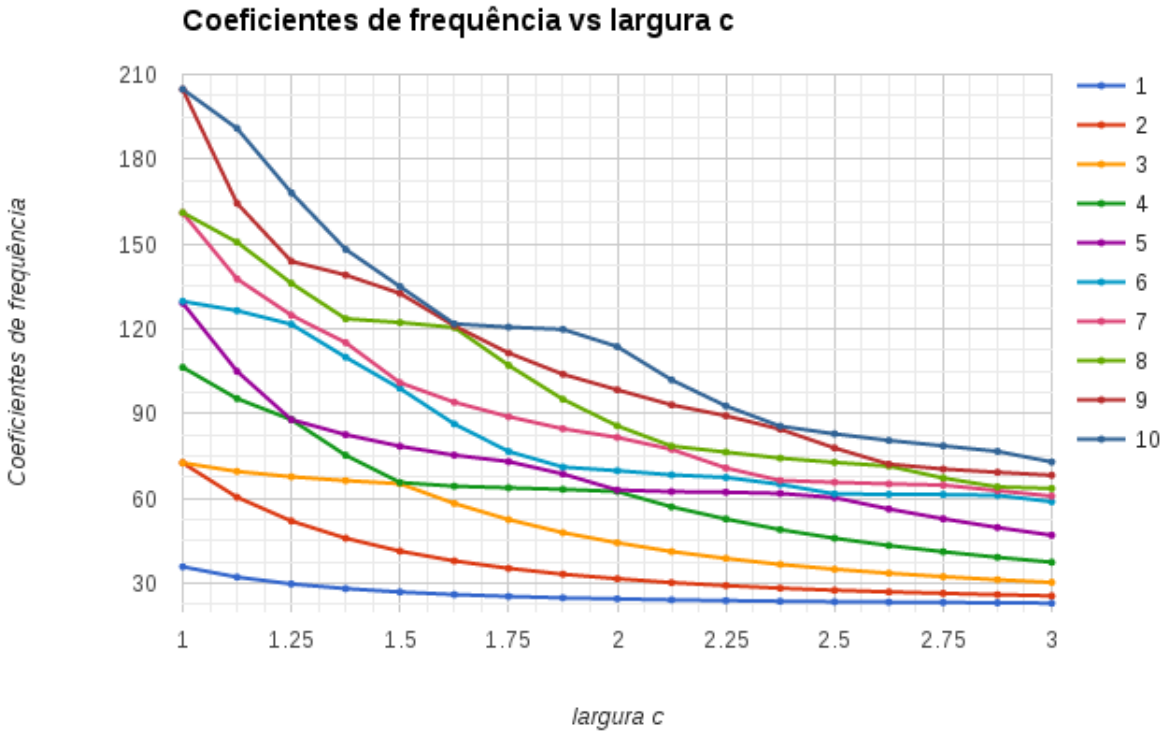


Figura 4.33: *Evolução dos 10 primeiros coeficientes de frequência. Placas de dimensão* $1.0 \times c$*, com* $1 \leq c \leq 3$*. Foram feitas 17 malhas, cada uma com 3000 elementos triangulares, em média.*

representantes dos modos de número 3 e 4; para c=1.625, o encontro das linhas 8, 9 e 10 e, para a reta vertical c=2.0, observamos novamente o encontro das linhas verde e roxa, dos modos 4 e 5. A partir desses candidatos podemos observar os modos de vibração relativos a eles nas imagens 4.35, 4.36, 4.37 e 4.38, respectivamente, e confrontar se esses modos são aparentemente puros ou combinados. Na Figura 4.35, relativa à c=1.250, os modos número 4 e 5 parecem ser uma combinação linear, no sentido que eles apresentam linhas curvas. Na Figura 4.36, relativa à c=1.5, identificamos os modos listados como (1,2) e (3,1), respectivamente, na Figura 4.37, relativa à c=1.625, o 8º modo foi identificado como (4,2) e os modos 9º e 10º parecem ser uma perturbação – que aqui tanto pode

ser uma ressonância como uma imprecisão numérica que influenciou na qualidade da figura – dos modos puros (1,3) e (5,1) e, na Figura 4.38, identificamos os modos listados como (4,1) e (1,2).

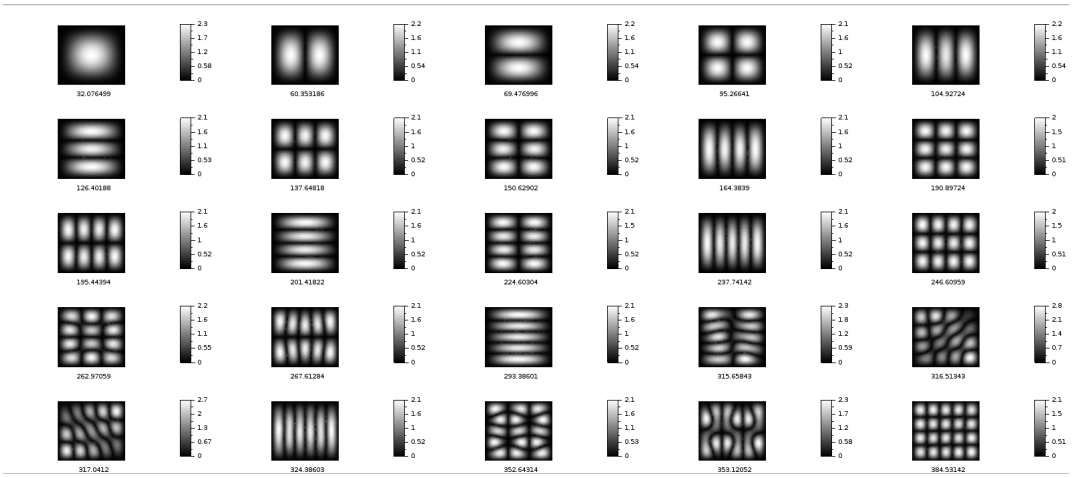


Figura 4.34: *Primeiras 25 autofunções/linhas nodais. Placa retangular de dimensão* $c \times 1.0$*, com* $c = 1.125$*.*

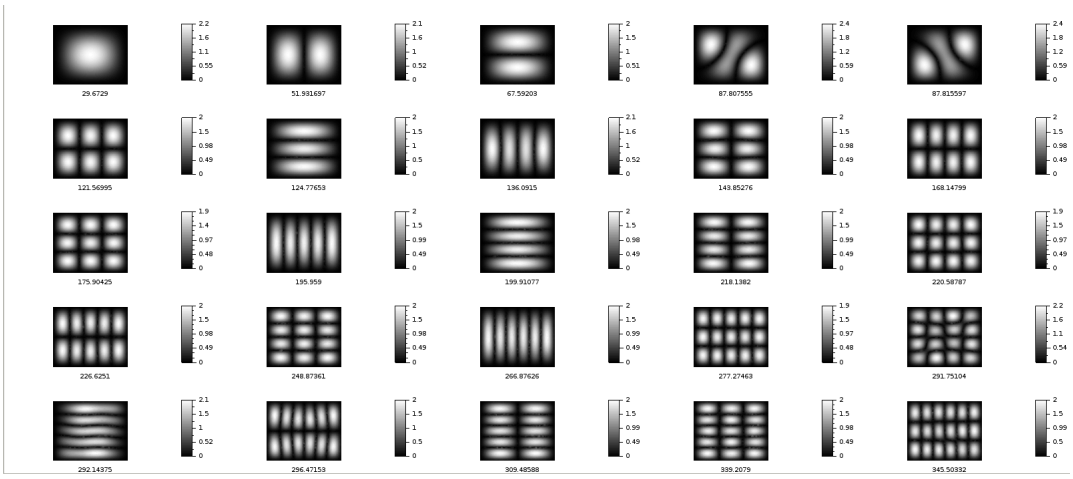


Figura 4.35: *Primeiras 25 autofunções/linhas nodais. Placa retangular de dimensão* $c \times 1.0$*, com* $c = 1.250$*.*

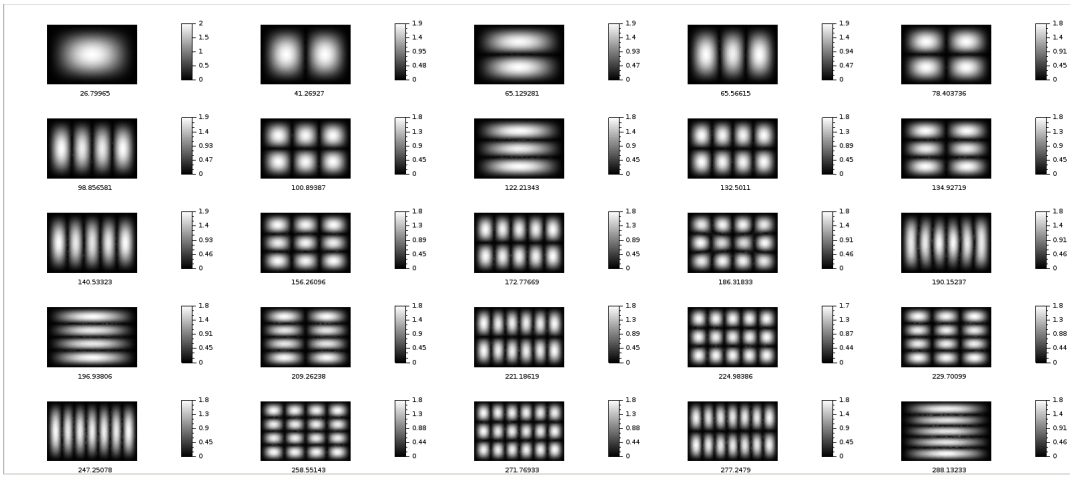


Figura 4.36: *Primeiras 25 autofunções/linhas nodais. Placa retangular de dimensão* $c \times 1.0$*, com* $c = 1.5$*.*

Apesar de, na busca por linhas curvas encontrarmos modos do tipo (m, n) bem definidos, isso

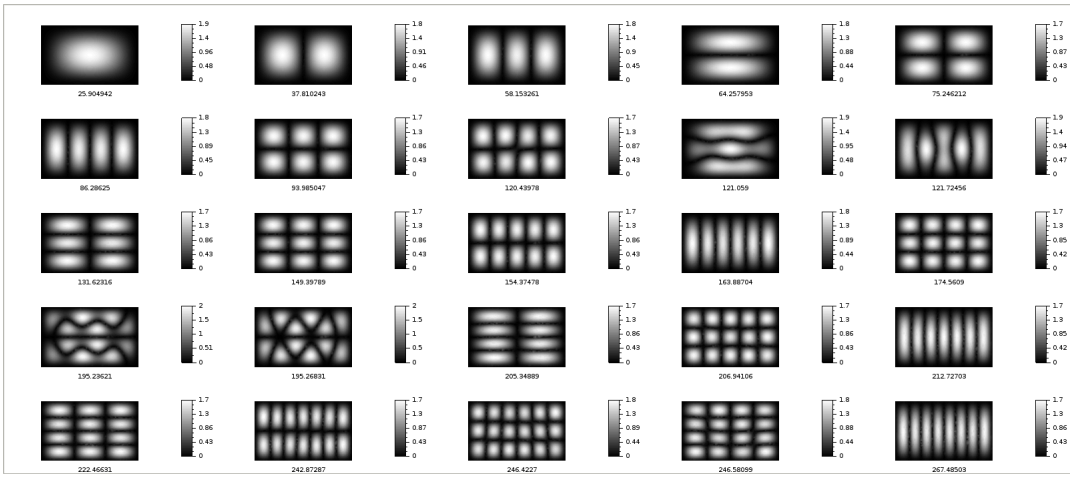


Figura 4.37: *Primeiras 25 autofunções/linhas nodais. Placa retangular de dimensão* $c \times 1.0$*, com* $c = 1.625$*.*

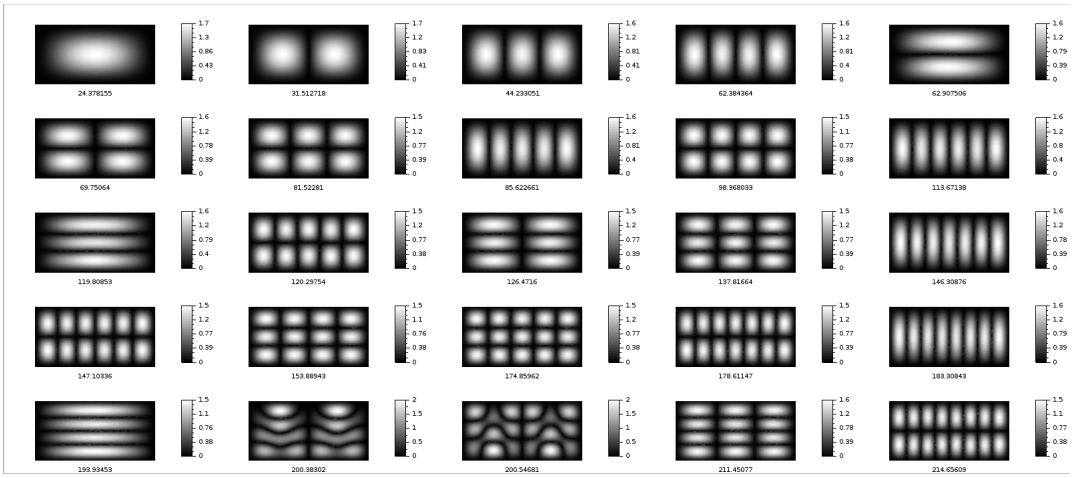


Figura 4.38: *Primeiras 25 autofunções/linhas nodais. Placa retangular de dimensão* $c \times 1.0$*, com* $c = 2.0$*.*

não é um problema. Quando olhamos para a evolução dos modos de vibração com a variação da proporção da placa, os casos em que os valores próximos não parecem ser ressonância (i.e., estão bem definidos) nos ajudam a identificar modos envolvidos numa combinação que aparece em alguma outra placa.

Para visualizar melhor as possíveis combinações dos modos de vibração, construímos outro tipo de gráfico. Na Figura 4.39, a evolução dos coeficientes de frequência com a variação da proporção da placa retangular se organiza por modo de vibração e não por ordem em que aparecem. Fixada uma cor, cada ponto do gráfico corresponde a um valor de c. Esses pontos também foram ligados de modo que cada curva colorida corresponde a um modo de vibração (m, n). As curvas com mesmo valor de n tem a mesma cor base, mas com a intensidade aumentando com o valor de m. Cruzamentos dessas curvas indicam que os modos de vibração possuem o mesmo coeficiente de frequência e por isso, nos permite não só localizar as ressonâncias mas também os modos de vibração envolvidos. Os dados para gerar esse gráfico foram organizados manualmente, com base na identificação dos modos de vibração feita pela observação das imagens das linhas nodais de cada placa – como fizemos nos exemplos anteriores. Além disso, para visualizar melhor os cruzamentos é necessário uma escolha adequada dos modos que serão plotados. Escolhemos os modos (m, n), $1 \leq m, n \leq 4$, para serem plotados.

A construção foi feita de maneira "empírica", por observação e identificação e interpretação dos modos de vibração nas imagens das linhas nodais. Situações ambíguas – i.e. modos de vibração mais complicados de distinguir por se tratar de combinação de modos puros – foram completadas "na mão". Por exemplo: os modos (3,1) e (2,2) não estão claros na placa c =1.25, mas na placa c=1.125 e c=1.5 eles aparecem claramente e em ordem diferentes. Eles são o 5º e o 4º para a placa c=1.125 e os 4º e 5º modos na placa c=1.500. Veja Figuras 4.34 e 4.36. Supondo que os coeficientes de frequência variem continuamente em função de c, podemos dizer que essa troca de posição nos indica que para algum valor de c entre 1.125 e 1.5 esses dois modos tiveram coeficientes de frequência iguais, que podem ter gerado uma combinação destes. Isso pode ser notado na placa $c = 1.25$ (Figura 4.35), pela imagens dos modos 4 e 5, para as quais inferimos que se trata de uma combinação dos modos (3,1) e (2,2). Assim, completamos o gráfico para esses modos com um ponto preto, utilizando uma média dos coeficientes de frequência dos modos 4 e 5 da placa c=1.250. A cor preta indica a ambiguidade na identificação dos modos puros nessa placa.

A partir do exemplo acima, elegemos as principais características do nosso gráfico (Figura 4.39).

Os pontos pretos nos cruzamentos indicam que a cor está indefinida, pois os coeficientes de frequência de dois modos estão muito próximos. Pontos pretos com um quadrado cinza em volta indicam que houve uma combinação linear de modos. Se essa combinação for de modos puros (m, n) com $1 \leq m, n \leq 4$, não há nada escrito em volta. Caso contrário, havendo um modo com n ou m maior do que 4 envolvido na ressonância, anotamos ao lado qual é esse modo.

Temos três pontos pretos sem o quadrado cinza em volta: do encontro das linhas (4,3) e (3,4) para $c = 1.0$, das linhas (3,1) e (1,2) para $c = 1.5$ e das linhas (4,1) e (1,2) para $c = 2.0$. Para esses pontos, os modos puros aparecem nitidamente nas imagens, mas os coeficientes de frequência são tão próximos que não os distinguimos nesse gráfico e no gráfico da Figura 4.33.

Quando aparecem modos não puros ou quase puros mas um pouco "perturbados", colocamos, junto ao ponto, uma figura correspondente aos modos que correspondem àquele autovalor. Na maioria dos casos, identificamos dois modos não puros. Mas há um caso diferente, em que aparecem três modos, e que já tinha sido identificado no gráfico da Figura 4.33. Trata-se dos modos 8, 9 e 10 da placa c=1.625. Um deles, o (4,2), está bem claro, mas os outros dois, o 9º e o 10º, parecem ter sofrido uma "perturbação" dos modos (1,3) e (5,1).

As curvas incompletas no gráfico correspondem a modos cujos autovalores – ou coeficientes de frequência – saíram do conjunto dos 25 primeiros (e portanto menores) que estamos apresentando. Também é interessante como a escolha que fizemos dos modos plotados (isto é,$1 \leq n, m \leq 4$) permite ver como aqueles que têm o mesmo valor de n vão se juntando à medida que c cresce.

Em resumo, esse gráfico é uma forma de apresentar uma interpretação dos nossos resultados principais – as linhas nodais da vibração – pela maneira como ele foi construído. E, depois de pronto, ele nos ajuda a inferir ressonâncias em placas não preestabelecidas aqui, i.e. para valores de c entre os valores utilizados. Se considerarmos que o autovalores (e portanto os coeficientes de frequência) variam continuamente em função de c, não é preciso se limitar a olhar cruzamentos nas linhas verticais para os valores de c pré-estabelecidos, afinal, não é só nesses pontos que pode haver cruzamentos.

Usando os valores de c como parâmetro listamos os seguintes cruzamentos: próximo à reta c=1.250 : modos (3,2) e (1,3); (2,3) e (4,1); (3,3) e (4,2); (2,4) e (4,3). Próximo à reta c=1.375: (4,1) e (1,3); (1,4) e (4,3). Próximo a reta c=1.5: (4,1) e (3,2); (4,2) e (2,3). Entre as retas c=1.750

e c=2.0: modos (4,1) e (2,2). Aqui, a continuidade é importante para poder afirmar que existe um valor de c entre os valores tabelados para o qual os autovalores são iguais. Isto posto, para uma investigação mais completa, seria necessário descobrir com mais precisão de c para o qual os modos puros se cruzam, construir uma placa com essa proporção e obter as imagens.

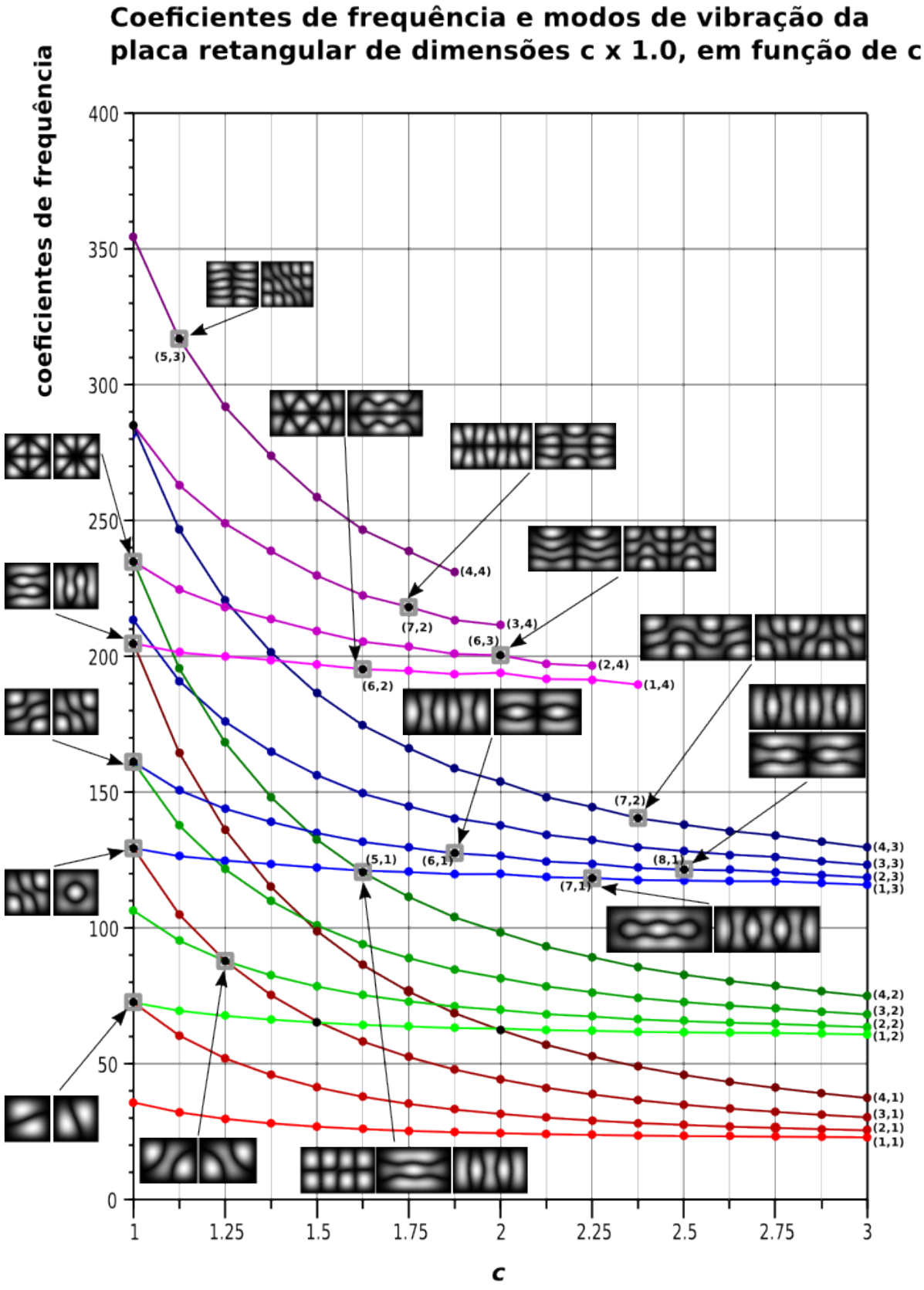


Figura 4.39: *Ressonâncias. Placas de dimensão* $c \times 1.0$*, com* $1 \leq c \leq 3$.

Capítulo 5

Conclusão

Neste capítulo apresentaremos as principais conclusões que obtivemos ao longo deste trabalho.

O objetivo de obter um software capaz de fornecer dados para obtenção de imagens das linhas nodais da vibração de placas de diferentes formatos poligonais foi cumprido. Implementamos e obtivemos resultados para três formulações: a aproximação polinomial quadrática *não conforme*, e duas aproximações polinomiais de quinto grau conformes, que chamamos de *conforme* e *conforme18* e, com as quais foi possível gerar as imagens pretendidas, como vimos por exemplo, nas figuras do Capítulo 4.

Esperávamos que as formulações conformes nos fornecessem os melhores resultados devido à sua característica de aproximar "por dentro" do espaço de funções admissíveis. Entretanto foi a quadrática, não conforme e mais simples, que apresentou resultados mais satisfatórios no sentido da convergência do primeiro coeficiente de frequência para o valor de referência.

As formulações de quinto grau, *conforme* e *conforme18*, foram implementadas como uma alternativa à primeira implementação de quinto grau conforme – que apresentou uma instabilidade com relação aos refinamentos de malhas, caracterizada principalmente pela inversão da matriz C_i (cuja descrição foi dada no Capítulo 2) – para a qual não obtivemos resultados relevantes e que por isso não recebeu um nome especial. Tal formulação conforme já se mostrava mais dispendiosa em termos de tempo de execução e espaço de memória, devido ao alto grau da aproximação polinomial e às inversões de matrizes de grande dimensão e, isso não foi diferente para as duas alternativas propostas.

No caso da formulação *conforme*, observamos que ela se tornou mais estável com relação à variação das malhas, mas sua execução – por possuir várias etapas adicionais geradas pelas interpolações – ficou bastante demorada, cerca de 13 horas para uma malha de aproximadamente 3000 elementos triangulares.

A formulação *conforme18* possui uma implementação mais concisa, apesar da dificuldade teórica com a qual nos deparamos para cuidar da restrição da derivada normal como polinômio cúbico na aresta. Essa é uma solução que garante uma homogeneidade na distribuição das variáveis nodais no elemento finito, propriedade que pode ser relevante para o desempenho numérico, mas que tem desdobramentos que devem ser tratados com cuidado na implementação, como por exemplo a etapa da inversão de matrizes. Essa formulação tem um tempo médio de execução um pouco menor, se comparada com a anterior – cerca de 6 horas para uma malha com 3000 elementos – mas ela apresenta um problema de instabilidade com relação a algumas malhas. Acreditamos que essa instabilidade não seja do mesmo tipo daquela primeira formulação conforme, que motivou esta

implementação, pois ela não se dá com o refinamento da malha. Simplesmente há malhas para as quais o programa não pôde ser executado até o fim. Vimos na Tabela 4.2 que as malhas para as quais não obtivemos resultados não eram tão finas. Com base nisto, percebemos que há aspectos na implementação que ainda precisam ser melhorados.

Apesar dessas questões e limitações, ainda sim, essas duas formulações nos forneceram resultados suficientes para que a análise do Capítulo 4 pudesse ser feita. Conseguimos executar os programas para diversas malhas e refinamentos e as imagens obtidas também apresentam boa qualidade no sentido de representarem bem os modos de vibração, como vimos nas figuras 4.21, 4.22 e 4.21 do Capítulo 4, por exemplo.

Acreditamos que as duas implementações *conforme* e *conforme18* podem ser melhoradas em vários aspectos e, para isso vislumbramos possibilidades numa organização da implementação e otimização dos códigos que gere menos custo nas operações computacionais, ou no uso de uma base não canônica para os polinômios aproximadores que ajude na eficiência numérica e, talvez, no estudo do uso de um sistema de coordenadas não cartesianas.

O estudo cuidadoso que foi feito com relação à formulação da EDP desde a modelagem física, com a energia potencial, e da sua forma fraca permitiu entender melhor o problema, a EDP e as constantes nela envolvidas. Uma delas nos chamou muito a atenção: o coeficiente de Poisson ν. Vimos que ele está associado às propriedades físicas do material, que aparece na modelagem da energia potencial associada a este problema, na constante $D = \dfrac{Eh^3}{12(1-\nu^2)}$ e na formulação fraca do problema de autovalor. Vimos também que, para o caso especial em que a condição de contorno considerada é a de *bordo grampeado*, essa constante desaparece do problema de autovalor, de modo que o valor escolhido para ela na forma fraca é irrelevante para as soluções (autovalores e autofunções), apesar de ν continuar influenciando a frequência de vibração. Os testes realizados para diferentes valores do parâmetro ν também nos mostraram que essa influência é pequena e esta conclusão é confirmada pelas imagens das linhas nodais para os diferentes valores ν que usamos (Figuras 4.28, 4.29 e 4.30). As diferenças encontradas nos autovalores e, portanto, no coeficientes de frequência podem ser devidas apenas a erros numéricos naturais (erro de arrendondamento e erro do método).

Este fato nos levou à seguinte pergunta: será então, que, estabelecida/fixada a condição de contorno de *bordo grampeado*, os modos de vibração de uma placa de material elástico, isotrópico e homogêneo, são essencialmente os mesmos, para qualquer material com essas propriedades? Nossas considerações teóricas e resultados sobre a variação do ν nos dizem que sim. Esse fato é curioso e interessante do ponto de vista físico – pensando em como placas de diferentes materiais vibram ou devem ter o mesmo comportamento nas vibrações — e também do ponto de vista matemático, pois, uma vez fixada esta condição de contorno temos liberdade de alterar a forma fraca para uma mais conveniente matematicamente e mais fácil de lidar.

Algumas trabalhos da literatura propõem a solução deste problema de autovalor com formas fracas e funcionais, a serem minimizados, mais simples do que aquela que utilizamos, mas só depois deste estudo é que foi possível entender o porquê. Neste sentido, foi importante estudar o efeito da variação de ν, mesmo para o caso da placa grampeada, onde não esperávamos grandes variações.

Observamos também que as linhas nodais que obtivemos para a para a placa retangular são muito parecidas com aquelas provenientes da vibração da membrana com fronteira presa. Esse fato também nos pareceu curioso e interessante, pois a modelagem deste problema para a placa é muito

mais complicada que o caso da membrana e, a implementação numérica também segue essa mesma característica. Uma impressão que fica é que poderíamos ter usado o modelo da membrana, ou que não é preciso toda essa modelagem de placas finas metálicas. Mas isso não é verdade. Observamos que essa semelhança se dá no caso especial da placa e membrana retangulares. Para fazer tal afirmação seria preciso um estudo mais completo dessa semelhança entre os modos puros, tanto para as placas retangulares, como para outros formatos e uma busca pela relação entre autovalores, frequência e dimensão do domínio, como existe para a membrana.

O estudo da evolução dos autovalores conforme a variação da proporção de uma placa retangular nos permitiu visualizar e identificar combinações lineares de alguns modos de vibração, fenômeno que chamamos de *ressonância*. Apesar de ter sido construído através de uma interpretação dos modos de vibração nas imagens que obtivemos numericamente, o gráfico 4.39 é interessante e nos permite investigar além dos valores preestabelecidos para a proporção da placa.

Tendo em vista essas considerações, enxergamos possibilidades e investigações para trabalhos futuros. Além da implementação computacional para o problema com o bordo livre que caracteriza melhor o problema de Chladni – já manifesto anteriormente – vemos algumas outras, ainda no caso de condição de contorno *bordo grampeado*:

- nos valer de soluções conhecidas para a vibração de uma barra rígida, de material com as mesmas propriedades que a placa, para estudar as soluções do problema no caso da placa.

- investigar a relação entre os parâmetros dos modos de vibração da placa retangular (m, n), suas dimensões e os autovalores;

- estudar modos de vibração e sua semelhança com os modos de vibração da membrana para placas de outros formatos.

Trabalhamos com vários aspectos do problema: a modelagem, a teoria, a implementação e interpretação de resultados e, apesar de não termos nos aprofundado muito em cada uma dessas etapas, foi interessante poder trabalhar com um pouco de tudo e sentir a pesquisa de um tema “do início ao fim”, isto é, desde a motivação experimental até resultados visuais computacionais.

Apêndice A

Figuras: Modos de vibração de placas retangulares

Cada figura a seguir contém as imagens dos 25 primeiros modos de vibração para placas retangulares de tamanho $1 \times c$ – com $1 \leq c \leq 3$ e c variando de 1.125 em 1.125 – que foram usadas para a construção do gráfico 4.39 de ressonâncias do capítulo 4. O número abaixo de cada uma das 25 imagens da figura é o coeficiente de frequência associado ao modo de vibração.

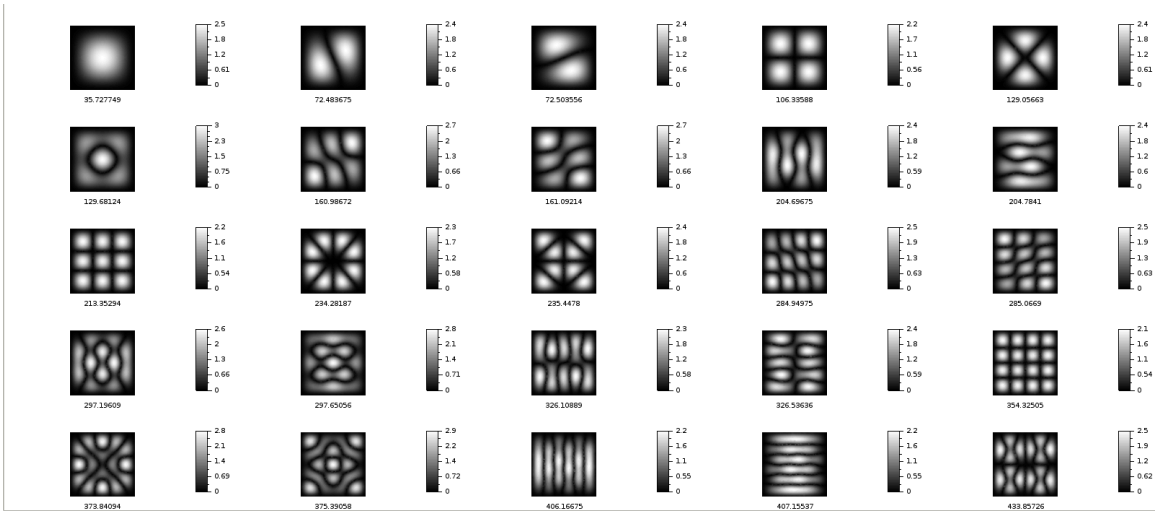


Figura A.1: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 1.0$.

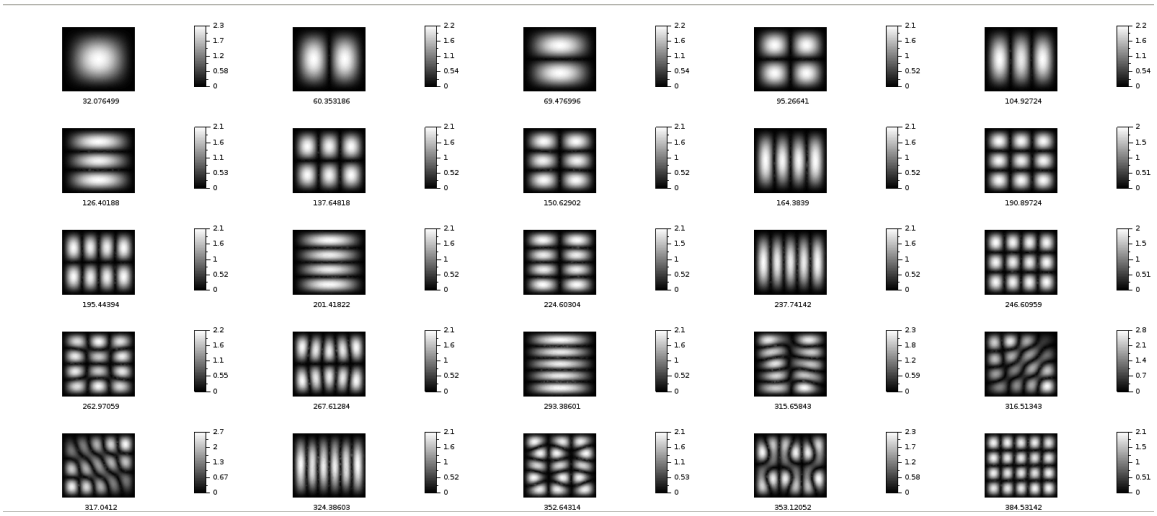


Figura A.2: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 1.125$*.*

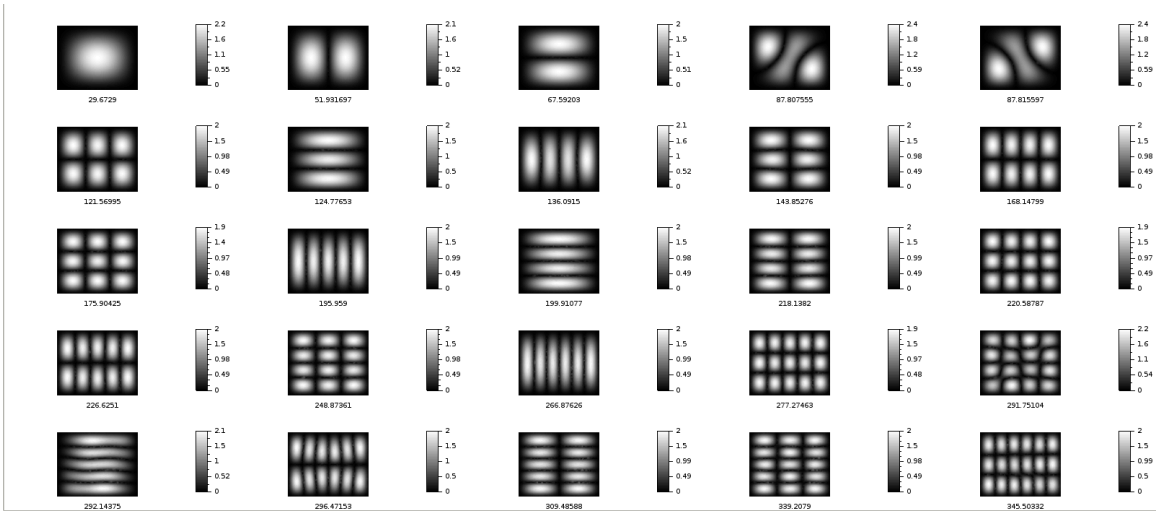


Figura A.3: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 1.250$*.*

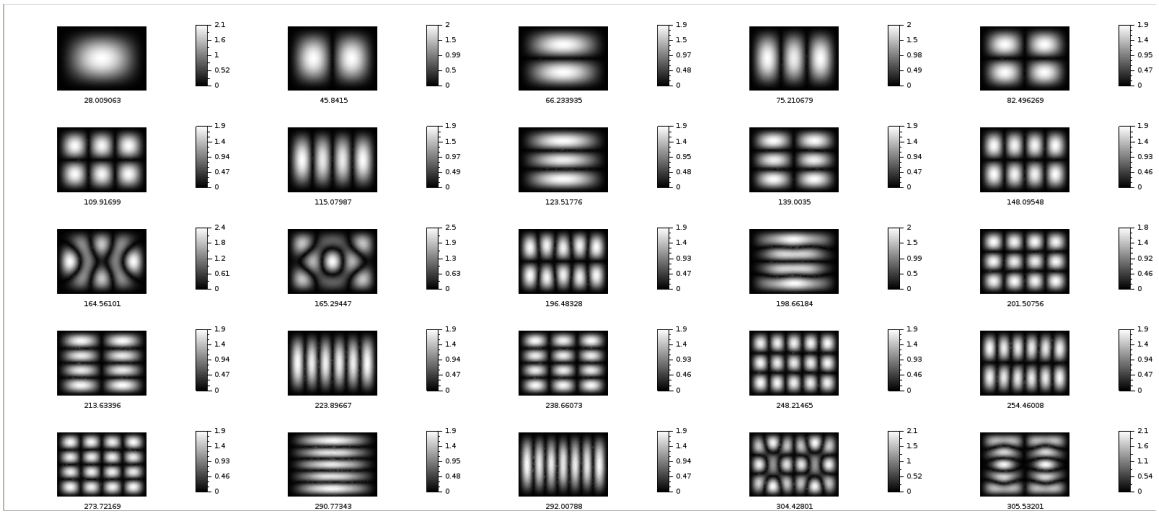


Figura A.4: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 1.375$*.*

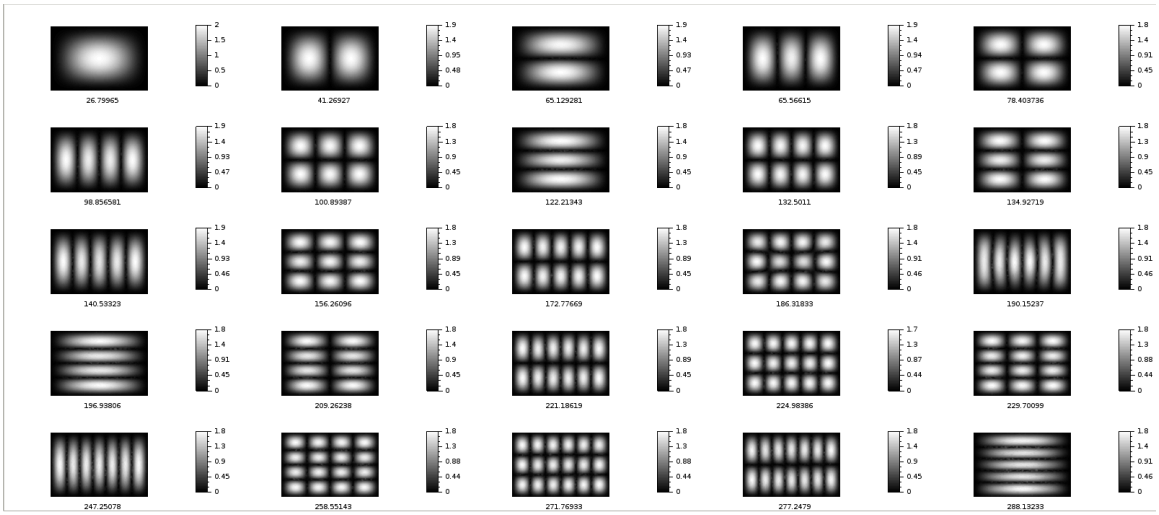


Figura A.5: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 1.5$*.*

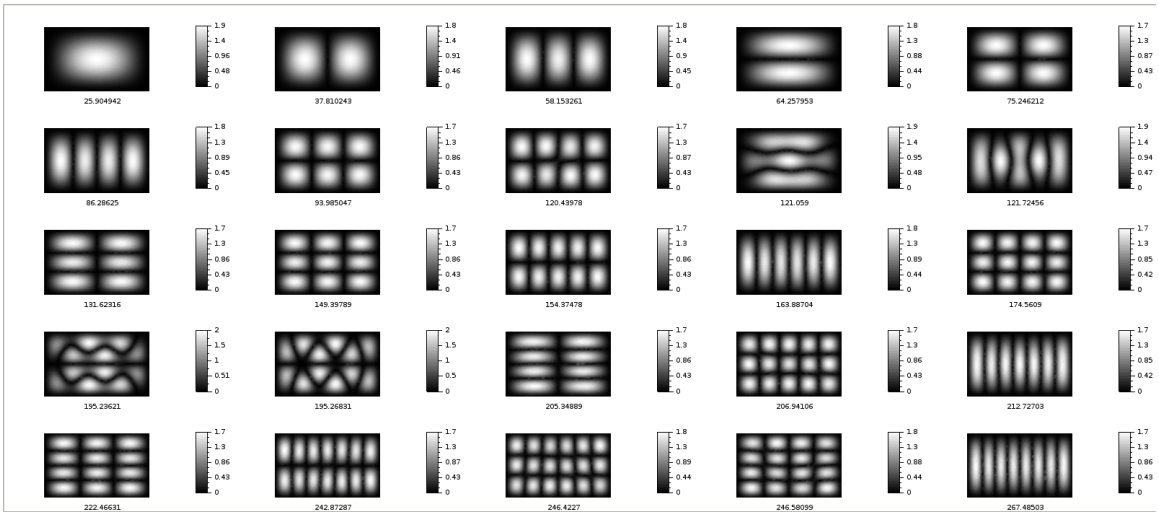


Figura A.6: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 1.625$*.*

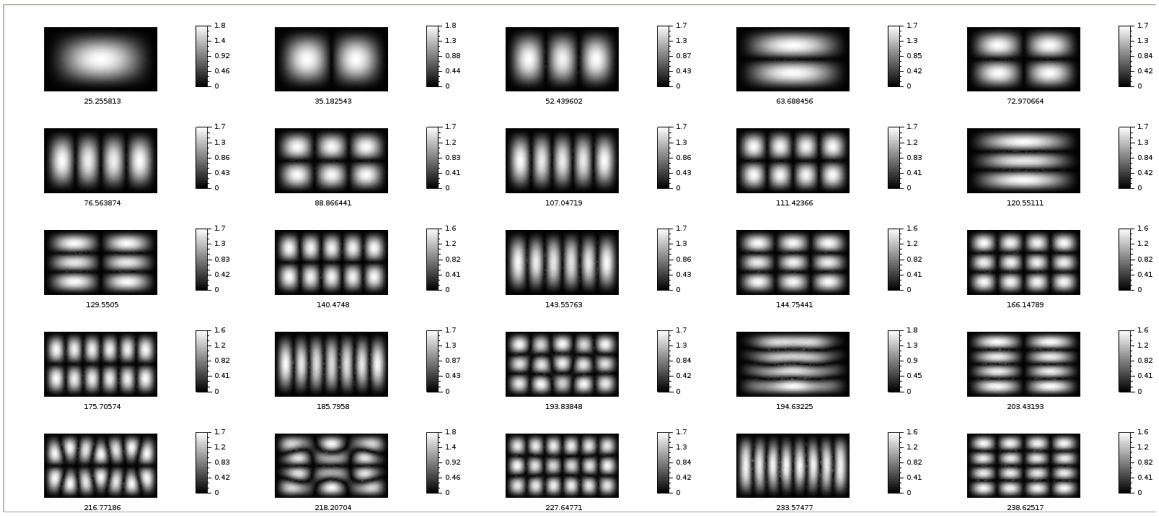


Figura A.7: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 1.750$*.*

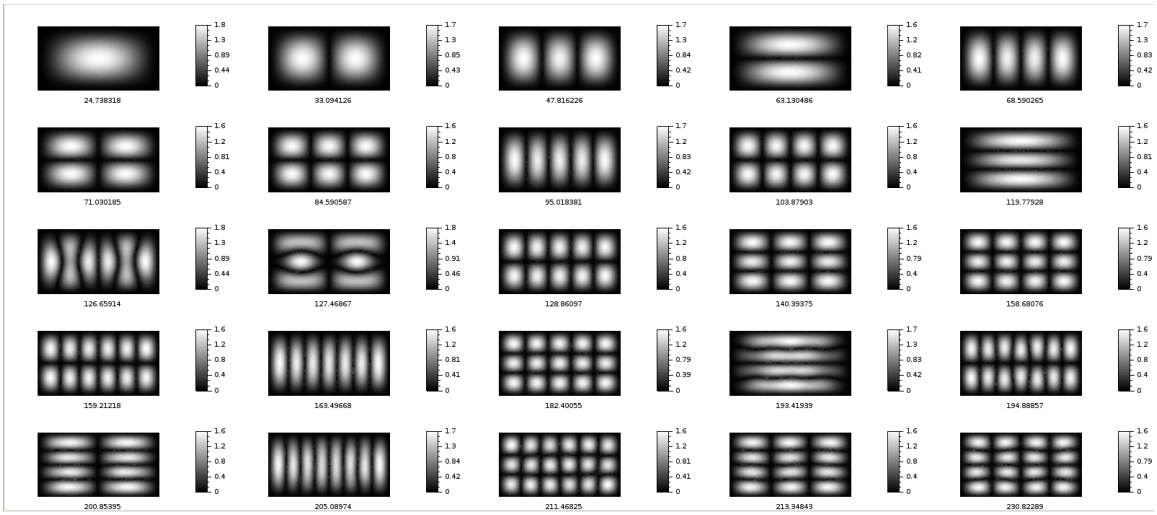


Figura A.8: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 1.875$*.*

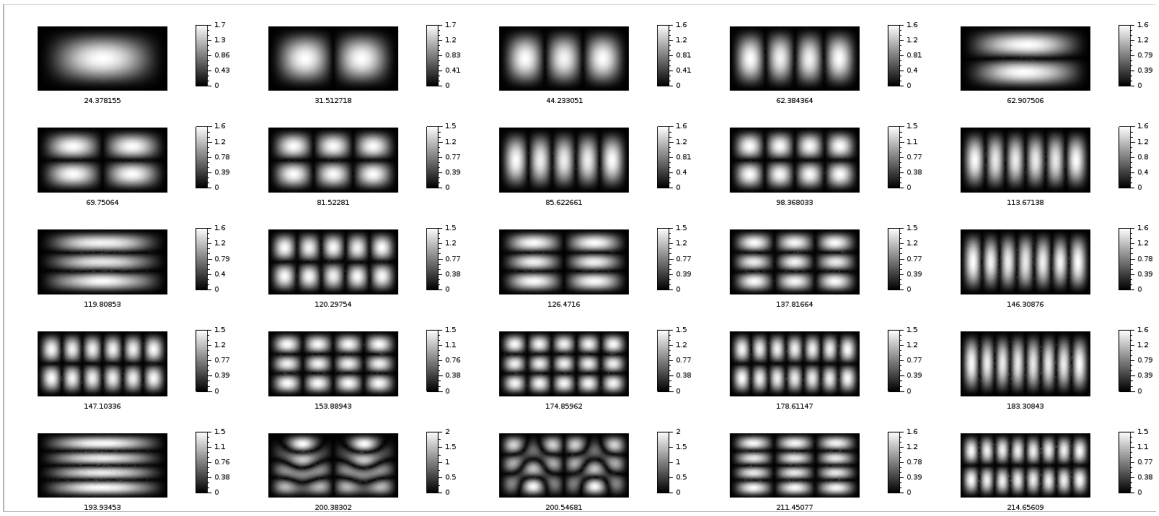


Figura A.9: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 2.0$*.*

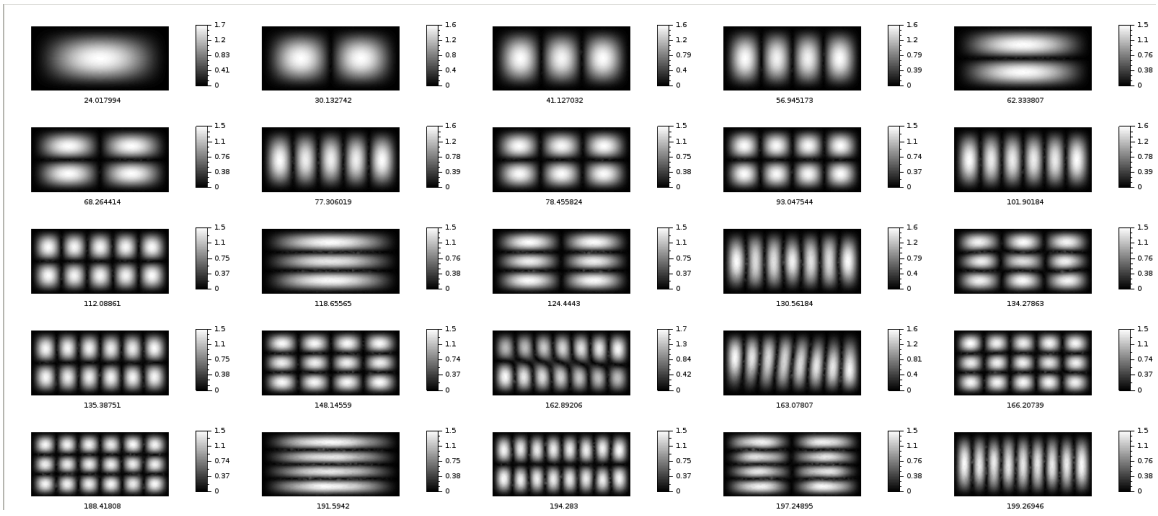


Figura A.10: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 2.125$*.*

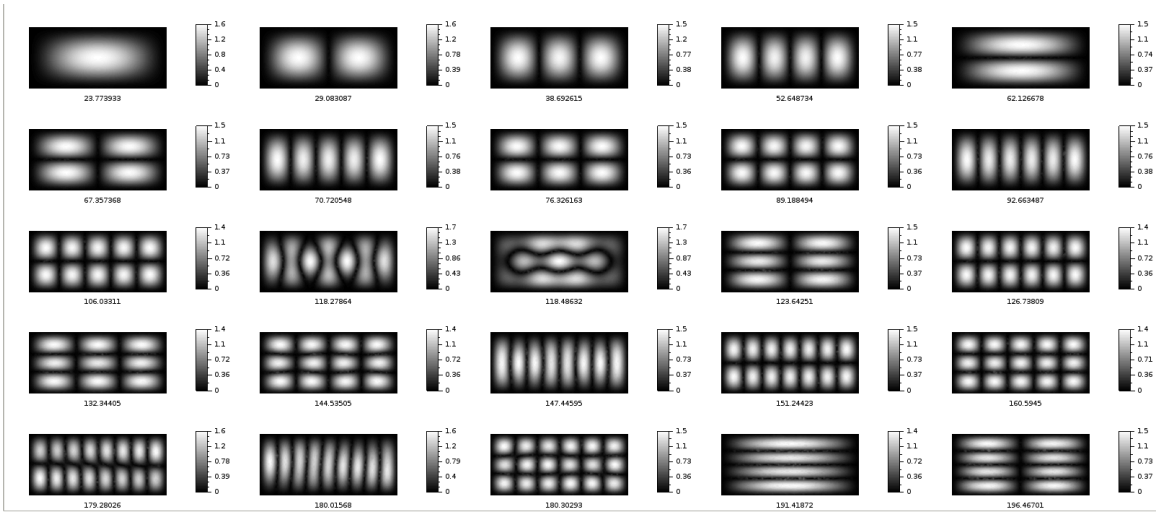


Figura A.11: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 2.250$*.*

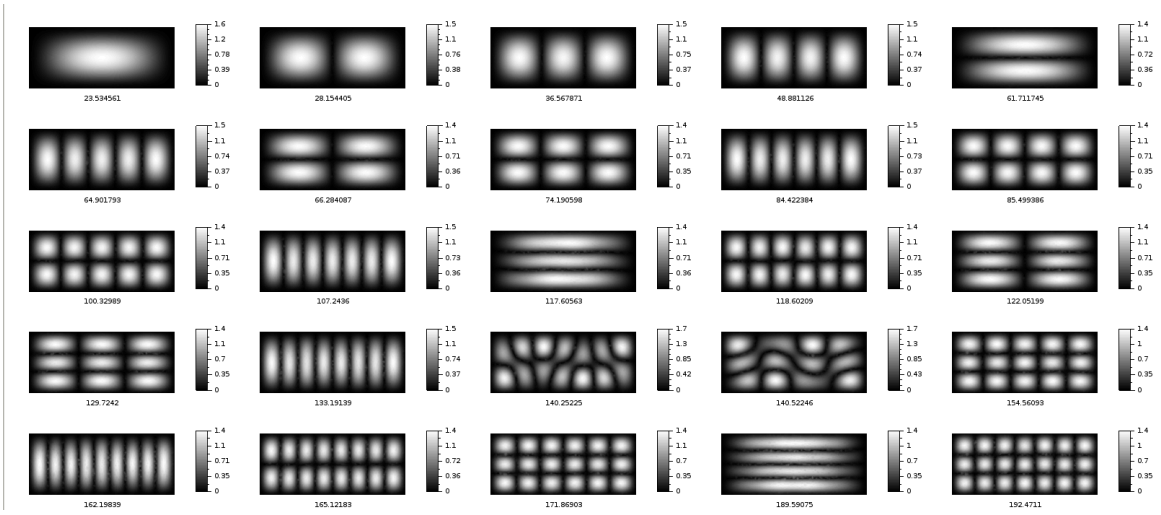


Figura A.12: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 2.375$*.*

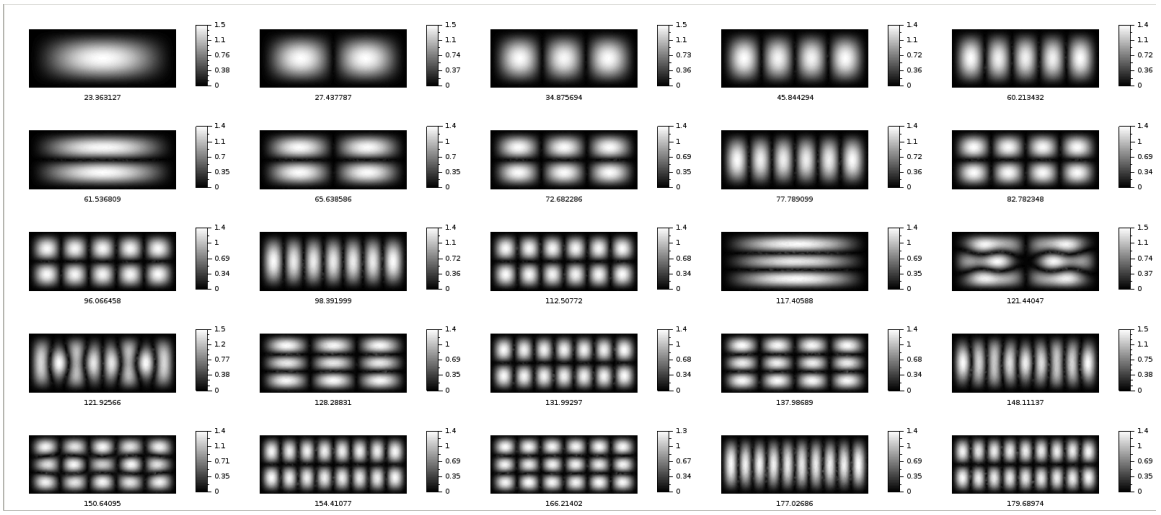


Figura A.13: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 2.500$*.*

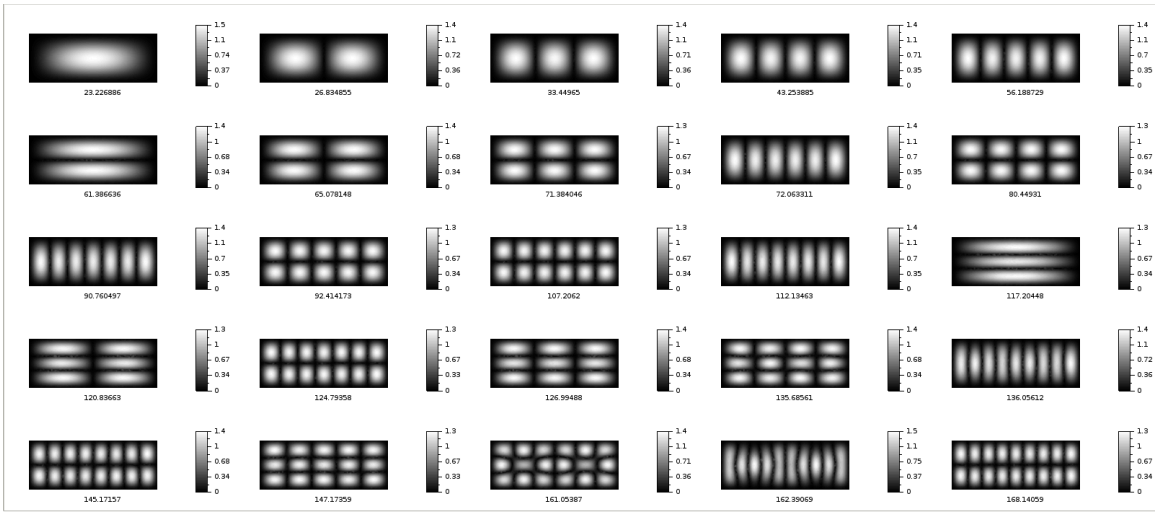


Figura A.14: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 2.625$*.*

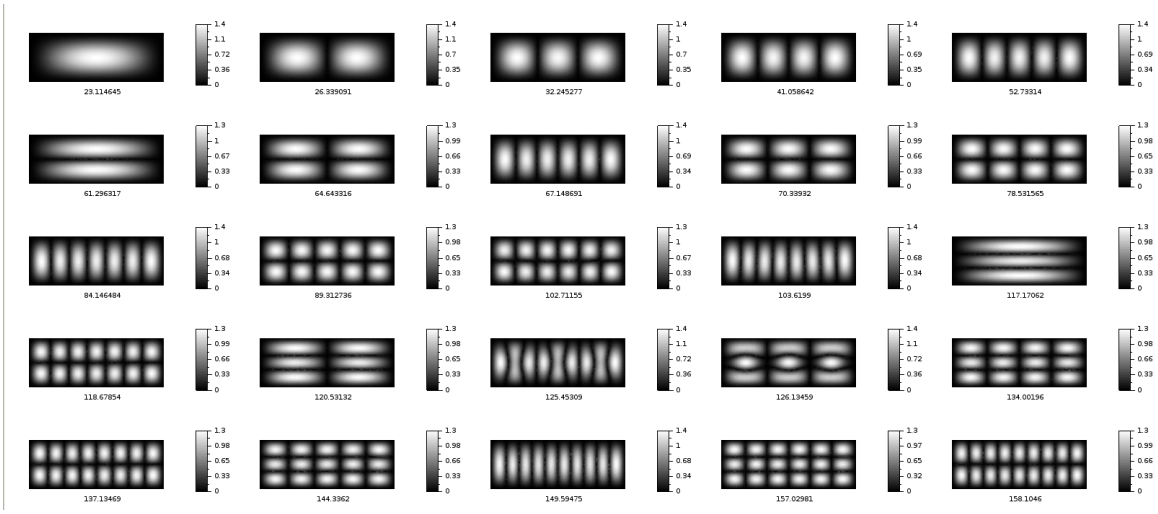


Figura A.15: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 2.750$*.*

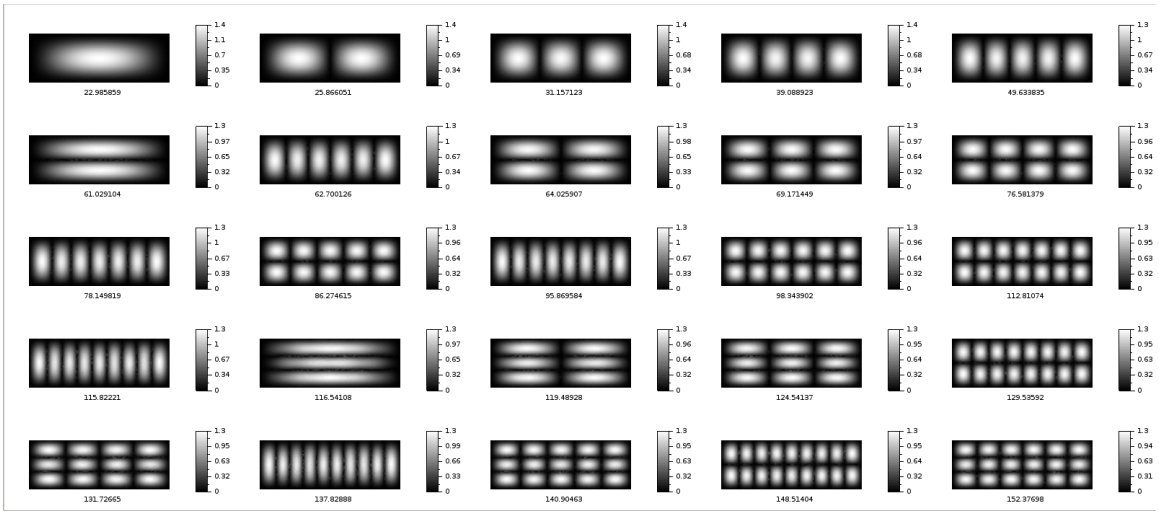


Figura A.16: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 2.875$*.*

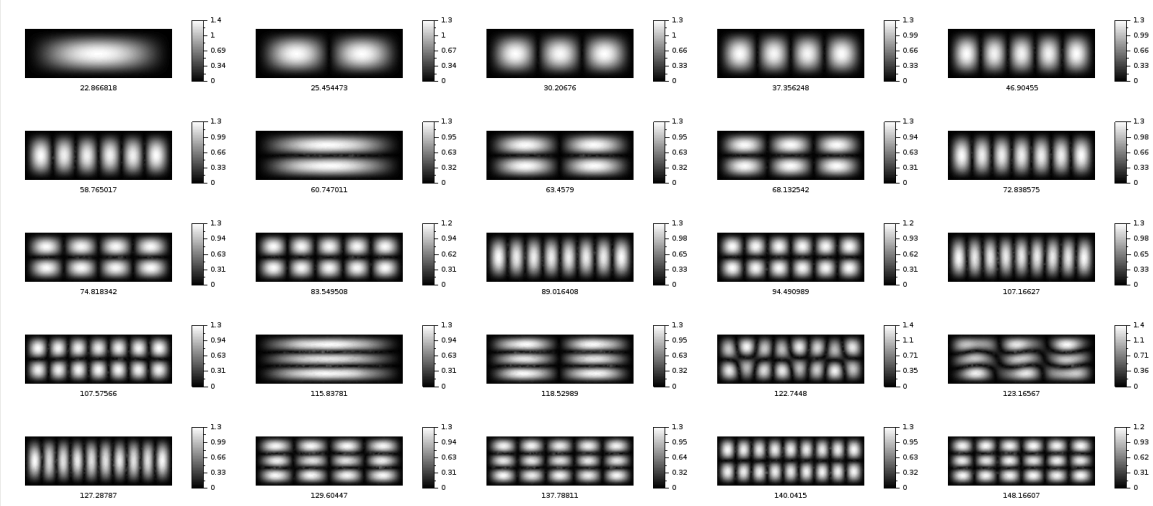


Figura A.17: *Placa retangular de dimensão* $c \times 1.0$*, com* $c = 3.0$*.*

Apêndice B

Códigos em Fortran 90

B.1 Formulação quadrática

```
PROGRAM Dani

!-------------------------------------------------------------------------------
-----------------------------------------!
! *.....Elementos Finitos : Elemento triangular não conforme
quadrático Versão 3 .....................**    !
!                                                                              !
!  Este programa calcula a discretização da integral variacional para o
problema de autovalores do                  !
!  bilaplaciano  e monta um problema de autovalores generalizados. Este
por sua vez é resolvido pela função         !
!  dsygv do Lapack. Ao final, ele prepara arquivos de leitura para a
plotagem gráfica pelo Scilab                !
!                                                                              !
!                                                                              !
! ALTERAÇOES NESSA VERSAO: Estou usando matrizes das integrais P, e as
matrizes D e E para montagem da matriz de massa   !
!                          Na versão anterior a matriz de massa era obtida
via  leitura de arquivo.                    !
!                          Código mais
enxuto                                      !
!                          Status:
Funcionando                                 !
!                                                                              !
!                                                                              !
!  Uses:  Subroutines dqtrf e dqtri do lapack para inverter
matrizes                                    !
!         SUBROUTINE dsygv( ITYPE, JOBZ, UPLO, N, A, LDA, B, LDB, W, WORK,
lwork, info) do Lapack                      !
!         para a solucão do problema de autovalores
generalizados                                                  !
!                                                                              !
!                                                                              !
!         Subrotina 'dqetri' do lapack para inversao de
matrizes                                                !
!-------------------------------------------------------------------------------
-----------------------------------------!
!
Enderecos                                                                      !
!                                                                              !
!'/home/danielapm/Documents/Malha/triangle_O2/Qua.5.ele'
lab                                                 !
!'/home/daniela/Documentos/Malha/triangle O2/Quar.1.ele'
casa                                                !
!'/home/posmap/daniela/Documentos/Malha/triangle_O2/Qua2.1.ele' rede
IME                                                 !
!-------------------------------------------------------------------------------
-----------------------------------------!

  Implicit none
  ! Variáveis da modelagem

  Integer :: nele,nnos,nospe,nedges      ! Numero de elementos
  triangulares da malha, numero de nós(vértices)
                                  !da malha, nº de nós por elemento
                                  triangular,numero de aresta,
                                  respectivamente
                                  !Serão lidos nos arquivos .ele , .node e
                                  .edge

```



```

  Integer, allocatable, dimension (:) :: fix, bordo   ! vetor que armazena
  a informação sobre o nós que ficarão fixos e sobre
                                  ! os nós do bordo, respectivamente.

  Integer :: r, dimensao              ! dimensão final das matrizes de
  massa e rigidez,
                                  !i.e. depois das condições de
                                  contornono serem
                                  add.

  Integer :: bordo_fixo=1, nn_fix    !bordo fixo recebe os valores 1 ou 0
  e indica se o bordo está ou não fixo.
                                  !nn_fix é a quantidade de nós fixos
                                  da malha

  Integer, allocatable, dimension(:,:) :: edge    !Matriz que armazenará
  os pontos iniciais e finais de cada aresta. terá dimensao 3*nedges

  Double precision, allocatable, dimension (:) :: x,y ! Vetores que
  armazenam as coordenadas (x,y) de cada nó: (x(i), y(i))

  Double precision, allocatable, dimension (:,:) :: normal ! Vetor vai
  representar a matriz que contem as coordenas dos vetores normais


  Double Precision, allocatable, dimension (:,:) :: S, M   ! Vetores que
  representam as Matrizes de Rigidez e de Massa.


  Integer:: ierr, dierr  ! controle da alocacão
  Integer:: i,j,k        ! Contadores
  Integer:: contabilizou cc
  Double precision :: tempo1,tempo2

  ! Variáveis que a funcão dsygv necessita

  Double precision, allocatable, dimension (:) :: w, work  !w é um vetor
  que recebera os n=r  autovalores
  Character :: jobz='V', uplo='U'
  Integer :: info, itype=1, lda, ldb, lwork

  ! Leitura de arquivos .edge, .node e .ele para alocação das váriaveis

  !Leitura do arquivo .ele gerado pelo Triangle. Malha com ordem 2
  Open(40,file='/home/daniela/Documents/Malha/triangle_O2/quadrado.5.ele',
  status='old')
  Read(40,*) nele, nospe
  print*, "Cada elemento triangular tem ",nospe,"nós"


  !Leitura do arquivo .node gerado pelo Triangle
  Open (41, file =
  '/home/daniela/Documents/Malha/triangle O2/quadrado.5.node', status='old')
  Read (41,*) nnos

  !Leitura do arquivo .edge gerado pelo Triangle - malha de ordem 2
  Open (42, file=
  '/home/daniela/Documents/Malha/triangle_O2/quadrado.5.edge', status='old')
  Read(42,*) nedges

  dimensao=nnos

  ! Alocação das variáveis alocáveis
  allocate (T(7,nele), stat = ierr)

```



```

  allocate (bordo(nnos),fix(nnos), stat = ierr)
  if (ierr/=0) then
    print*,"Erro de alocação do variavel bordo"
    stop
  end if

  allocate (x(nnos), y(nnos), stat = ierr)
  if (ierr/=0) then
    print*,"Erro de alocação das variaveis x e y"
    stop
  end if

  allocate (edge(3,nedges), normal (2,nedges), stat = ierr)
  if (ierr/=0) then
    print*,"Erro de alocação do variavel edge"
    stop
  end if

  print*,"nnos,nele,nedges",nnos,nele,nedges

  Call Le arquivos(nele, nnos, nospe, nedges, x, y, bordo, T, bordo fixo,
  edge)

  !Verifica quantos (nn fix) e quais nós da malha que estão fixos (vetor
  fix).
  Call Define_condicoes_contorno(nnos,bordo,bordo_fixo,dimensao, r, fix,
  nn fix)

  !A partir daqui, o valor de r é conhecido e posso alocar as matrizes S e M
  allocate (M(r,r), S(r,r), stat = ierr)
  if (ierr/=0) then
    print*,"Erro de alocação das variaveis S e M"
    stop
  end if

  contabilizou cc = 0

  Call Define vetores normais(nedges,nnos,edge,normal, x,y)

  ! Monta matrizes de rigidez e massa globais
  print*,"entrei em monta_matriz"
  Call Monta_matrizes (nele, nnos, nospe,nedges,edge,normal, x, y, fix, T,
  nn fix,r, dimensao, S, M, contabilizou cc)
  print*,"sai de monta matriz"

  !Registra em arquivo as matrizes de Rigidez e de Massa iniciais, pois
  elas serão modificadas pela função dsygv
  !Call Escreve_matrizes(r,M,S)

  print*,"Dimensão final das matrizes:",r

  lda=r
  ldb=r
  lwork=3*r-1 ! Tentar um valor melhor para lwork. Ver documentação da fc
  dsygv

  print*,"lwork",lwork

 !---- Alocação das variáveis para dsygv-----------------------!

  allocate (w(r), work(lwork), stat = ierr)
  if (ierr/=0) then
    print*,"Erro de alocação das variáveis w e work"
    stop

```



```
  print*,"entrei em DSYGV - resolvendo o autovalor generalizado"

  Call dsygv (itype, jobz, uplo, r, S, lda, M, ldb, w, work, lwork, info)
  if(info .eq. 0) then
    Print*, "info=",info , "Sucess na resolucão do problema de autovalor
    generalizado"  ! Controle da funcão dsygv: info=0 indica sucesso nos
    calculos
  end if

  ! Impressão dos autovalores e autovetores em arquivo
  Call Imprime autovetores(r,w,S)
  print*,"Sai de imprime autovetores"

  ! Prepara arquivos para o scilab
  print*,"entrei em dados plotagem"
  Call Dados plotagem(nedges,nnos,nele,x,y,S,w,T,r,fix)
  print*,"acabei!"


END PROGRAM Dani


!-------------------------------------Subrotinas-------------------------------
-----------------------------!

!--------------------------------LEITURA DE
ARQUIVOS-------------------------------------------------!

Subroutine Le arquivos (nele, nnos, nospe, nedges, x, y, bordo, T,
bordo_fixo, edge)

  Implicit none
  Integer, intent(in) :: nele, nnos, nospe, bordo_fixo, nedges
  Integer, intent(out) :: T(7,nele)                ! T é matriz
  (nospe+1)x(nele) que na coluna j guarda o numero
                                  !do triangulo i, e seus vertices
                                  (i 1, i 2, i 3, i 4, i 5, i 6),
                                  !nas linhas seguintes, segundo a
                                  numeracão global da malha

  Integer, intent(out) :: bordo(nnos)      ! Esses inteiros recebem
  valores 0 ou 1 e marcam os vertices que estao no bordo

  Double precision, intent(out) :: x(nnos), y(nnos)
  !double precision, dimension (36) :: S2        ! matriz simetrica cte
  da integral 2
  Integer, intent(out) :: edge(3,nedges)          ! matriz que guarda os
  pontos final e inicial das arestas da malha

  ! Variáveis locais
  Integer:: i,j,k
  Double precision :: c,b

  ! Leitura do arquivo .ele gerado pelo Triangle. Malha com ordem 2
  Do j=1,nele
    Read (40,*) T(1,j),T(2,j),T(3,j),T(4,j),T(5,j),T(6,j),T(7,j)
  End do
  Close(40)

  !Leitura do arquivo .node gerado pelo Triangle

  Do j=1, nnos
    Read (41,*) c, x(j), y(j), b, bordo(j)  ! a malha quadrado e
    tri escaleno têm 5 parametros pra serem lidos.
  End Do

```

```
209
210       Do i=1, nedges
211         Read(42,*) edge(1,i),edge(2,i),edge(3,i)
212       end do
213       Close(42)
214
215
216     End subroutine Le_arquivos
217
218     !-------------------------------DEFINIÇÃO DOS VETORES NORMAIS-----------------------------------------------!
219
220     Subroutine  Define vetores normais(nedges,nnos,edge,normal, x,y)
221
222       Implicit none
223       Integer, intent(in) :: nedges, nnos
224       Integer, intent(in) :: edge(3,nedges)
225       Double precision, intent(out) :: normal(2,nedges)  ! matriz de vetores normais. Cada coluna i tem as corrdenadas do i-esimo vetor normal: (normal(1,i),normal(2,i))
226       Double precision,intent(in) :: x(nnos),y(nnos)
227
228     !Variáveis locais
229
230       Integer :: i,j,k
231       Double precision :: normal unitaria(2,nedges), norma(nedges)
232
233       Do i=1,nedges
234         !A aresta i tem pto inicial edge(2,i) e pto final edge(3.i), segundo a informação que o arquivo .edge fornece
235         !As coordenadas desses pontos são [x(edge(2,i)); y(edge(2,i))] e [x(edge(2,i)) ; y(edge(2,i))]
236         !As coordenadas dos vetores normais sao definidas pela rotacão da aresta i, por um angulo de 90 graus no sentido anti-horário
237         !print*,i
238         !print*,edge(2,i),edge(3,i)
239         if(edge(2,i) .LT. edge(3,i)) then
240           normal(1,i)=-(y(edge(3,i))-y(edge(2,i)))
241           normal(2,i)=x(edge(3,i))-x(edge(2,i))
242         end if
243
244         if (edge(2,i) .GT. edge(3,i)) then
245           normal(1,i)=-(y(edge(2,i))-y(edge(3,i)))
246           normal(2,i)=x(edge(2,i))-x(edge(3,i))
247         end if
248       End do
249
250     ! tornando a normal unitária
251       Do i=1,nedges
252         norma(i) = sqrt(normal(1,i)**2 + normal(2,i)**2)
253
254         normal unitaria(1,i)=normal(1,i)/norma(i)
255         normal_unitaria(2,i)=normal(2,i)/norma(i)
256
257         normal(1,i) = normal unitaria(1,i)
258         normal(2,i) = normal_unitaria(2,i)
259
260       End do
261
262     End Subroutine Define vetores normais
263
264
265     !-----------------------------DEFINIÇÃO DAS CONDICÕES DE CONTORNO-----------------------------------------!
266
```



```
    enquanto, define quais vertices da malha permanecerao fixos.
268
269     Subroutine Define condicoes contorno(nnos,bordo,bordo fixo,dimensao, r, fix ,nn_fix)
270
271       Integer, intent(in) :: nnos, bordo(nnos), bordo fixo, dimensao
272       Integer, intent(out) :: fix(nnos),r,nn fix
273
274       !Variávies locais
275       Integer :: i,j
276
277
278       If (bordo fixo.eq.1) then
279         print*,"BORDO FIXO!"
280         Do i=1,nnos
281           fix(i)=bordo(i) ! Condicão de bordo fixo.
282         End do
283       End if
284
285       nn fix=0
286
287       !Laco no numero de nós da malha. Verificação da quantidade dos que estão fixos.
288       Do i=1,nnos
289         if ((fix(i).eq. 1) ) then  ! se o vertice i esta fixo,
290           nn fix=nn fix+1
291         end if
292       End do
293
294       r=dimensao-nn_fix   ! dimensao das matrizes S e M globais, após contabilizar as condições de contorno.
295       print*,"nn fix,dimensao, r",nn fix,dimensao,r
296
297     End Subroutine Define condicoes contorno
298
299     !-------------------------------------------------------------------------------------------------------!
300
301     !------------------------MONTAGEM DAS MATRIZES DE MASSA E RIGIDEZ GLOBAIS-----------------------------------!
302
303     Subroutine Monta matrizes(nele, nnos, nospe,nedges,edge,normal, x, y, fix, T , nn_fix,r, dimensao, S, M, contabilizou_cc)
304
305       Implicit none
306
307       Integer, intent(in) :: nele, nnos, nospe,r, nedges, nn fix, dimensao
308       Integer, intent(in) ::  T(7,nele), fix(nnos), edge(3,nedges)
309       Double precision, intent(in) ::  x(nnos), y(nnos), normal(2,nedges)
310
311       Double precision,intent(out) :: S(r,r), M(r,r)
312       Integer, intent(out) :: contabilizou cc
313
314       ! Variaveis locais
315
316       Double Precision :: a(3),b(3),Ji ! a e b são vetores que contem as coordenadas do vetor normal n=(a,b)
317       Double Precision, Dimension(6,6) :: Si, Mi, C, D, P, Aux, E, F
318       Double precision, allocatable, dimension (:,:) :: S_, M_ ! matrizes auxiliares para a montagem de M e S
319       Integer, allocatable, dimension (:) :: vi    ! Vetor auxiliar que guarda os vertices do iésimo triangulo
320                                                    ! segundo a numeracão global da malha
321       Integer i,j,k,l,ierr, n_i(3)
```



```
324       Integer :: pot(6,2)
325
326
327       allocate(S_(1:dimensao,1:dimensao), M_(1:dimensao,1:dimensao), vi(1:nospe), stat=ierr)
328       if(ierr/=0) then
329         print*, "Erro na alocação das variáveis M_,S_, e vi"
330         stop
331       end if
332
333       !Montagem da matriz pot (potencias de x e y)
334       k=1
335       Do i=0,2   ! 2 é  grau da aproximação
336         Do j=0,i
337           pot(k,1)=i-j
338           pot(k,2)=j
339           k=k+1
340         End do
341       End do
342
343       ! Inicializacão de S  e M  com zero
344       Do i=1,nnos
345         S_(j,:)= 0.0D0
346         M (j,:)= 0.0D0
347       End do
348
349       Call Monta_matriz_P(P,pot,2,6)
350
351
352
353       Do i=1,nele   !------------------------------- Laco no número de triângulos ----------------------!
354
355         Do j=1,nospe
356           vi(j)=T(j+1,i) ! Os vertices do triangulo i são: T(2,i), T(3,i), T(4,i), T(5,i), T(6,i) e T(7,i)
357         End Do
358
359         Ji=((x(vi(2)))-x(vi(1)))*(y(vi(3))-y(vi(1)))- (x(vi(3))-x(vi(1)))*(y(vi(2))-y(vi(1)))
360
361         !-------------------Definição da matriz de vetores normais locais n -------------
362         !Vamos estabelecer uma numeração local para os vetores normais globais cada elemento triangular i tem 3 vetores normais: n i(j), j=1,2,3
363
364         j=1
365         Do while (j.LE.nedges)
366           if( ((vi(1) .eq. edge(2,j)) .and. (vi(2) .eq. edge(3,j))) .or. ((vi(2) .eq. edge(2,i)) .and. (vi(1) .eq. edge(3,i)))) then
367             n i(3)=j !n i(3) é vetor oposto ao vertice 3 e é o j-ésimo vetor normal da tabela normal
368             j=nedges+1
369           else
370             j=j+1
371           end if
372         End do
373
374         j=1
375         Do while (j.LE.nedges)
376           if( ((vi(2) .eq. edge(2,i)) .and. (vi(3) .eq. edge(3,i))) .or. ((vi(3) .eq. edge(2,j)) .and. (vi(2) .eq. edge(3,j)))) then
377             n i(1)=j !n i(1) é é vetor oposto ao vertice 1 e é  o j-ésimo vetor normal da tabela normal
378             j=nedges+1
```



```
381           end if
382         End do
383
384         j=1
385         Do while (j.LE.nedges)
386           if( ((vi(1) .eq. edge(2,j)) .and. (vi(3) .eq. edge(3,i))) .or. ((vi(3) .eq. edge(2,j)) .and. (vi(1) .eq. edge(3,j)))) then
387             n i(2)=j !n i(2) é é vetor oposto ao vertice 2 e é o j-ésimo vetor normal da tabela normal
388             j=nedges+1
389           else
390             j=j+1
391           end if
392         End do
393
394         Do j=1,3
395           a(j)=normal(1,n_i(j))
396           b(j)=normal(2,n i(j))
397         End do
398
399       ! ------- Montagem da matriz C, que relaciona as variaveis nodais com os parametros da aproximação. u=C*alfa --- !
400         Call Monta_matriz_C (C,x,y,vi,nnos,nospe,a,b,i)
401         Call Monta matriz D (D,x,y,vi,nospe,nnos,2,6,i)
402         Call Monta matriz E (E,x,y,vi,nospe,nnos,2,6,pot)
403         Call Monta matriz Si (Ji,C,Si,sigma, x,y,vi,nnos,nospe)
404         Call Monta_matriz_Mi (Mi,D,E,C,P,Ji)
405
406
407
408       !-------------- Monta matrizes de rigidez e massa globais S  e M ------------------------!
409
410       ! Para cada triangulo i, as matrizes recebem nas posicões (vi(k), vi(l)) ! a componente (k,l) das matrizes Si e Mi
411
412         Do l=1,nospe
413           Do k=1, nospe
414             !print*,"posicões nao nulas da matriz de massa:",vi(l),vi(k)
415             S (vi(k),vi(l))= S (vi(k),vi(l)) + Si(k,l)
416             M (vi(k),vi(l))= M (vi(k),vi(l)) + Mi(k,l)
417           End Do
418         End Do
419
420       End Do  !---------------------- FIM DO LAÇO NO NUMERO DE TRIÂNGULOS-----------------------------!
421
422       Call Contabiliza condicoes de contorno (S ,M ,fix,nnos,contabilizou cc,r)
423       print*, "Contou com condições de contorno?",contabilizou_cc
424
425       Call Redimensiona matrizes(dimensao,r,M,S,M ,S )
426
427
428       CONTAINS
429         Double Precision Function Det (A,n)
430
431         implicit none
432         integer:: n ! n é a dimensao da matriz A
433         double precision, dimension (n,n) :: A
434
435         ! Variáveis locais
436
437         integer :: lda
438         integer, dimension( n ) :: ipiv
439         integer :: info, i
440
```

```
441       double precision:: determinante
442
443       determinante=1.0D0
444       lda=n
445
446       ! Decomposição LU da matriz A
447       Call dgetrf (n,n,A,lda,ipiv,info)
448
449       if (info /= 0) then
450         stop 'Matriz é numericamente singular! Decomposição LU não é unica'
451       end if
452
453      ! Calculo do determinante de A:
454       do  i=1,n
455        determinante=determinante*A(i,i)
456       end do
457       Det=determinante
458
459      End function Det
460
461    End Subroutine Monta matrizes
462
463    Subroutine Monta matriz Mi(Mi,D,E,C,P,Ji)
464
465     Implicit none
466
467     Double precision, intent(in) :: Ji, D(6,6), E(6,6), C(6,6), P(6,6)
468     Double precision, intent(out) :: Mi(6,6)
469
470     ! Variáveis locais
471     Integer :: i,k,l
472     Double precision :: F(6,6) , Aux in(6,6), Aux out(6,6)
473
474     Aux in(1:6,1:6)=P(1:6,1:6)
475
476     F=matmul(D,E)
477
478     Call Calcula matriz semelhante(F,6,6,Aux in,6,F,6,6,Aux out)
        !Aux out=F^T*Aux in*F
479
480     Aux in(1:6,1:6)=Aux out(1:6,1:6)
481
482     Call Calcula_matriz_semelhante(C,6,6,Aux_in,6,C,6,6,Aux_out)
        !Aux out=C^T*Aux in*C
483
484     Mi=Ji*0.5D0*(Aux_out+transpose(Aux_out))
485
486     ! Teste de simetria de Mi
487     Do l=1,6
488       Do k=1,6
489         if (Mi(k,l).ne.Mi(k,l)) Print*,"Matriz Mi não é simetrica. Erro no
            elemento Mi", k,l
490       End do
491     End do
492
493    End Subroutine Monta_matriz_Mi
494
495    Subroutine Monta matriz Si (Ji,C,Si,sigma, x,y,vi,nnos,nospe)
496
497     Implicit none
498
499     Double precision, intent(out) :: Si(6,6)
500     Double precision, intent(in) :: Ji, sigma
501     Double precision,intent(in) :: x(nnos), y(nnos), C(6,6)
502     Integer,intent(in) :: nnos, nospe, vi(nospe)
```



```
505     Integer :: i,j,k,l
506     Double Precision, dimension (1,6) :: Dxx,Dyy,Dxy ! ! Matrizes do operador
        derivacão
507     Double Precision, dimension (6,6) :: A,Aux_out,B,G,I_1,I_2,I_3,I_4!
        matrizes auxiliares
508     Double precision, dimension (1,1) :: P
509     Double Precision, dimension (1,1) :: Aux in
510
511     P(1,1)=0.5D0
512
513     !Definição de Dxx, Dyy e Dxy, as matrizes de derivação
514     Dxx(1:1,1:6)=0.0D0
515     Dyy(1:1,1:6)=0.0D0
516     Dxy(1:1,1:6)=0.0D0
517
518     Dxx(1,4)=2.0D0
519     Dxy(1,5)=1.0D0
520     Dyy(1,6)=2.0D0
521
522     Aux in=P    ! P é de dim 1
523
524     ! Matriz da forma quadratica da Integral do quadrado de wxx
525     Call Calcula matriz semelhante(Dxx,1,6,Aux in,1,Dxx,1,6,I 1)
        !I_1=Dxx^T*Aux_in*Dxx
526
527     ! Matriz da forma quadratica da Integral do quadrado de wyy
528     Call Calcula_matriz_semelhante(Dyy,1,6,Aux_in,1,Dyy,1,6,I_2)
        !I 2=Dxx^T*Aux in*Dxx
529
530     !Matriz da forma quadratica da Integral do quadrado de wxy
531     Call Calcula matriz semelhante(Dxy,1,6,Aux in,1,Dxy,1,6,I 3)
        !I 3=Dxx^T*Aux in*Dxx
532
533     !Call Calcula matriz semelhante(C,6,6,A,6,C,6,6,Aux out)  !Aux out=C^T*A*C
534     !I 3=0.5D0*(transpose(Aux out)+Aux out)
535     !I 3=Aux out
536
537     ! Matriz da forma bilinear da Integral do quadrado de wyy*wxx
538     !Para ser simetrica, ela deve ser montada considerndo a soma
        (Dxx^T)*P*Dyy + (Dyy^T)*P*Dxx
539
540     Call Calcula matriz semelhante(Dxx,1,6,Aux in,1,Dyy,1,6,A)
        !A=Dxx^T*Aux_in*Dyy
541     Call Calcula matriz semelhante(Dyy,1,6,Aux in,1,Dxx,1,6,B)
        !B=Dyy^T*Aux_in*Dxx
542     I_4=0.5D0*(A+B)
543
544     !Call Calcula matriz semelhante(C,6,6,G,6,C,6,6,Aux out)  !Aux out=C^T*G*C
545     !I 4=0.5D0*(transpose(G)+G)
546     !I_4=G
547
548     G=(I 1+I 2+2.0D0*(1.D0-sigma)*I 3+2.D0*sigma*I 4)
549
550     Call Calcula matriz semelhante(C,6,6,G,6,C,6,6,Aux out)  !Aux out=C^T*Aux*C
551
552     Si=Ji*0.5D0*(transpose(Aux_out)+Aux_out) ! simetrização. sem ela S
        global ficou assimetrica..
553     !Si=Ji*Aux out
554
555     ! Teste de simetria de Si
556     Do l=1,6
557       Do k=1,6
558         if (Si(k,l).ne.Si(k,l)) Print*,"Matriz Si não é simetrica. Erro no
            elemento Si", k,l
559       End do
```



```
566     Implicit none
567     Double precision, intent(out) :: P(n,n)
568     Integer, intent(in) :: pot (6,2)
569     integer, intent(in) :: grau_aprox
570
571     ! Variaveis locais
572     Integer :: n,k,i,j
573     Double precision:: numerador,denominador !
574     !var auxiliares, pois Fat é fc inteira. o quociente tava dando um num
        inteiro.
575
576     !n é a dimensao de P, e depende do grau do polinomio que sera integrado.
577     !n=21 se grau_aprox =5
578     !n=10 se grau_aprox =3
579     !Essas sao as duas possibilidades nesse programa
580
581     j=(grau_aprox+2)*(grau_aprox+1)/2  ! soma de n+1 elementos da p.a. de
        razao 1,cujo 1º elemento é 1.
582     Do k=1,i
583       Do i=1,j
584         numerador=(Fat((pot(k,1)+pot(i,1)))*(Fat(pot(k,2)+pot(i,2))))
585         denominador=(Fat(pot(k,1)+pot(i,1)+pot(k,2)+pot(i,2)+2))
586         P(k,i)=numerador/denominador  ! Divisão entre dois numeros reais
587       End do
588     End do
589
590
591    CONTAINS
592     Integer Function Fat(n)           !Função que calcula o fatorial de um
        numero inteiro >= 0
593
594      Implicit none
595      Integer, intent (in) :: n
596      Integer :: produto, i
597
598      produto=1
599      Do i=1,n
600        produto=produto*i
601      End do
602      Fat=produto
603
604     End Function Fat
605
606
607    End Subroutine Monta matriz P
608
609
610    Subroutine Monta matriz D (D,x,y,vi,nospe,nnos,grau aprox,n,i)
611
612     !Montagem da matriz D, que relaciona os parametros a1,...a6 com os
        parametros alfa's
613     ! Open (101,
        file='/home/posmap/daniela/Documentos/fortran/Teste naoconforme D.txt',
        status='old')
614
615     Implicit none
616     Double precision, intent(out) :: D(n,n)
617     Double precision, intent(in) :: x(nnos),y(nnos)
618     Integer, intent(in) :: i,vi(nospe)
619     Integer, intent(in) :: nospe,nnos, n, grau aprox
620
621     ! Variaveis locais
622     Integer :: j,p,l,m,inicio bloco
623     Double precision :: soma,formula,produto,determinante
624     Double precision :: x_(1:3),y_(1:3)
```



```
628     x_(3)=x(vi(3))-x(vi(1))
629     x_(2)=x(vi(2))-x(vi(1))
630     y_(3)=y(vi(3))-y(vi(1))
631     y_(2)=y(vi(2))-y(vi(1))
632
633     Do j=0,grau aprox
634       inicio_bloco=1+((j*(j+1))/2)
635       Do p=0,j
636         Do l=0,j
637           soma=0.0D0
638           Do m=0,l
639             !Queremos calcular
                (Binomial(j-p,l-m))*(Binomial(p,m))*(x(vi(3))**(l-m))*(x(vi(2))**(
                j+m-l-p))*(y(vi(3))**(m))*(y(vi(2))**(p-m))
640             !mas vamos separar esses fatores para facilitar
641             produto=Binomial(j-p,l-m)*Binomial(p,m)
642
643             if (produto .eq. 0.0D0) then    !se produto é zero, a formula é
                zero. Isso também implica a existencia de expoentes
                negativos,para os quais
644                                             !não podemos calcular as
                                                pontencias dos xi's
645               formula=0.0D0
646             else ! i.e. se os fatores do produto forem ambos nao nulos, não
                teremos expoentes negativos
647               formula=produto*((x (3)**(l-m))*(x (2)**(j+m-l-p))*(y (3)**(m))*(
                  y (2)**(p-m)) )
648             end if
649
650             if (isnan(formula)) then   ! Verificar a ocorrencia de NaN
651               print*,"expoentes negativos para ",j-p-l+m,"e",p-m
652             end if
653
654             soma=soma+formula
655
656           End do
657
658           D(inicio bloco+l,inicio bloco+p)=soma
659         End do
660       End do
661     End do
662
663
664    CONTAINS
665
666    Integer Function Fat(n)           !Função que calcula o fatorial de um
       numero inteiro >= 0
667
668     Implicit none
669     Integer, intent (in) :: n
670     Integer :: produto, i
671
672     produto=1
673     Do i=1,n
674       produto=produto*i
675     End do
676     Fat=produto
677
678    End Function Fat
679
680
681    Integer Function Binomial(n,p)
682    ! Função que retorna o coeficiente binomial de dois inteiros quaisquer.
683    !Combinação de n, de p a p. Se p>n, binomial = 0. n e p devem ser nao
       negativos!!
```

```
687       integer :: bin
688
689       if (p.GT.n) then ! se p>n, então o binomial é zero
690         bin= 0
691       else
692         bin=Fat(n)/(Fat(p) * Fat(n-p))
693       end if
694         Binomial=bin
695
696     End function Binomial
697
698     Double precision Function Det (A,n)
699
700       Implicit none
701       Integer:: n ! n é a dimensao da matriz A
702       Double precision, dimension (n,n) :: A
703
704       ! Variáveis locais
705
706       Integer :: lda
707       Integer, dimension( n ) :: ipiv
708       Integer :: info, i
709       Double precision :: determinante
710
711       determinante=1.0D0
712       lda=n
713
714       ! Decomposição LU da matriz A
715       Call dgetrf (n,n,A,lda,ipiv,info)
716
717       if (info /= 0) then
718         stop 'Matriz é numericamente singular! Decomposição LU não e unica'
719       end if
720
721       ! Calculo do determinante de A:
722       Do  i=1,n
723         determinante=determinante*A(i,i)
724       End do
725       Det=determinante
726
727     End function Det
728
729   End Subroutine Monta_matriz_D
730
731
732     Subroutine Monta_matriz_E (E,x,y,vi,nospe,nnos,grau_aprox,n,pot)
733
734       Implicit none
735       Double precision, intent(out)  :: E(n,n)
736       Double precision, intent(in) :: x(nnos),y(nnos)
737       Integer, intent(in) :: vi(nospe), pot(6,2)
738       Integer, intent(in) :: nnos, nospe, n, grau_aprox
739
740       ! Variaveis locais
741       Integer :: i,k
742
743       E=0.0D0
744       !Diagonal de E vale 1.0
745       Do j=1,n
746         E(j,j)=1.0D0
747       End do
748
749     ! Linha 1 da matriz tem o polinomio de grau 2 nas variaveis x1,y1
750       Do k=1,6
751         E(1,k)=(x(vi(1))**pot(k,1))*(y(vi(1))**pot(k,2))
```



```
752     End do
753
754     !Demais linhas, acima da diagonal principal:
755
756     !"Bloco  grau 1"
757     E(2,4)=2.0D0*x(vi(1))
758     E(2,5)=y(vi(1))
759
760     E(3,5)=x(vi(1))
761     E(3,6)=2.0D0*y(vi(1))
762
763     CONTAINS
764
765     Double precision Function Det (A,n)
766
767      Implicit none
768      Integer:: n ! n é a dimensao da matriz A
769      Double precision, dimension (n,n) :: A
770
771      ! Variáveis locais
772
773      Integer :: lda
774      integer, dimension( n ) :: ipiv
775      integer :: info, i
776      Double precision :: determinante
777
778      determinante=1.0D0
779      lda=n
780
781      ! Decomposição LU da matriz A
782      Call dgetrf (n,n,A,lda,ipiv,info)
783
784      if (info /= 0) then
785        stop 'Matriz é numericamente singular! Decomposição LU não e unica'
786      end if
787
788     ! Calculo do determinante de A:
789      do  i=1,n
790       determinante=determinante*A(i,i)
791      end do
792      Det=determinante
793
794     End function Det
795
796   End Subroutine Monta matriz E
797
798   Subroutine Monta matriz C (C,x,y,vi,nnos,nospe,a,b,k)
799
800     Implicit none
801
802     Integer, intent(in) :: nnos, nospe, k, vi(nospe)
803     Double precision, intent(in) :: a(3),b(3), x(nnos),y(nnos)
804     Double precision, intent(out) :: C(6,6)
805
806     !Variáveis locais
807     Integer :: i,j
808     Double precision :: C (6,6)
809
810     C_(1:6,1:6)=0.0D0
811
812     Do i=1,3
813       C (i,1)=1.0D0
814       C (i+3,1)=0.0D0
815       C (i,2)=x(vi(i))
816       C_(j+3,2)=a(j)
```



```
817       C_(j,3)=y(vi(j))
818       C (i+3,3)=b(i)
819       C (i,4)=x(vi(i))*x(vi(i))
820       C_(j+3,4)=2*a(j)*x(vi(j+3))
821       C (i,5)=x(vi(i))*y(vi(i))
822       C (i+3,5)= a(j)*y(vi(i+3))+b(i)*x(vi(i+3))
823       C_(j,6)=y(vi(j))*y(vi(j))
824       C (i+3,6)=2*b(i)*y(vi(i+3))
825     End do
826
827     ! inversao da matriz C
828     Call Inverte matriz (C ,C,6,i)
829
830   End Subroutine Monta_matriz_C
831
832       !----------CONTABILIZAR NA MONTAGEM DAS MATRIZES AS CONDICÕES DE CONTORNO-------------------------------------!
833
834   Subroutine Contabiliza condicoes de contorno (S ,M ,fix,nnos, contabilizou_cc,r)
835
836     Implicit none
837
838     Integer, intent(in) :: nnos, fix(nnos),r
839     Integer, intent(out) :: contabilizou cc
840     Double precision, intent(inout) :: M (nnos,nnos), S (nnos,nnos)
841
842     !Variáveis locais
843     Integer:: i,i
844
845     ! Utilizando a info dos vértices que estão fixos na montagem das matrizes: As linhas e colunas relativas aos vértices que estão
846     !fixos são deletadas, uma vez que o valor da função nesses pontos ja é conhecido
847     print*,"Contabilizando as condicoes de contorno"
848
849     ! Laco no numero de nós. Verificação dos vértices que estão fixos
850     Do i=nnos,1,-1
851       if ((fix(i).eq. 1) .and. (i.ne.nnos)) then  ! se o vertice i esta fixo e se este não corresponde a
852                                                   ! ultima linha, então deslocamos as linhas e as colunas da matriz para cima e para a esquerda
853         Do j=i,nnos-1
854           S_(j,:)=S_(j+1,:)
855           S (:,j)=S (:,j+1)
856           M (j,:)=M (j+1,:)
857           M_(:,j)=M_(:,j+1)
858         End do
859
860       end if
861     End do
862     contabilizou cc=1
863
864   End Subroutine Contabiliza_condicoes_de_contorno
865
866     !-----Redimensiona  matrizes---
867   Subroutine Redimensiona_matrizes(dimensao,r,M,S,M_,S_)
868
869     Implicit none
870
871     Integer, intent(in) :: r, dimensao
872     Double precision, intent (in) :: M (dimensao,dimensao),S (dimensao,dimensao)
873     Double precision, intent(out) :: M(r,r),S(r,r)
```



```
877
878     Do j=1,r
879       Do i=1,r
880         S(i,j)=S_(i,j)
881         M(i,j)=M (i,j)
882       End do
883     End do
884
885   ! Teste de simetria das matrizes de massa e rigidez
886     Do i=1,r
887       Do j=1,r
888         if (S(i,j).ne.S(j,i)) Print*,"Matriz S não é simetrica. Erro no elemento S", i,j
889         if (M(i,j).ne.M(j,i))  Print*,"Matriz M não é simetrica.",M(i,j),"e diferente de",M(j,i)
890       End do
891     End do
892
893   End subroutine Redimensiona matrizes
894
895     !.....
896   Subroutine Inverte matriz (A,invA,n,k)
897   !Essa subrotina inverta a matriz A e devolve sua inversa na matria invA
898
899     Implicit none
900     Integer, intent(in):: n,k  !k indica o k-esimo triangulo. NO caso de problemas, quero saber em qual triangulo ele aconteceu.
901     Double precision,intent(in) :: A (n,n)
902     Double precision,intent(out) :: invA (n,n)
903
904   ! Variáveis locais
905
906     Integer :: m, lda, lwork
907     Double precision, dimension (n) :: work
908     Double precision, dimension( n, n ) :: B, C1,C2, I , D1, D2  ! matriz que será fatorizada A= PLU
909     Double precision :: normal,norma2, soma1 (6), soma2(6)        ! a norma de matriz que sera utilizada é a norma do sup
910     Integer, dimension( n ) ::ipiv
911     Integer :: info1,info2,i,l ! inseri nova variavel info2, para que elas nao se confundam nas chamadas dessas duas funções.
912
913     m=n
914     lda=n
915     lwork =n
916     B=A
917
918     ! Matriz Identidade
919     I=0.D0
920
921     Do j=1,6
922       I(j,j)=1.D0
923     End do
924
925
926     Call dgetrf(m,n,B,lda,ipiv,info1) ! Computa a fatorização LU da matriz B=A
927
928     if (info1 .ne. 0) then
929       print*,"Função inverte matriz. Fatorização LU comprometida no triangulo",k,"da malha."
930       if (info1 > 0)then
931         print*, "info1=",info1
932         print*,"O valor de U na diagonal",info1," é zero. E ese valor é ", B( info1,info1)
933         Do j=1,6
```

```
      end if
      if(info1 < 0) then
        print*,"info1=", info1, "O valor (-info)-esimo é ilegal"
      end if
      stop
    else
    !print*,"info1=", info1, "Sucesso na fatorização LU"
      Call dgetri (n, B, lda, ipiv, work, lwork,info2)
      if (info2 .ne. 0) then
        print*,"info2=",info2
        print*,"Erro ao inverter a matriz C relativa ao triangulo",k
      else
      ! print*,"Inversão realizada com sucesso no triangulo",k
      end if
    end if

    invA=B
    C1=matmul(invA,A) ! A^-1*A tem que dar a Id
    C2=matmul(A,invA) ! A*A^-1 tem que dar a Id
    D1=C1-I
    D2=C2-I

    ! Verificando a distancia se A*A^-1 = Id : Faremos usando usando a norma do sup ||A|| =  maximo da soma dos modulos das linhas da matriz

    Do l=1,3
      soma1(l)=0.D0
      soma2(l)=0.D0
      Do i=1,3
        soma1(l) = soma1(l) + abs(D1(l,j))
        soma2(l) = soma2(l) + abs(D2(l,j))
      End do
    End do
    norma1=max(soma1(1), soma1 (2), soma1 (3))
    norma2=max(soma2(1), soma2 (2), soma2 (3))

     ! Se uma dessas normas for muito grande, sinalizar para o usuário
    if (norma1 >= 10.D-10) then    ! a precisão aqui é de 10^16. Vou considerar razoável erros até a decima casa decimal
      print*,"Testando a inversão:"
      print*,"ATENÇÃO! INVERSAO DE MATRIZ COM problemas NO TRIANGULO",k
    end if

    !print*,"norma de A^-1*A - Id :", k, norma1
    !print*,"norma de A*A^-1 -Id ",k, norma2

  End Subroutine Inverte matriz
    !-----------------------------------------------------


    !-------------------------------------------------------MAtrizes semelhantes--------!
  Subroutine Calcula_matriz_semelhante(A,nA,mA,S,nS,B,nB,mB,produto)

    ! Retorna em produto uma matriz "semelhante" a Matriz S.  produto= A^T*S*B
    ! Essa função é mais geral pois permite o fazer o produto a esquerda e a direita com matrizes diferentes e não necessariamente quadradas.
    ! A tem dimensão nAxmA, S é quadrada e tem dimensão nS e B tem dimensão nBxmB.

    Implicit none
    Integer, intent(in) :: nS, nA, mA, nB, mB
    Double precision, intent(in) :: A(nA,mA), S(nS,nS), B(nB,mB)
    Double precision, intent(out), dimension(1:mA,1:mB) :: produto
```



```
    Double precision :: M1(mA,nS),M2(mA,mB)

    !print*, "dimensão de A",nA,mA ,"dimensão de S", nS, "Dimensao de B",nB,mB
    if(nA==nS) then
     M1=matmul(transpose(A),S)
    ! print*, "Tamanho de M1", size(M1)
    else
     print*,"Impossível multiplicar matrizes, dimensões não batem"
     stop
    end if
    if(nS==nB) then
     M2=matmul(M1,B)
     !print*, "Tamanho de M2", size(M2)
    else
     print*,"Impossível multiplicar matrizes, dimensões não batem"
     stop
    end if
    produto(1:mA,1:mB)=M2(1:mA,1:mB)


  End subroutine Calcula matriz semelhante


    !-----------------REGISTRO DAS MATRIZES DE RIGIDEZ E DE MASSA EM ARQUIVO .txt-------------------------------!


  Subroutine Escreve matrizes(r,M,S)

    Implicit none

    Integer:: nnos,r
    Double precision:: M(r,r), S(r,r)

    !Variaveis locais
    Integer:: i,j


    Open(43, file= '/home/daniela/Documents/fortran/placa naoconforme/Saida/Matriz rigidez.txt', status='old')
    Open(44, file= '/home/daniela/Documents/fortran/placa naoconforme/Saida/Matriz massa.txt', status='old')
    Write(43,*),"Dimensao da matriz:",r
    Write(44,*),"Dimensao da matriz:",r
    Do j=1,r
      Do i=1,r
        Write(43,*),S(i,i)
        Write(44,*),M(i,i)
      End do
    End do

    ! escrever em forma de matriz:
    Write(43,*)
    Write(43,*)
    Write(44,*)
    Write(44,*)
    Do i=1,r
      Write(43,'(12f9.3)'),S(i,:)
      Write(44,'(12f9.3)'),M(i,:)
    End do

  End subroutine Escreve matrizes
```



```
    !----------------IMPRESSÃO DOS AUTOVALORES E AUTOVETORES EM ARQUIVO .txt--------------------------------!

  Subroutine Imprime_autovetores(r,w,S)

    Implicit none
    Integer, intent(in):: r
    Double precision, intent(in) :: w(r),S(r,r)

    !Variaveis locais
    Integer :: i,j

    print*,"Estou em imprime autovetores, r=",r

    Open(45,file= '/home/daniela/Documents/fortran/placa naoconforme/Saida/Autovetores.txt' ,status='old')
    Write(45,*) r
    Do i=1,25              ! escreve 25 dos r primeiros autovalores
      Write(45,*) w(i)
    End do
    Write(*,*),"Primeiro autovalor:",w(1)

    !Do i=1,r
     !Write(45,*)
     !Write(45,*) "Autovalor",i,":", w(i)
     !Write(45,*)
     !Write(45,*)"autovetor associado:"
     !Do j=1, r
      !Write(45,'(f24.20)'),S(j,i)
     !End do
    !End do
    Close(45)

  End subroutine Imprime autovetores

    !------------------------------ ARQUIVOS PARA O SCILAB (função fec)---------------------------------------!
    ! Só vale para o anzats linear. arrumar outra função do scilab-------------------------!

  Subroutine Dados plotagem(nedges,nnos,nele,x,y,S,w,T,r,fix)

    Implicit none
    Integer, intent(in) :: nedges, nnos,r,nele, T(7,nele),fix(nnos)
    Double precision, intent(in) :: x(nnos), y(nnos), w(r),S(r,r)

    ! Variáveis locais
    Integer :: i,j,k,l,p ! contadores

    Open (50,file= '/home/daniela/Documents/fortran/placa_naoconforme/Saida/coordxyO2_scilab.txt', status='old')
    p=nnos-nedges                                  ! é o numero de vertices da malha, sem considerar os pontos medios dos triangulos

    Do i=1,p
      Write(50,*) x(i), y(i)
    End do

    Open (51,file= '/home/daniela/Documents/fortran/placa_naoconforme/Saida/TO2_scilab.txt', status='old')
    Do i=1,nele
      Write(51,*) T(1,i), T(2,i), T(3,i), T(4,i), T(5,i), T(6,i), T(7,i)
```



```
    End do

    Open (52,file= '/home/daniela/Documents/fortran/placa_naoconforme/Saida/autofuncO2_scilab.txt', status='old')

    Do k=1,25 ! ao inves de escrever todos os r autovetores .. escreverei só os 25 primeiros autovetores.
      l=1
      Do j=1,p
        !Do i=1,nnos
        !se o vertice estava fixo, escreva 0.0, caso contrario, escreva o proximo elemento do vetor S(i)
        if (fix(j).eq.1) then
          Write(52,*) 0.00
          !i=i+1
        else
          !Write(52,*) S(l+r*(k-1))
          Write(52,*) S(l,k)
          l=l+1
        end if

      End do
    End do

    Close(50)
    Close(51)
    Close(52)

  End subroutine Dados_plotagem
```

B.2 Formulação de quinto grau - *conforme*

```
PROGRAM Dani

!-----------------------------------------------------------------------
! *.......Elementos Finitos : Elemento triangular com aproximacão de
quinto grau - variacão do elemento conforme ....**
!
!  Este programa calcula a discretização da integral variacional para o
problema de autovalores do
!  bilaplaciano  e monta as matrizes de massa e rigidez para o problema de
autovalores generalizados.
!  Este por sua vez é resolvido pela funcão dsygv do Lapack.
!  Ao final, este programa prepara arquivos de leitura para a plotagem
gráfica pelo Scilab.
!
!  Nesta variacão do elemento conforme, usamos interpolacão polinomial
para alterar as variáveis nodais,
! criando um elemento finito tal que é possivel fazer a aproximacão no
triangulo padrão e evitar assim diversas
! inversões da matriz que relaciona coeficientes e variáveis nodais.
!
!
!
!  Uses:  Subroutines dgtrf e dgtri do lapack para inverter matrizes
!         SUBROUTINE dsygv( ITYPE, JOBZ, UPLO, N, A, LDA, B, LDB, W, WORK,
lwork, info) do Lapack
!         para a solução do problema de autovalores generalizados
!
!         Subrotina ... do lapack para inversao de matrizes
!-----------------------------------------------------------------------
!'/home/danielapm/Documents/Malha/triangle O2/Qua.5.ele' lab
!'/home/daniela/Documentos/Malha/triangle O2/Quar.1.ele' casa
!'/home/posmap/daniela/Documentos/Malha/triangle O2/Quar.1.ele' redeIME

  Implicit none
  ! Variáveis da modelagem. Usaremos alocação dinamica de memória

  Integer :: nele,nnos,nospe,nedges
  ! Numero de elementos triangulares da malha, numero de nós da malha
  ! nº de nós por elemento triangular,numero de aresta, respectivamente
  !Serão lidos nos arquivos .ele ..node e .edge do gerador de malhas

  Integer :: nv vertice,nvertices ! numero de variáveis nodais por vertice
de triangulo e numero total de vertices da malha

  Integer :: dimensao ! dimensao das matrizes de massa e rigidez globais

  Integer, allocatable, dimension (:,:) :: T
  ! Matriz que contem o numero de cada elemento triangular e seus
  !respictivos vertices segundo a numeração global da malha

  Integer, allocatable, dimension (:) :: fix, bordo
  ! vetor que armazena a informacão sobre o nós que ficarão fixos e sobre
  ! os nós do bordo, respectivamente.

  Integer :: r
  ! dimensão final das matrizes de massa e rigidez,
  !i.e. depois das condições de contornono serem add.
  r<=dimensao

  Integer :: bordo fixo=1,nv fix,nm fix
  !bordo fixo recebe os valores 1 ou 0 e indica se o bordo está ou não
fixo. Essa variavel tb pode ser booleana
  !nnfix é a quantidade de vertices fixos, sem contar aqueles do bordo
```



```
  Integer, allocatable, dimension (:,:) :: edge
  ! Matriz que armazenara os pontos iniciais e finais de cada aresta.

  !Double precision, allocatable, dimension (:) :: x_fix,y_fix
  !Vetores que armazenam as coordenadas dos nós que ficarão fixos
  !essas coordenadas sao lidas no arquivo condicoes de contorno

  Double precision, allocatable, dimension (:) :: x,y
  !Vetores que armazenam as coordenadas (x,y) de cada nó:
  !(x(i), y(i)) são as coordenadas do i-ésimo nó.

  Double precision, allocatable, dimension (:,:) :: normal
  !Matriz que contem as coordenas dos vetores normais

  Double Precision, allocatable, dimension (:,:) :: S, M
  ! Matrizes de Rigidez e de Massa globais.

  Integer:: i,j,k          !Contadores
  Integer:: contabilizou cc ! indica se as condições de contorno foram ou
nao contabilizadas.
  !Aqui, a variavel pode ser do tipo booleana.
  Integer :: ierr, dierr   ! indicadores de erro na alocacão de memória
  Character (len=15) :: malha


  ! Variáveis que a funcão dsygv necessita:

  Double precision,allocatable, dimension(:) :: w, work
  !w é um vetor que recebera os autovalores
  Character :: jobz='V', uplo='U'
  Integer :: info, itype=1, lda, ldb, lwork


  !-----------------------------Início---------------------------------------
  ---------------------------!

   nv vertice = 6   ! num de var nodais por vertice de triangulo

  !------ Leitura de arquivos .edge, .node e .ele para alocação das
  variaveis-------------------!

  !Leitura do arquivo .ele gerado pelo Triangle. Malha com ordem 2
   Open(40,file='/home/daniela/Documents/Malha/triangle O2/quadrado.5.ele',
    status='old')
   Read(40,*) nele, nospe
   !print*, "Cada elemento triangular tem ",nospe,"nós"

   !Leitura do arquivo .node gerado pelo Triangle
   Open (41,file =
   '/home/daniela/Documents/Malha/triangle O2/quadrado.5.node', status=
   'old')
   Read (41,*) nnos

   !Leitura do arquivo .edge gerado pelo Triangle - malha de ordem 2
   Open (42, file=
   '/home/daniela/Documents/Malha/triangle O2/quadrado.5.edge', status=
   'old')
   Read(42,*) nedges

   !Leitura do arquivo condicoes de contorno.txt, definido pelo usuario.
   !Open (46,
   file='/home/daniela/Documents/fortran/placa conforme nova/Entrada/condico
   es de contorno.txt',status='old')
   !Read(46,*) bordo_fixo
```



```
    nvertices=nnos-nedges
    dimensao=6*nvertices+nedges   !dimensao inicial das matrizes globais.
    esse é o numero total de variaveis nodais: 6por vertices e 1 por aresta


    ! Alocação de variaveis
    allocate (T(7,nele), stat = ierr)
    if (ierr/=0) then
     print*,"Erro de alocacão do variavel T"
     stop
    end if

    allocate (bordo(nnos),fix(nnos), stat = ierr)
    if (ierr/=0) then
      print*,"Erro de alocação do variavel bordo"
      stop
    end if

    allocate (x(nnos), y(nnos), stat = ierr)
    if (ierr/=0) then
      print*,"Erro de alocacão das variaveis x e y"
      stop
    end if

    allocate (edge(3,nedges), normal(2,nedges), stat = ierr)
    if (ierr/=0) then
      print*,"Erro de alocação do variavel edge"
      stop
    end if

    ! Leitura das demais informações dos arquivos provenientes do gerador
    de malhas
    Call Le_arquivos(nele, nnos, nedges, x, y, bordo, T, bordo_fixo,edge)

    fix=0

  !Esta funcão verifica os vértices da malha que estão fixos e guarda a
  informação no vetor fix. Se fixo, fix(i)=1, caso contrario, fix(i)=0
    Call Define condicoes contorno(nnos,nvertices,dimensao,r,fix,bordo,
    bordo fixo,nv fix,nm fix) ! Varredura na malha, só diz respeito aos
    nós, e não às variáveis
    print*,"sai de def cc"

  !Depois de definir as condicoes de contorno, sabemos a dimensão final r
  das matrizes de massa e rigidez.
    allocate (M(r,r), S(r,r), stat = ierr)
    if (ierr/=0) then
      print*,"Erro de alocacão das variaveis S e M"
      stop
    end if

    contabilizou_cc = 0
    print*,"entrei em de def normais"
    Call Define vetores normais(nedges,nnos,edge,normal,x,y)
    print*,"sai de def normais"

  !-------FUNÇÃO PRINCIPAL: MONTAGEM DAS MATRIZES PARA O PROBLEMA DE
  AUTOVALORES GENERALIZADOS-------!
  !Monta matrizes de rigidez e massa, locais e globais, bem como todas as
  matrizes auxiliares necessárias

    print*,"entrei em monta matriz"
    Call Monta matrizes&
    (nele, nnos, nospe, nedges,edge,normal,x, y, fix, T, S, M,r,
    contabilizou_cc,nv_vertice, dimensao,nvertices,nm_fix,nv_fix)
```



```
    !if (allocated (bordo)) deallocate (bordo, stat=dierr)
    !if (dierr/=0) then
    !  print*,"Erro de desalocação da variavel bordo"
    !  stop
    !end if

    !if (allocated (edge)) deallocate (edge, stat=dierr)
    !if (dierr/=0) then
    !  print*,"Erro de desalocação da variavel edge"
    !  stop
    !end if

    !if (allocated (normal)) deallocate (normal, stat=dierr)
    !if (dierr/=0) then
    !  print*,"Erro de desalocação da variavel normal"
    !  stop
    !end if

  !Registra em arquivo as matrizes de Rigidez e de Massa iniciais,
  !pois elas serão modificadas pela função dsygv
  ! if (contabilizou_cc .eq. 1) then
     !Call Escreve matrizes(r,M,S)
  !else
    !Call Escreve matrizes(dimensao,M,S)
  ! end if

  !Inicialização de variáveis que serão usadas na função dsygv
    lda=r
    ldb=r
    lwork=3*r-1

  !Alocação das variaveis w e work. Fizemos isso só aqui, pois
  precisávamos do valor da dimensão final r das
  ! matrizes
    allocate (w(r), work(lwork), stat = ierr)
    if (ierr/=0) then
      print*,"Erro de alocação das variáveis w e work"
      stop
    end if

  ! Resolve o problema generalizado de autovalores do tipo Au=B(lambda)u:
    print*,"entrei em DSYGV"
    Call dsygv (itype, jobz, uplo, r, S, lda, M, ldb, w, work, lwork, info)
    Print*, "info DSYGV=",info    ! Controle da funcão dsygv: info=0 indica
    sucesso nos calculos
    Write(*,*),"Primeiro autovalor:",w(1)
    !write(*,*), "Nao imprimi dados plotagem e autovalores"
  ! Impressão dos autovalores e autovetores em arquivo
    Call Imprime_autovetores(r,w,S)

  ! Prepara arquivos para o scilab
    print*,"entrei em Dados Plotagem"
    Call Dados plotagem(nedges,nnos,nele,x,y,S,T,r,fix)
    print*,"Ufa, acabou!"

  END PROGRAM Dani

  !--------------------Subrotinas que serão usadas no programa
  principal-----------------------------------------------------!

  !----------------------------------LEITURA DE
  ARQUIVOS-------------------------------------------------!

Subroutine Le_arquivos (nele, nnos, nedges, x, y, bordo, T, bordo_fixo, edge)
```

```
230     ! Subrotina diferente daquelas das outras implementações pois decidimos
        uma ordem para os vetores arestas na leitura.
231
232       Implicit none
233       !------- Variaveis dummy-------------!
234       Integer, intent(in) :: nele, nnos, bordo fixo, nedges
235       Integer, intent(out) :: T(7,nele), bordo(nnos)
236       ! T é matriz (nospe+1)x(nele) que na coluna j guarda o numero
237       ! do triangulo i, e seus nós (i 1,i 2,i 3,i 4,i 5 e i 6) nas linhas
          sequintes, segundo a numeração global da malha.
238       ! As tres primeiras são vertices de triangulo e os outros tres sao
          pontos medios das arestas.
239
240       Double precision, intent(out) ::  x(nnos), y(nnos)   ! As posições desse
          vetor de inteiros recebem valores 0 ou 1. Os vertices que estao no
          bordo sao marcados com 1
241       Integer, intent(out) :: edge(3,nedges)
242       ! Matriz que guarda o numero da aresta da malha e os pontos final e
          inicial dessas,
243       ! nessa ordem, nas linhas 1,2 e 3 respectivamente.
244
245       !----- Variáveis locais-----------!
246       Integer:: i,j,maior
247       Double precision :: c,b
248
249       !Leitura do arquivo .ele gerado pelo Triangle. Malha com ordem 2
250       Do j=1,nele
251         Read(40,*) T(1,j),T(2,i),T(3,i),T(4,i),T(5,i),T(6,i),T(7,i)
252       End do
253       Close(40)
254
255       !Leitura do arquivo .node gerado pelo Triangle
256       Do i=1,nnos
257         Read (41,*) c, x(i), y(i), b, bordo(i) ! ! Malha quadrado e
            tri escaleno que têm 5 parametros pra ler
258       End Do
259       Close (41)
260
261       !Leitura do arquivo .edge gerado pelo Triangle - malha de ordem 2
262       !NOVO: Ordenação do vetor edge de modo que edge(2,i) guarde sempre o nó
          de menor indice.
263       Do i=1, nedges
264         Read(42,*) edge(1,j),edge(2,j),edge(3,j)
265         If(edge(2,i) > edge(3,i)) then
266           maior=edge(2,i)
267           edge(2,j)=edge(3,j)
268           edge(3,i)=maior
269         End if
270       End do
271       Close(42)
272
273     End subroutine Le arquivos
274
275   !-------------------------------DEFINIÇÃO DOS VETORES
      NORMAIS-----------------------------------------------!
276
277     Subroutine  Define vetores normais(nedges,nnos,edge,normal,x,y)
278
279       Implicit none
280       !------ Variaveis dummy---------!
281       Integer, intent(in) :: nedges, nnos
282       Double precision, intent(in) :: x(nnos),y(nnos)
283       Integer, intent(in):: edge(3,nedges)
284
285       Double precision, intent(out) :: normal(2,nedges)
```



```
287
288
289       !-----Variáveis locais-----------!
290       Integer :: i,maior
291       Double precision :: normal unitaria(2,nedges),norma(nedges)
292
293       !As coordenadas desses pontos são [x(edge(2,i)); y(edge(2,i))] e
          [x(edge(2,i)) ; y(edge(2,i))]
294       !As coordenadas do vetor aresta sao:
          [x(edge(3,i))-x(edge(2,i));y(edge(3,i))-y(edge(2,i))]
295       !As coordenadas dos vetores normais sao definidas pela rotação da aresta
          i,
296       !por um angulo de 90 graus no sentido anti-horário
297       print*,"Estou dentro de def normais"
298       Do i=1,nedges
299         !Definimos as coordenadas x e y do i-ésimo vetor normal:
            [normal(1,i);normal(2,i)]
300         normal(1,i)=y(edge(2,i))-y(edge(3,i))
301         normal(2,i)=x(edge(3,i))-x(edge(2,i))
302       End do
303
304       Do i=1,nedges
305         norma(i) = sqrt(normal(1,i)**2 + normal(2,i)**2)
306
307         normal unitaria(1,i)=normal(1,i)/norma(i)
308         normal unitaria(2,i)=normal(2,i)/norma(i)
309
310         normal(1,i) = normal unitaria(1,i)
311         normal(2,i) = normal unitaria(2,i)
312       End do
313       print*,"Saindo de Define vetores normais"
314
315     End Subroutine Define vetores normais
316
317
318     !-----------------------------DEFINIÇÃO DAS CONDIÇÕES DE
        CONTORNO---------------------------------------!
319
320     ! Esta subrotina define as condicões de contorno que serao utilizadas.
321     !(Nesse caso), define quais e quantos nós da malha permanecerao fixos.
322
323      Subroutine Define condicoes contorno(nnos,nvertices,dimensao,r,fix,bordo,
        bordo_fixo,nv_fix,nm_fix)
324
325       Implicit none
326
327       Integer, intent(in) :: nnos, bordo(nnos), bordo fixo, dimensao,nvertices
328       Integer, intent(out) :: r, nv fix, nm fix, fix(nnos)
329
330       !Variávies locais
331       Integer :: i,k
332
333       if (bordo_fixo.eq.1) then
334         print*,"BORDO FIXO!"
335         Do i=1,nnos
336           fix(i)=bordo(i) ! Condição de bordo fixo. O vetor fix é igual ao
              vetor bordo
337         end do
338       End if
339
340       nv fix=0 ! numero de vertices fixos
341       nm fix=0 ! numero de ptos medios de aresta fixos
342
343     !Contando quantos pontos fixo sao vertices
344       Do i=1,nvertices
```



```
348       End do
349
350     !Contando quantos pontos fixos sao pontos medios
351       Do i=1+nvertices,nnos
352         if ((fix(i).eq. 1) ) then  ! se o vertice i esta fixo,
353           nm fix=nm fix+1
354         end if
355       End do
356
357       r=dimensao-(3*nv_fix+nm_fix) ! dimensão das matrizes de massa e rigidez
358
359     End Subroutine Define condicoes contorno
360
361   !-------------------------------------------------------------------------------
      ------------------------------!
362
363   !------------------------MONTAGEM DAS MATRIZES DE MASSA E RIGIDEZ
      GLOBAIS----------------------------------!
364
365     Subroutine Monta_matrizes&
366     (nele,nnos,nospe,nedges,edge,normal,x,y,fix,T,S,M,r,contabilizou cc,
        nv vertice,dimensao,nvertices,nm fix,nv fix)
367
368       Implicit none
369
370       Integer, intent(in) :: nele, nnos, nospe,r, nedges, dimensao , nvertices,
          nv_vertice
371       Integer, intent(in) :: T(7,nele), fix(nnos), edge(3,nedges), nv fix, nm fix
372       Integer, intent(out) :: contabilizou cc
373       Double precision, intent(in) :: x(nnos), y(nnos), normal(2,nedges)
374       Double precision, intent(out) :: S(r,r), M(r,r)
375
376
377       !---------------------- Variáveis
          locais-----------------------------------!
378
379
380       Double Precision :: a(3),b(3),Ji
381       ! a e b sao vetores que contem as coordenadas do vetor normal unitário
          n=(a,b), em cada triangulo Ti
382
383       Double Precision, Dimension(21,21) :: Si, Mi ! MAtrizes de rigidez e
          massa elementares
384       Double Precision, Dimension(21,21) :: inv A,C,D,E,P,NC,F ! Matrizes
          auxiliares na montagem de Mi e Si.
385       Double precision, Dimension (10,10) :: P1,D1,E1 ! Matrizes auxiliares na
          montagem de Mi e Si.
386       Double precision, allocatable :: S (:,:), M (:,:)
387       !matrizes auxiliares para a montagem de M e S (globais)
388
389       Integer :: pot(21,2)
390       ! Matriz auxiliar para montagem das matrizes P e P1.
391       ! Essa matriz guarda as potencias de x e y para uso em.
392       ! A potencia do x está na coluna 1 e a do y, na coluna 2
393
394       Integer, allocatable :: vi(:)
395       ! Vetor auxiliar que guarda a numeracao global dos nos do iésimo
          triangulo.
396
397       Integer :: ui(21)
398       ! Vetor auxiliar que guarda as posições globais de cada variável nodal
          do i-esimo triangulo.
399       ! Será usado na montagem das matrizes globais
400
401       Integer :: ierr, i,j,k, n_i(3), fim(3),inicio(3), inicio_local(nedges),
```



```
          iniciais e finais de cada aresta local
404
405       Double precision, parameter :: sigma = 0.3D0 ! parametro da edp:
          coeficiente de poisson
406
407
408       print*,"Dimensao inicial da matriz global:",dimensao
409
410       Allocate (vi(nospe), S (dimensao,dimensao), M (dimensao,dimensao), stat=
          ierr)
411       if (ierr/=0) then
412         print*,"Erro de alocacão das variaveis vi,S,M"
413         stop
414       end if
415
416       !Inicializacão de S  e M  com zero
417       S = 0.0D0
418       M = 0.0D0
419
420     !Montagem da matriz pot (potencias de x e y)
421       k=1
422       Do i=0,5   ! 5 é  grau da aproximação
423         Do j=0,i
424           pot(k,1)=i-j
425           pot(k,2)=j
426           k=k+1
427         End do
428       End do
429
430        ! Leitura da matriz que relaciona os coeficientes da aproximação com as
          (novas) variáveis nodais.
431        !Nesse programa, essa independe da geometria da malha
432
433       Open(120,file=
          '/home/daniela/Documents/fortran/placa conforme nova/Entrada/inversaDani.t
          xt', status='old')
434       Do i=0,2
435         !print*
436         !print*,"Coluna de ",1+5*i,"a",5+5*i
437         Do i=1,21
438           Read(120,*) inv A(i,i*5+1),inv A(i,i*5+2),inv A(i,i*5+3),inv A(i,i*5+4),
              inv A(i,i*5+5)
439           !
              print*,inv A(i,i*5+1),inv A(i,i*5+2),inv A(i,i*5+3),inv A(i,i*5+4),inv
              A(i,i*5+5)
440         End do
441       End do
442
443       Do i=1,21
444         Read(120,*) inv A(i,16),inv A(i,17),inv A(i,18),inv A(i,19),inv A(i,20),
            inv A(i,21)
445       End do
446
447       Close(120)
448
449       !Montagem da Matriz das integrais P e P1.
450       !Essas matriz sao constantes e não depende da geometria da malha
451       Call Monta_matriz_P (P1,pot,3,10) ! para ser usada em Si
452       Call Monta_matriz_P (P,pot,5,21)  ! para ser usada em Mi
453       !Call Escreve Matriz em Arquivo (P1,21,60,"Matriz P.txt","old")
454
455     !------------Escrita de arquivo sobre o estudo da inversao de
        matrizes----------------!
456     !
        Open(110,file='/home/daniela/Documents/fortran/placa_conforme/Estudo_invers
```

```
460
461
462     !   -------------------------LAÇO NO NÚMERO DE TRIANGULOS
        ---------------------------------!
463
464       Do i=1,nele
465         !Essa matriz é remontadas para cada triangulo e não pode sofrer
          influencia da que foi montada no laço anterior
466         D=0.0D0
467
468         Do j=1,nospe
469           vi(j)=T(j+1,i) ! Os vertices do triangulo i são: T(2,i), T(3,i),
            T(4,i), T(5,i), T(6,i) e T(7,i)
470         End Do
471
472         Ji=((x(vi(2)))-x(vi(1)))*(y(vi(3))-y(vi(1))) - (x(vi(3))-x(vi(1)))*(y(vi(2
            ))-y(vi(1)))  !Determinante da matriz de mudança de coordenadas
473
474         !print*, "triangulo i, Ji =",i,Ji
475
476         !--------------LISTA LOCAL DE VETORES NORMAIS
          UNITÁRIOS-----------------------------------------------------!
477         !--------Definição do vetor que guarda o indice global do vetor
          normal, nas posições locais 1,2 ou 3:
478         !Vamos estabelecer uma numeração local para os vetores normais
          globais. Cada elemento triangular i tem 3 vetores normais: n i(j),
            j=1,2,3
479         !Essa numeracao local é a mesma para as arestas.
480
481         j=1
482         Do while (j.LE.nedges)
483           if( ((vi(1) .eq. edge(2,j)) .and. (vi(2) .eq. edge(3,j))) .or. ((vi(2)
              .eq. edge(2,j)) .and. (vi(1) .eq. edge(3,j)))) then
484             n i(3)=j !n i(3) é vetor oposto ao vertice 3 e é o j-ésimo vetor
                normal da tabela normal
485             if ( vi(1) == edge(2,j)) then
486               inicio_local(j)=1
487               fim local(j)=2
488             else
489               inicio_local(j)=2
490               fim local(j)=1
491             end if
492
493             j=nedges+1
494           else
495             j=j+1
496           end if
497         End do
498
499         j=1
500         Do while (j.LE.nedges)
501           if( ((vi(2) .eq. edge(2,j)) .and. (vi(3) .eq. edge(3,j))) .or. ((vi(3)
              .eq. edge(2,j)) .and. (vi(2) .eq. edge(3,j)))) then
502             n i(1)=j !n i(1) é é vetor oposto ao vertice 1 e é  o j-ésimo
                vetor normal da tabela normal (tal tabela seque a indexação da
                lista de arestas)
503             if ( vi(2) == edge(2,j)) then
504               inicio local(j)=2
505               fim_local(j)=3
506             else
507               inicio local(j)=3
508               fim local(j)=2
509             end if
510
511             j=nedges+1
```



```
514           end if
515         End do
516
517         j=1
518         Do while (j.LE.nedges)
519           if( ((vi(1) .eq. edge(2,j)) .and. (vi(3) .eq. edge(3,j))) .or. ((vi(3)
              .eq. edge(2,j)) .and. (vi(1) .eq. edge(3,j)))) then
520             n_i(2)=j !n_i(2) é é vetor oposto ao vertice 2 e é o j-ésimo vetor
                normal da tabela normal
521             if ( vi(1) == edge(2,j)) then
522               inicio_local(j)=1
523               fim local(j)=3
524             else
525               inicio_local(j)=3
526               fim local(j)=1
527             end if
528
529             j=nedges+1
530           else
531             j=j+1
532           end if
533         End do
534         !print*,"inicio local e fim local"
535         !print*, inicio_local,fim_local
536         !print*,
537         !Pontos iniciais e finais (globais) das arestas locais. A numeração
          local das arestas é a mesma das normais:
538         Do k=1,3
539           inicio(k)=edge(2,n i(k))
540           fim(k)=edge(3,n_i(k))
541           !print*,"Aresta",k," do triangulo",i,"tem inicio e fim global nos
              vertices:",inicio(k),fim(k)
542           !print*,"Estes correspondem aos pontos locais
              ",inicio local(n i(k)),fim local(n i(k))
543           !print*,"ni(k)",n i(k)
544         End do
545
546         !O triangulo i,tem vetores normais unitarios n i(j), i=1,2,3 e estes
          tem coordenadas a(i),b(i), i=1,2,3
547         Do j=1,3
548           a(j)=normal(1,n i(j))
549           b(j)=normal(2,n i(j))
550         End do
551
552       !Montagem do Vetor que guarda o indice global de cada variavel nodal do
          triangulo Ti, em suas posições 1,2,...,21
553         Call Monta_Var nodais (vi,ui,nvertices,nv vertice)
554
555         !Call Monta matriz C (C,x,y,vi,i,nele,nospe,a,b)  ! A matriz C
            relaciona as variaveis nodais com os parametros da aproximação. alfa=C*u
556         !Call Monta matriz D (D,x,y,vi,nospe,5,21) ! A matriz D, que relaciona
            os parametros a1,...a21 com os parametros alfa's, da aproximação em Ti
            e em To respectivamente
557         !Call Monta matriz D (D1,x,y,vi,nospe,3,10) ! É como a matriz D, mas
            de dim menor.
558         !Call Monta_matriz_E (E,x,y,vi,nospe,5,21,pot)
559         !Call Monta matriz E (E1,x,y,vi,nospe,3,10,pot)
560
561         Call Monta_matriz_F(F,21,nnos,nospe,vi,x,y)
562         Call Monta matriz NC BETA (NC,i,x,y,inicio,fim,inicio local,fim local,
            nedges,n i, nnos)
563         !Call Monta matriz NC (NC,i,x,y,inicio,fim,inicio local,fim local,nedges)
564
565         !Call Escreve Matriz em Arquivo (NC,21,61,"Matriz NC.txt","old") !
            PAra teste
```



```
568         !Call Escreve_Matriz_em_Arquivo (E1,21,63,"Matriz_E1.txt","old") !
            PAra teste
569         !Call Escreve matrizes 1(21,P,C,D,D)
570
571         Call Monta matriz Mi (Ji,F,NC,P,inv A,Mi) !Matriz de massa elementar
572         Call Monta matriz Si (Ji,F,P1,inv A,NC,Si,sigma,x,y,vi,nnos,nospe) !
            Matriz de rigidez elementar
573
574         !Call Escreve matrizes(21,Mi,Si)
575
576         !--------Montagem das matrizes de rigidez e massa globais S  e M :
            -----------!
577         Call Monta matrizes globais(S ,M ,Si,Mi,ui,dimensao)
578
579
580       End Do  !--------------------- FIM DO LAÇO (em i) NO NUMERO DE
          TRIÂNGULOS----------------------------!
581       Close(110)
582       !print*,"inicio local e fim local"
583        !print*, inicio_local,fim_local
584       print*,
585
586       Call Contabiliza_condicoes_de_contorno (S_,M_,fix,nnos, contabilizou_cc,r,
          dimensao,nvertices, nv fix,nm fix)
587
588       !if (contabilizou cc .eq. 1) then
589       Call Redimensiona_matrizes(dimensao,r,M,S,M_,S_)
590       !else
591        ! Call Redimensiona matrizes(dimensao,dimensao,M,S,M ,S )! nesse caso,
          não há redimensionamento, as matrizes continuam com a mesma dimensao
592       !end if
593
594       !Call Escreve_matrizes(r,M,S)
595
596     End Subroutine Monta matrizes
597
598   !-------------------------------Subrotinas usadas em Monta Matriz
      ------------------------------------!
599
600
601     Subroutine Monta matriz NC BETA (NC,i,x,y,inicio,fim,inicio local,fim local
      ,nedges,n i, nnos)
602       ! Esta matriz relaciona dois conjuntos de tipos de variáves nodais no
          elemento triangular, para a aproximacão de 5º grau.
603       Implicit none
604
605       Double precision, intent(out) :: NC (21,21)
606       Double precision, intent(in) :: x(nnos),y(nnos)
607       Integer, intent(in) :: i,nedges, nnos
608       Integer, intent(in) :: inicio(3),fim(3),inicio local(nedges),fim local(
          nedges),n i(3)
609
610       !Variaveis locais
611       Integer:: i,k
612       Double precision :: l(3)
613
614       NC=0.0D0
615
616     !Blocos Id:
617
618       Do i=1,3
619        Do k=1,3
620          if(k==i) then
621            NC(i,k)=1.0D0
622            NC(j+12,k+9)=1.0D0
```



```
626        End do
627      End do
628
629
630      Do i=1,3  ! laco nas arestas do triangulo
631        l(i)=sqrt((x(fim(i))-x(inicio(i)))**2 + (y(fim(i))-y(inicio(i)))**2)
632
633       !Valores de u no ponto medio da aresta 1: !Posições onde sinal é
          relevante tem marcação '!', neste caso é no fim da aresta
634        NC(3+j,inicio local(n_i(j)))=0.5D0
635        NC(3+i,fim local(n i(i)))=0.5D0
636
637        NC(3+i,inicio local(n i(i))+3)= 0.15625D0*(x(fim(i))-x(inicio(i)))  !
638        NC(3+j,fim_local(n_i(j))+3)=-0.15625D0*(x(fim(j))-x(inicio(j)))     !
639        NC(3+i,inicio local(n i(i))+6)=0.15625D0*(y(fim(i))-y(inicio(i)))   !
640        NC(3+i,fim local(n i(i))+6)=-0.15625D0*(y(fim(i))-y(inicio(i)))     !
641
642        NC(3+i,inicio local(n i(i))+9)=0.015625D0*(x(fim(i))-x(inicio(i)))**2
643        NC(3+i,fim local(n i(i))+9)=0.015625D0*(x(fim(i))-x(inicio(i)))**2
644        NC(3+j,inicio_local(n_i(j))+12)=0.03125D0*(x(fim(j))-x(inicio(j)))*(y(fim
           (j))-y(inicio(j)))
645        NC(3+i,fim local(n i(i))+12)=0.03125D0*(x(fim(i))-x(inicio(i)))*(y(fim(i
           ))-y(inicio(j)))
646        NC(3+i,inicio local(n i(i))+15)=0.015625D0*(y(fim(i))-y(inicio(i)))**2
647        NC(3+i,fim local(n i(i))+15)=0.015625D0*(y(fim(i))-y(inicio(i)))**2
648
649       !Valores de u_x no ponto medio da aresta 1 : !Posições onde sinal é
          relevante tem marcação '!', neste caso é no inicio da aresta
650        NC(6+i,inicio local(n i(i)))=-1.875D0*(x(fim(i))-x(inicio(i)))/(l(i))**2) !
651        NC(6+j,fim_local(n_i(j)))=1.875D0*(x(fim(j))-x(inicio(j)))/(l(j))**2)    !
652
653        NC(6+i,inicio local(n i(i))+3)=-0.4375D0*((x(fim(i))-x(inicio(i)))**2)/(l
           (j)**2)
654        NC(6+i,fim local(n i(i))+3)=-0.4375D0*((x(fim(i))-x(inicio(i)))**2)/(l(i
           )**2)
655        NC(6+j,inicio_local(n_i(j))+6)=-0.4375D0*(x(fim(j))-x(inicio(j)))*(y(fim(
           j))-y(inicio(i)))/(l(i)**2)
656        NC(6+i,fim local(n i(i))+6)=-0.4375D0*(x(fim(i))-x(inicio(i)))*(y(fim(i
           ))-y(inicio(i)))/(l(i)**2)
657
658        NC(6+i,inicio local(n i(i))+9)= -0.03125D0*((x(fim(i))-x(inicio(i)))*(x(
           fim(i))-x(inicio(i)))**2) /(l(i)**2)  !
659        NC(6+j,fim_local(n_i(j))+9)=  0.03125D0*((x(fim(j))-x(inicio(j)))*(x(fim(
           j))-x(inicio(i)))**2) /(l(i)**2)     !
660        NC(6+i,inicio local(n i(i))+12)= -0.0625D0*((y(fim(i))-y(inicio(i)))*(x(
           fim(j))-x(inicio(j)))**2) /(l(j)**2)  !
661        NC(6+i,fim local(n i(i))+12)= 0.0625D0*((y(fim(i))-y(inicio(i)))*(x(fim(i
           ))-x(inicio(i)))**2) /(l(i)**2)      !
662        NC(6+j,inicio_local(n_i(j))+15)= -0.03125D0*((x(fim(j))-x(inicio(j)))*(y(
           fim(i))-y(inicio(i)))**2) /(l(i)**2) !
663        NC(6+i,fim local(n i(i))+15)= 0.03125D0*((x(fim(i))-x(inicio(i)))*(y(fim(
           j))-y(inicio(j)))**2) /(l(j)**2)     !
664
665        NC(i+6,i+18)=-(y(fim(i)) - y(inicio(i)))/l(i)
666
667       !Valores de u_y no ponto medio da aresta 1: !Posições onde sinal é
          relevante tem marcação '!', neste caso é no inicio da aresta
668        NC(9+i,inicio local(n i(i)))=-1.875D0*(y(fim(i))-y(inicio(i)))/(l(i))**2)  !
669        NC(9+j,fim_local(n_i(j)))=1.875D0*(y(fim(j))-y(inicio(j)))/(l(j))**2)      !
670
671        NC(9+i,inicio local(n i(i))+3)=-0.4375D0*(x(fim(i))-x(inicio(i)))*(y(fim(
           j))-y(inicio(j)))/(l(j)**2)
672        NC(9+i,fim local(n i(i))+3)=-0.4375D0*(x(fim(i))-x(inicio(i)))*(y(fim(i
           ))-y(inicio(i)))/(l(i)**2)
673        NC(9+j,inicio_local(n_i(j))+6)=-0.4375D0*((y(fim(j))-y(inicio(j)))**2)/(l
```

```
676         NC(9+j,inicio_local(n_i(j))+9)=-0.03125D0*((y(fim(j))-y(inicio(j)))*(x(
            fim(i))-x(inicio(i)))**2) /(l(i)**2)  !
677         NC(9+j,fim_local(n_i(j))+9)=0.03125D0*((y(fim(i))-y(inicio(i)))*(x(fim(i
            ))-x(inicio(j)))**2) /(l(j)**2)      !
678         NC(9+i,inicio_local(n_i(i))+12)=-0.0625D0*((x(fim(i))-x(inicio(i)))*(y(
            fim(i))-y(inicio(i)))**2) /(l(i)**2)  !
679         NC(9+i,fim_local(n_i(i))+12)=0.0625D0*((x(fim(i))-x(inicio(i)))*(y(fim(i
            ))-y(inicio(j)))**2) /(l(j)**2)       !
680         NC(9+i,inicio_local(n_i(i))+15)=-0.03125D0*((y(fim(i))-y(inicio(i)))*(y(
            fim(i))-y(inicio(i)))**2) /(l(i)**2) !
681         NC(9+j,fim_local(n_i(j))+15)=0.03125D0*((y(fim(j))-y(inicio(j)))*(y(fim(j
            ))-y(inicio(i)))**2) /(l(i)**2)      !
682
683         NC(j+9,j+18)=(x(fim(j)) - x(inicio(j)))/l(j)
684
685        End do
686
687       End Subroutine
688
689
690       Subroutine Monta matriz NC (NC,i,x,y,inicio,fim,inicio local,fim local,
          nedges)
691        ! Esta matriz relaciona dois conjuntos de tipos de variáves nodais no
           elemento triangular, para a aproximação de 5ºgrau.
692        ! Funciona bem para malhas simples. A versão beta é mais geral
693        Implicit none
694
695        Double precision :: NC (21,21)
696        Double precision :: x(*),y(*), l(3)
697        Integer :: i,nedges
698        Integer :: inicio(3),fim(3),inicio local(nedges),fim local(nedges)
699
700        !Variaveis locais
701        Integer:: i,k, sinal1, sinal2, resto
702
703        NC=0.0
704
705
706        resto=mod(i,2)
707
708        if(resto == 0) then ! i é par e portanto
709          sinal1=1
710          sinal2=-1
711         ! print*, "O triangulo", i," é par", sinal1,sinal2
712        end if
713        if(resto == 1) then ! i é impar
714         sinal1=-1
715         sinal2=1
716         !print*, "O triangulo", i," é impar", sinal1,sinal2
717       end if
718
719       !Blocos Id:
720
721       Do i=1,3
722        Do k=1,3
723          if(k==j) then
724            NC(i,k)=1.0
725            NC(i+12,k+9)=1.0
726            NC(j+15,k+12)=1.0
727            NC(i+18,k+15)=1.0
728          end if
729        End do
730       end do
731
732
```



```
734         l(j)=sqrt((x(fim(j))-x(inicio(j)))**2 + (y(fim(j))-y(inicio(j)))**2)
735         if (i==1) then
736           !Valores de u no ponto medio da aresta 1 -- verificar sinais
737            NC(3+j,2)=0.5
738            NC(3+i,3)=0.5
739            NC(3+i,5)=sinal2*0.15625*(x(fim(i))-x(inicio(i)))
740            NC(3+i,6)=sinal1*0.15625*(x(fim(i))-x(inicio(i)))
741            NC(3+j,8)=sinal2*0.15625*(y(fim(j))-y(inicio(j)))
742            NC(3+i,9)=sinal1*0.15625*(y(fim(i))-y(inicio(i)))
743            NC(3+i,11)=0.015625*(x(fim(i))-x(inicio(i)))**2
744            NC(3+j,12)=0.015625*(x(fim(j))-x(inicio(j)))**2
745            NC(3+i,14)=0.03125*(x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(i)))
746            NC(3+i,15)=0.03125*(x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(i)))
747            NC(3+j,17)=0.015625*(y(fim(j))-y(inicio(j)))**2
748            NC(3+i,18)=0.015625*(y(fim(i))-y(inicio(i)))**2
749
750           !Valores de u_x no ponto medio da aresta 1
751            NC(6+i,2)=sinal1*1.875*(x(fim(i))-x(inicio(i)))/(l(i)**2)
752            NC(6+i,3)=sinal2*1.875*(x(fim(i))-x(inicio(i)))/(l(i)**2)
753            NC(6+i,5)=-0.4375*((x(fim(i))-x(inicio(i)))**2)/(l(i)**2)
754            NC(6+i,6)=-0.4375*((x(fim(i))-x(inicio(i)))**2)/(l(i)**2)
755            NC(6+i,8)=-0.4375*(x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(i)))/(l(i
               )**2)
756            NC(6+j,9)=-0.4375*(x(fim(j))-x(inicio(j)))*(y(fim(j))-y(inicio(j)))/(l(j
               )**2)
757            NC(6+i,11)=sinal1*0.03125*((x(fim(i))-x(inicio(i)))*(x(fim(i))-x(inicio
               (j)))**2) /(l(j)**2)
758            NC(6+i,12)=sinal2*0.03125*((x(fim(i))-x(inicio(i)))*(x(fim(i))-x(inicio
               (j)))**2) /(l(i)**2)
759            NC(6+j,14)=sinal1*0.0625*((y(fim(j))-y(inicio(j)))*(x(fim(j))-x(inicio(
               i)))**2) /(l(i)**2)
760            NC(6+i,15)=sinal2*0.0625*((y(fim(i))-y(inicio(i)))*(x(fim(i))-x(inicio(
               i)))**2) /(l(i)**2)
761            NC(6+i,17)=sinal1*0.03125*((x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio
               (j)))**2) /(l(i)**2)
762            NC(6+i,18)=sinal2*0.03125*((x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio
               (j)))**2) /(l(j)**2)
763
764            NC(7,19)=-(y(fim(i)) - y(inicio(i)))/l(i)
765
766             !Valores de u_y no ponto medio da aresta 1
767            NC(9+i,2)=sinal1*1.875*(y(fim(i))-y(inicio(i)))/(l(i)**2)
768            NC(9+j,3)=sinal2*1.875*(y(fim(j))-y(inicio(j)))/(l(j)**2)
769            NC(9+i,5)=-0.4375*(x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(i)))/(l(i
               )**2)
770            NC(9+j,6)=-0.4375*(x(fim(j))-x(inicio(j)))*(y(fim(j))-y(inicio(j)))/(l(j
               )**2)
771            NC(9+i,8)=-0.4375*((y(fim(i))-y(inicio(i)))**2)/(l(i)**2)
772            NC(9+i,9)=-0.4375*((y(fim(i))-y(inicio(i)))**2)/(l(i)**2)
773            NC(9+j,11)=sinal1*0.03125*((y(fim(j))-y(inicio(j)))*(x(fim(j))-x(inicio
               (j)))**2) /(l(i)**2)
774            NC(9+i,12)=sinal2*0.03125*((y(fim(i))-y(inicio(i)))*(x(fim(i))-x(inicio
               (j)))**2) /(l(j)**2)
775            NC(9+i,14)=sinal1*0.0625*((x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(
               i)))**2) /(l(i)**2)
776            NC(9+j,15)=sinal2*0.0625*((x(fim(j))-x(inicio(j)))*(y(fim(j))-y(inicio(
               i)))**2) /(l(i)**2)
777            NC(9+i,17)=sinal1*0.03125*((y(fim(i))-y(inicio(i)))*(y(fim(i))-y(inicio
               (j)))**2) /(l(j)**2)
778            NC(9+i,18)=sinal2*0.03125*((y(fim(i))-y(inicio(i)))*(y(fim(i))-y(inicio
               (i)))**2) /(l(i)**2)
779
780            NC(10,19)=(x(fim(i)) - x(inicio(i)))/l(i)
781
782         end if
```



```
787           NC(3+j,3)=0.5
788           NC(3+i,4)=sinal2*0.15625*(x(fim(i))-x(inicio(i)))
789           NC(3+i,6)=sinal1*0.15625*(x(fim(i))-x(inicio(i)))
790           NC(3+j,7)=sinal2*0.15625*(y(fim(j))-y(inicio(j)))
791           NC(3+i,9)=sinal1*0.15625*(y(fim(i))-y(inicio(i)))
792           NC(3+i,10)=0.015625*(x(fim(i))-x(inicio(i)))**2
793           NC(3+j,12)=0.015625*(x(fim(j))-x(inicio(j)))**2
794           NC(3+i,13)=0.03125*(x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(i)))
795           NC(3+i,15)=0.03125*(x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(i)))
796           NC(3+j,16)=0.015625*(y(fim(j))-y(inicio(j)))**2
797           NC(3+i,18)=0.015625*(y(fim(i))-y(inicio(i)))**2
798
799
800           !Valores de u_x no ponto medio da aresta 2
801           NC(6+i,1)=sinal1*1.875*(x(fim(i))-x(inicio(i)))/(l(i)**2)
802           NC(6+i,3)=sinal2*1.875*(x(fim(i))-x(inicio(i)))/(l(i)**2)
803           NC(6+j,4)=-0.4375*((x(fim(j))-x(inicio(j)))**2)/(l(j)**2)
804           NC(6+i,6)=-0.4375*((x(fim(i))-x(inicio(i)))**2)/(l(i)**2)
805           NC(6+i,7)=-0.4375*(x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(i)))/(l(i
               )**2)
806           NC(6+i,9)=-0.4375*(x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(i)))/(l(i
               )**2)
807           NC(6+j,10)=sinal1*0.03125*((x(fim(j))-x(inicio(j)))*(x(fim(j))-x(inicio
               (i)))**2) /(l(i)**2)
808           NC(6+i,12)=sinal2*0.03125*((x(fim(i))-x(inicio(i)))*(x(fim(i))-x(inicio
               (j)))**2) /(l(i)**2)
809           NC(6+j,13)=sinal1*0.0625*((y(fim(j))-y(inicio(j)))*(x(fim(j))-x(inicio(
               i)))**2) /(l(i)**2)
810           NC(6+i,15)=sinal2*0.0625*((y(fim(i))-y(inicio(i)))*(x(fim(i))-x(inicio(
               j)))**2) /(l(j)**2)
811           NC(6+i,16)=sinal1*0.03125*((x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio
               (i)))**2) /(l(i)**2)
812           NC(6+j,18)=sinal2*0.03125*((x(fim(j))-x(inicio(j)))*(y(fim(j))-y(inicio
               (i)))**2) /(l(i)**2)
813
814           NC(8,20)=-(y(fim(j)) - y(inicio(j)))/l(j)
815
816             !Valores de u_y no ponto medio da aresta 2
817           NC(9+i,1)=sinal1*1.875*(y(fim(i))-y(inicio(i)))/(l(i)**2)
818           NC(9+i,3)=sinal2*1.875*(y(fim(i))-y(inicio(i)))/(l(i)**2)
819           NC(9+i,4)=-0.4375*(x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(i)))/(l(i
               )**2)
820           NC(9+j,6)=-0.4375*(x(fim(j))-x(inicio(j)))*(y(fim(j))-y(inicio(j)))/(l(j
               )**2)
821           NC(9+i,7)=-0.4375*((y(fim(i))-y(inicio(i)))**2)/(l(i)**2)
822           NC(9+j,9)=-0.4375*((y(fim(j))-y(inicio(j)))**2)/(l(j)**2)
823           NC(9+i,10)=sinal1*0.03125*((y(fim(i))-y(inicio(i)))*(x(fim(i))-x(inicio
               (i)))**2) /(l(i)**2)
824           NC(9+j,12)=sinal2*0.03125*((y(fim(j))-y(inicio(j)))*(x(fim(j))-x(inicio
               (i)))**2) /(l(i)**2)
825           NC(9+i,13)=sinal1*0.0625*((x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(
               j)))**2) /(l(j)**2)
826           NC(9+i,15)=sinal2*0.0625*((x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(
               i)))**2) /(l(i)**2)
827           NC(9+i,16)=sinal1*0.03125*((y(fim(i))-y(inicio(i)))*(y(fim(i))-y(inicio
               (j)))**2) /(l(j)**2)
828           NC(9+i,18)=sinal2*0.03125*((y(fim(i))-y(inicio(i)))*(y(fim(i))-y(inicio
               (i)))**2) /(l(i)**2)
829
830           NC(11,20)=(x(fim(i)) - x(inicio(i)))/l(i)
831
832         end if
833
834         if(i==3) then
835          !Valores de u no ponto medio da aresta 3
```



```
840           NC(3+j,7)=sinal2*0.15625*(y(fim(j))-y(inicio(j)))
841           NC(3+i,8)=sinal1*0.15625*(y(fim(i))-y(inicio(i)))
842           NC(3+i,10)=0.015625*(x(fim(i))-x(inicio(i)))**2
843           NC(3+j,11)=0.015625*(x(fim(j))-x(inicio(j)))**2
844           NC(3+i,13)=0.03125*(x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(i)))
845           NC(3+i,14)=0.03125*(x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(i)))
846           NC(3+j,16)=0.015625*(y(fim(j))-y(inicio(j)))**2
847           NC(3+i,17)=0.015625*(y(fim(i))-y(inicio(i)))**2
848
849         !Valores de u_x no ponto medio da aresta 3
850           NC(6+i,1)=sinal1*1.875*(x(fim(i))-x(inicio(i)))/(l(i)**2)
851           NC(6+i,2)=sinal2*1.875*(x(fim(i))-x(inicio(i)))/(l(i)**2)
852           NC(6+i,4)=-0.4375*((x(fim(i))-x(inicio(i)))**2)/(l(i)**2)
853           NC(6+j,5)=-0.4375*((x(fim(j))-x(inicio(j)))**2)/(l(j)**2)
854           NC(6+i,7)=-0.4375*(x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(i)))/(l(i
               )**2)
855           NC(6+j,8)=-0.4375*(x(fim(j))-x(inicio(j)))*(y(fim(j))-y(inicio(j)))/(l(j
               )**2)
856           NC(6+i,10)=sinal1*0.03125*((x(fim(i))-x(inicio(i)))*(x(fim(i))-x(inicio
               (j)))**2) /(l(j)**2)
857           NC(6+i,11)=sinal2*0.03125*((x(fim(i))-x(inicio(i)))*(x(fim(i))-x(inicio
               (j)))**2) /(l(i)**2)
858           NC(6+j,13)=sinal1*0.0625*((y(fim(j))-y(inicio(j)))*(x(fim(j))-x(inicio(
               i)))**2) /(l(i)**2)
859           NC(6+i,14)=sinal2*0.0625*((y(fim(i))-y(inicio(i)))*(x(fim(i))-x(inicio(
               j)))**2) /(l(j)**2)
860           NC(6+i,16)=sinal1*0.03125*((x(fim(j))-x(inicio(j)))*(y(fim(j))-y(inicio
               (j)))**2) /(l(i)**2)
861           NC(6+i,17)=sinal2*0.03125*((x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio
               (j)))**2) /(l(j)**2)
862
863           !contribuição da derivada normal em u_x da aresta 3.
864           NC(9,21)=-(y(fim(j)) - y(inicio(j)))/l(j)
865
866             !Valores de u_y no ponto medio da aresta 3
867           NC(9+j,1)=sinal1*1.875*(y(fim(j))-y(inicio(j)))/(l(j)**2)
868           NC(9+j,2)=sinal2*1.875*(y(fim(j))-y(inicio(j)))/(l(j)**2)
869           NC(9+i,4)=-0.4375*(x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(i)))/(l(i
               )**2)
870           NC(9+i,5)=-0.4375*(x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(i)))/(l(i
               )**2)
871           NC(9+i,7)=-0.4375*((y(fim(i))-y(inicio(i)))**2)/(l(i)**2)
872           NC(9+j,8)=-0.4375*((y(fim(j))-y(inicio(j)))**2)/(l(j)**2)
873           NC(9+i,10)=sinal1*0.03125*((y(fim(i))-y(inicio(i)))*(x(fim(i))-x(inicio
               (j)))**2) /(l(j)**2)
874           NC(9+j,11)=sinal2*0.03125*((y(fim(j))-y(inicio(j)))*(x(fim(j))-x(inicio
               (j)))**2) /(l(i)**2)
875           NC(9+i,13)=sinal1*0.0625*((x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(
               j)))**2) /(l(j)**2)
876           NC(9+i,14)=sinal2*0.0625*((x(fim(i))-x(inicio(i)))*(y(fim(i))-y(inicio(
               i)))**2) /(l(i)**2)
877           NC(9+j,16)=sinal1*0.03125*((y(fim(j))-y(inicio(j)))*(y(fim(j))-y(inicio
               (j)))**2) /(l(i)**2)
878           NC(9+i,17)=sinal2*0.03125*((y(fim(i))-y(inicio(i)))*(y(fim(i))-y(inicio
               (i)))**2) /(l(i)**2)
879
880           NC(12,21)=(x(fim(i)) - x(inicio(i)))/l(i)
881
882         End if
883
884       End do
885
886     End Subroutine
887
888
```

```
893     Double precision, intent(out) :: P(n,n)
894     Integer, intent(in) :: pot (21,2)
895     integer, intent(in) :: grau_aprox
896
897     ! Variaveis locais
898     Integer :: n,k,i,j
899     Double precision :: numerador,denominador !
900     !var auxiliares, pois Fat é fç inteira. o quociente tava dando um num
        inteiro.
901
902     !n é a dimensao de P, e depende do grau do polinomio que sera integrado.
903     !n=21 se grau_aprox =5
904     !n=10 se grau_aprox =3
905     !Essas sao as duas possibilidades nesse programa
906
907     j=(grau_aprox+2)*(grau_aprox+1)/2  ! soma de n+1 elementos da p.a. de
        razao 1,cujo 1º elemento é 1.
908     Do k=1,j
909       Do i=1,j
910         numerador=(Fat((pot(k,1)+pot(i,1)))*(Fat(pot(k,2)+pot(i,2))))
911         denominador=(Fat(pot(k,1)+pot(i,1)+pot(k,2)+pot(i,2)+2))
912         P(k,i)=numerador/denominador
913       End do
914     End do
915
916
917   CONTAINS
918     Integer Function Fat(n)           !Funcão que calcula o fatorial de um
        numero inteiro >= 0
919
920     Implicit none
921     Integer, intent (in) :: n
922     Integer :: produto, i
923
924     produto=1
925     Do i=1,n
926      produto=produto*i
927     End do
928     Fat=produto
929
930   End Function Fat
931
932
933   End Subroutine Monta_matriz_P
934
935   Subroutine Monta_matriz_F(F,n,nnos,nospe,vi,x,y)
936
937     Implicit none
938     Double precision, intent(out) :: F(1:n,1:n)
939     Double precision, intent(in) :: x(nnos), y(nnos)
940     Integer, intent(in) :: vi(nospe)
941     Integer, intent(in) :: n, nnos, nospe
942
943     ! Variaveis locais
944     Integer :: j, ierr
945     Double precision :: x_(1:3), y_(1:3)
946
947     x_(3)=x(vi(3))-x(vi(1))
948     x_(2)=x(vi(2))-x(vi(1))
949     y_(3)=y(vi(3))-y(vi(1))
950     y_(2)=y(vi(2))-y(vi(1))
951
952     F(1:n,1:n)=0.0D0
953
954     Do j=1,6
```



```
956     End do
957
958     Do j=1,3
959       F(6+j,6+j)=x_(2)
960       F(9+i,9+j)=y_(3)
961
962       F(6+j,9+j)=y_(2)
963       F(9+j,6+j)=x_(3)
964
965       F(12+i,12+j)=x_(2)**2
966       F(12+j,15+j)=2*x_(2)*y_(2)
967       F(12+i,18+j)=y_(2)**2
968
969       F(15+j,12+j)=x_(2)*x_(3)
970       F(15+i,15+j)=x_(2)*y_(3)+x_(3)*y_(2)
971       F(15+i,18+j)=y_(2)*y_(3)
972
973       F(18+i,12+j)=x_(3)**2
974       F(18+i,15+j)=2*x_(3)*y_(3)
975       F(18+i,18+j)=y_(3)**2
976
977     End do
978
979   End Subroutine Monta_matriz_F
980
981
982     !-----------Montagem da matriz de massa elementar  Mi-----!
983   Subroutine Monta_matriz_Mi (Ji,F,NC,P,C,Mi)
984
985     Implicit none
986     !Variaveis dummy
987     Double precision, intent (in) , dimension (21,21) :: F,P,C,NC
988     Double precision, intent (out), dimension (21,21) :: Mi
989     Double precision, intent (in) :: Ji
990
991     !Variaveis locais
992     Integer :: i,j,k
993     Double precision, dimension (21,21) :: Aux,Aux_simetrica
994
995     Aux=P ! Matriz auxiliar para que P não se modifique nesta função e possa
        ser usada por outras
996
997     Call Calcula_matriz_semelhante(C,Aux,21) ! Aux recebe C^T *Aux* C
998     Call Calcula_matriz_semelhante(F,Aux,21) ! Aux recebe F^T *Aux* F
999     Call Calcula_matriz_semelhante(NC,Aux,21) ! Aux recebe F^T *Aux* F
1000
1001
1002    ! Simetrizando a matriz Aux.
1003    !Isso é necessário pois o produto por F e por C não preservou a simetria
        de P
1004
1005    Aux_simetrica=0.5*(transpose(Aux)+Aux)
1006
1007    Mi=Ji*Aux_simetrica
1008
1009    ! Teste de simetria de Mi
1010    Do i=1,21
1011      Do j=1,21
1012
1013        if (Mi(i,j) .ne. Mi(j,i))       print*,i,j,"Matriz Mi nao é simetrica"
1014        !if (abs (Mi(i,j)-Mi(j,i)) >= 0.0000001 ) Print*,"Matriz Mi não é
            simetrica. Diferença no elemento",i,j
1015      End do
1016    End do
1017
```



```
1019
1020
1021    !-------Montagem da matriz de rigidez elementar Si--------!
1022    Subroutine Monta_matriz_Si (Ji,F,P,C,NC,Si,sigma, x,y,vi,nnos,nospe)
1023
1024     Implicit none
1025
1026     Double precision, intent(in), dimension (21,21) :: F,NC,C
1027     Double precision, intent(in), dimension (10,10) :: P
1028     Double precision, intent(out) :: Si(21,21)
1029     Double precision, intent(in) :: Ji, sigma
1030     Double precision, intent(in) :: x(nnos), y(nnos)
1031     Integer, intent(in) :: nnos, nospe, vi(nospe)
1032
1033     !Variaveis locais
1034     Integer :: i,j
1035     Double Precision, dimension (10,21) :: Dxx,Dyy,Dxy ! ! Matrizes do
        operador derivacão
1036     Double Precision, dimension (21,21) :: A,B,G,I_1,I_2,I_3,I_4,I_5,I_6!
        matrizes auxiliares
1037     Double precision :: a1,a2,a3,a4,a5,a6, f1, f2, f3
1038
1039     Double Precision, dimension (10,10) :: Aux
1040
1041
1042
1043     !Definição de Dxx, Dyy e Dxy, as matrizes de derivação
1044     Dxx=0.00
1045     Dyy=0.00
1046     Dxy=0.00
1047
1048     Dxx(1,4)=2.00
1049     Dxx(2,7)=6.00
1050     Dxx(4,11)=12.00
1051     Dxx(7,16)=20.00
1052
1053     Dxx(3,8)=Dxx(1,4)
1054     Dxx(6,13)=Dxx(1,4)
1055     Dxx(10,19)=Dxx(1,4)
1056     Dxx(5,12)=Dxx(2,7)
1057     Dxx(9,18)= Dxx(2,7)
1058     Dxx(8,17)=Dxx(4,11)
1059
1060     Dxy(1,5)=1.00
1061     Dxy(2,8)=2.00
1062     Dxy(4,12)=3.00
1063     Dxy(5,13)=4.00
1064     Dxy(8,18)=6.00
1065
1066     Dxy(3,9)=Dxy(2,8)
1067     Dxy(6,14)=Dxy(4,12)
1068     Dxy(10,20)=Dxy(5,13)
1069     Dxy(7,17)=Dxy(5,13)
1070     Dxy(9,19)=Dxy(8,18)
1071
1072     Dyy(1,6)=2.00
1073     Dyy(3,10)=6.00
1074     Dyy(6,15)=12.00
1075     Dyy(10,21)=20.00
1076
1077     Dyy(2,9)=Dyy(1,6)
1078     Dyy(4,13)=Dyy(1,6)
1079     Dyy(7,18)=Dyy(1,6)
1080     Dyy(5,14)=Dyy(3,10)
1081     Dyy(8,19)= Dyy(3,10)
```



```
1083
1084
1085    ! Matriz da forma quadratica da Integral do quadrado de wxx
1086    Aux=P    ! P é de dim 10
1087
1088    A=matmul(matmul(transpose(Dxx),Aux),Dxx) ! A tem dim 21
1089
1090    Call Calcula_matriz_semelhante(C,A,21)
1091    Call Calcula_matriz_semelhante(F,A,21)
1092    Call Calcula_matriz_semelhante(NC,A,21)
1093
1094    I_1=0.5D0*(transpose(A)+A)
1095
1096    ! Matriz da forma quadratica da Integral do quadrado de wyy
1097    Aux=P
1098
1099    A=matmul(matmul(transpose(Dyy),Aux),Dyy) ! A tem dim 21
1100    Call Calcula_matriz_semelhante(C,A,21)
1101    Call Calcula_matriz_semelhante(F,A,21)
1102    Call Calcula_matriz_semelhante(NC,A,21)
1103
1104    I_6=0.5D0*(transpose(A)+A)
1105
1106    ! Matriz da forma quadratica da Integral do quadrado de wxy
1107    Aux=P
1108
1109    A=matmul(matmul(transpose(Dxy),Aux),Dxy) ! A tem dim 21
1110    Call Calcula_matriz_semelhante(C,A,21)
1111    Call Calcula_matriz_semelhante(F,A,21)
1112    Call Calcula_matriz_semelhante(NC,A,21)
1113
1114    I_4=0.5D0*(transpose(A)+A)
1115
1116    ! Matriz da forma bilinear da Integral do quadrado de wyy*wxx
1117
1118    !Para I_4 ser simetrica, ela deve ser montada considerndo a soma
        (Dxx^T)*P*Dyy + (Dyy^T)*P*Dxx
1119
1120    Aux=P
1121
1122    A=matmul(matmul(transpose(Dxx),Aux),Dyy) ! A tem dim 21
1123    B=matmul(matmul(transpose(Dyy),Aux),Dxx) ! B tem dim 21
1124    G=0.5D0*(A+B)
1125
1126
1127    Call Calcula_matriz_semelhante(C,G,21)
1128    Call Calcula_matriz_semelhante(F,G,21)
1129    Call Calcula_matriz_semelhante(NC,G,21)
1130    I_3=0.5D0*(transpose(G)+G)
1131
1132    ! Matrizes I_5 e I_6 são novas, para o caso da aproximacão de quinto
        grau ser feita diretamente no triangulo padrão
1133    Aux=P
1134    A=matmul(matmul(transpose(Dxx),Aux),Dxy) ! A tem dim 21
1135    B=matmul(matmul(transpose(Dxy),Aux),Dxx) ! B tem dim 21
1136    G=0.5D0*(A+B)
1137    Call Calcula_matriz_semelhante(C,G,21)
1138    Call Calcula_matriz_semelhante(F,G,21)
1139    Call Calcula_matriz_semelhante(NC,G,21)
1140    I_2=0.5D0*(transpose(G)+G)
1141
1142    Aux=P
1143    A=matmul(matmul(transpose(Dyy),Aux),Dxy) ! A tem dim 21
1144    B=matmul(matmul(transpose(Dxy),Aux),Dyy) ! B tem dim 21
1145    G=0.5D0*(A+B)
```

```
1146
1147      Call Calcula matriz semelhante(C,G,21)
1148      Call Calcula matriz semelhante(F,G,21)
1149      Call Calcula_matriz_semelhante(NC,G,21)
1150      I 5=0.5D0*(transpose(G)+G)
1151
1152      ! Fim da montagem das matrizes auxiliares
1153
1154
1155      ! Definicão dos coeficientes para a montagem de Si
1156
1157      f1=(x(vi(3))-x(vi(1)))**2 + (y(vi(3)) -y(vi(1)))**2
1158      f2=(x(vi(2)) - x(vi(1)))*(x(vi(3))- x(vi(1))) +(y(vi(2)) - y(vi(1)))*(y(vi(3
          )) - y(vi(1)))
1159      f3=(x(vi(2))-x(vi(1)))**2 + (y(vi(2)) -y(vi(1)))**2
1160
1161      a1=(f1**2)/Ji**3
1162      a2=-4*f2*f1/Ji**3
1163      a3=2*(f2**2 + sigma*(Ji**2))/Ji**3
1164      a4=(4*f2**2+2*(1-sigma)*Ji**2)/Ji**3
1165      a5=-4*f2*f3/Ji**3
1166      a6=f3**2/Ji**3
1167
1168
1169      Si=a1*I 1+a2*I 2+a3*I 3+a4*I 4+a5*I 5+a6*I 6
1170
1171    ! Teste de simetria das matrizes de rigidez
1172     Do i=1,21
1173       Do j=1,21
1174         if (Si(i,j) .ne. Si(j,i))      print*,i,j,"Matriz Si nao é simetrica"
1175         !if (abs (Mi(i,j)-Mi(j,i)) >= 0.0000001 ) Print*,"Matriz Mi não é
              simetrica. Diferenca no elemento",i,j
1176       End do
1177     End do
1178
1179
1180    End Subroutine Monta_matriz_Si
1181
1182
1183    !----------------------Monta vetor de variaveis nodais
          ---------------------------!
1184
1185    !Essa subrotina monta um vetor ui, para cada triangulo i, análogo ao
          vetor vi que relaciona os
1186    !indices locais e globais dos nós; mas ui considera todas as variáveis
          incidentes em cada nó,
1187    !Dessa forma a dimensão de ui é 21, o numero de variáveis totais por
          elemento.
1188    !As variáves estão aqui ordenadas como nas matrizes elementares:
1189    !u1,u2,u3,ux1,ux2,ux3,uy1,uy2,uy3,...,uyy1,uyy2,uyy3, e as derivadas
          normais de u nos pontos medios das arestas.
1190
1191
1192    Subroutine Monta Var nodais (vi,ui,nvertices,nv vertice)
1193     Implicit none
1194
1195     Integer, intent(in) :: vi(6)
1196     Integer, intent(out) :: ui(21)
1197     Integer, intent(in) :: nvertices,nv_vertice !numero total de vertices da
          malha e numero de variáveis por vertice
1198
1199
1200     ! Var locais
1201     Integer:: l,i,k
1202
```



```
1205       if (k .le. 3*nv_vertice) then ! Então estamos sobre os vertices
1206         l=1+int((k-1)/3)
1207         i=1+mod((k-1),3)
1208         ui(k)=vi(j) + nvertices*(l-1)
1209       else                          ! Caso contrarios, estamos sobre arestas
1210         i=4+ mod(k-1,3)
1211         ui(k)=vi(j) + nvertices*(nv vertice-1)
1212       end if
1213
1214     End do
1215
1216    End Subroutine Monta Var nodais
1217
1218
1219    Subroutine Monta matrizes globais(S ,M ,Si,Mi,ui,dimensao)
1220
1221     Implicit none
1222
1223     Integer, intent (in) :: dimensao, ui(21)
1224     Double Precision, intent (in) :: Si(21,21),Mi(21,21)
1225     Double Precision, intent(inout) :: S (dimensao,dimensao),M (dimensao,
          dimensao)
1226
1227     ! Variáveis locais
1228     Integer ::l,k
1229
1230     Do l=1,21
1231       Do k=1,21
1232         !print*,"posicões nao nulas da matriz de massa:",vi(l),vi(k)
1233         S_(ui(k),ui(l))= S_(ui(k),ui(l)) + Si(k,l)
1234         M (ui(k),ui(l))= M (ui(k),ui(l)) + Mi(k,l)
1235       End Do
1236     End Do
1237
1238
1239    End Subroutine Monta matrizes globais
1240
1241
1242      !----------CONTABILIZAR NA MONTAGEM DAS MATRIZES, AS CONDICÕES DE
          CONTORNO-----------------------------------!
1243
1244    Subroutine Contabiliza condicoes de contorno (S ,M ,fix,nnos,
          contabilizou_cc,r,dimensao,nvertices,nv_fix,nm_fix)
1245
1246     Implicit none
1247
1248     Integer, intent(in) :: nnos, fix(nnos),r,dimensao,nvertices, nv fix,nm fix
1249     Integer, intent(out) :: contabilizou cc
1250     Double precision, intent(inout) :: M (dimensao,dimensao), S (dimensao,
          dimensao)
1251
1252     !Variáveis locais
1253     Integer:: i,j,k,n, ierr ! nv_fix: numero de vertices de triangulos q
          estao fixos e nm fix: numero de ptos medios q estao fixos na malha
1254     Integer, allocatable :: eliminar(:)
1255     Integer, allocatable :: varnod_eliminar(:,:) ! matriz que contém os
          indices das variaveis nodais que serao eliminadas
1256
1257     !-----------------------------------------Condição de Pontos fixos,
          do bordo ou outros ----------------------------------------!
1258
1259     ! Utilizando a info dos vértices que estão fixos na montagem das
          matrizes: As linhas e colunas relativas aos vértices que estão
1260     !fixos são deletadas, uma vez que o valor da funcão nesses pontos ja é
          conhecido, e não é necessário resolver o problema para esses pontos.
```



```
1262     !o valor de w e de suas primeiras derivadas: w,wx,wy,e dw/dn,
1263     !(este ultimo nos pontos medios das arestas)
1264     !Nesse caso, para cada vertice fixo, que for vertice de triangulo, temos
          que excluir as linhas da matriz que correspondem aos valores
1265     !de w,wx,wy nesse vertice, i.e., para cada vertice i que for vertice de
          triangulo, temos 3 linhas e colunas para excluir.
1266     !Para cada vertice que for pto medio de arestas, 1 linha e 1 coluna.
1267
1268     print*,"Contabilizando as condicoes de contorno, r=",r
1269     n=3*nv_fix+nm_fix
1270
1271     allocate (eliminar(1:n),varnod eliminar(4,nnos), stat = ierr)
1272     if (ierr/=0) then
1273       print*,"Erro de alocação do variavel eliminar e varnod_eliminar"
1274       stop
1275     end if
1276
1277    !montando o vetor eliminar, das linhas e colunas que serao eliminadas.
          esse vetor tem tamanho nv fix*3+nm fix
1278     k=1
1279     Do i=1,nvertices
1280       if (fix(i).eq. 1)  then  ! se o vertice i esta fixo,
1281         varnod_eliminar(1,k)=i
1282         Do j=0,2
1283           eliminar(k+nv fix*j)=nvertices*j+i   ! guardo em eliminar, o valor
              ##, nas posicões das linhas que serão eliminadas.
1284           varnod_eliminar(j+2,k)= nvertices*j+i
1285         End do
1286         k=k+1
1287       end if
1288     End do
1289
1290     k=1
1291     ! Laco no numero de nós que são pontos medios de triangulos. Verificacão
          daqueles que estão fixos
1292     Do i=nvertices+1,nnos
1293       If (fix(i).eq.1) then
1294         eliminar(k+3*nv fix)=i+5*nvertices !5=nv vertice-1 esse 3 sao de
              quantas variaveis por vertice iremos eliminar, no caso sao só 3, o
              valor de w de suas primeiras derivadas
1295
1296         varnod eliminar(1,k+nv fix)=i
1297         varnod_eliminar(2,k+nv_fix)=i+5*nvertices
1298         !print*,"posicao",k+3*nv fix,"do vetor eliminar
              recebe",nvertices*5+i
1299         k=k+1
1300
1301       End if
1302     End do
1303
1304     Do i=n,1,-1    ! laco decrescente no numero de variaveis, para o
          redimensionamento da matriz
1305       if (eliminar(i).ne.dimensao) then ! se eliminar(i) nao for a ultima
          linha/coluna então faremos o deslocamento das linhas e colunas abaixo
          e a direita de eliminar(i) até dimensao-1.
1306         !print*,"Vertice",eliminar(i),"esta fixo."    ! Deslocamos a matriz
              uma unidade para cima e para a esquerda, respect.
1307         Do i=eliminar(i),dimensao-1
1308           S_(j,:)=S_(j+1,:)
1309           S (:,i)=S (:,i+1)
1310           M (i,:)=M (i+1,:)
1311           M_(:,j)=M_(:,j+1)
1312         End do
1313       End if
1314     End do
```



```
1316     contabilizou_cc=1
1317
1318    End Subroutine Contabiliza condicoes de contorno
1319
1320   !---------Redimensiona  matrizes-----------------------------!
1321
1322    Subroutine Redimensiona_matrizes(dimensao,r,M,S,M_,S_)
1323     Implicit none
1324
1325     Integer, intent(in) :: dimensao,r
1326     Double precision, intent(in) :: M (dimensao,dimensao),S (dimensao,dimensao)
1327     Double precision, intent(out) :: M(r,r),S(r,r)
1328
1329     !Variaveis locais
1330     Integer :: i,j
1331
1332     print*,"Redimensionando as matrizes, de dim",dimensao,"para",r
1333     Do j=1,r
1334       Do i=1,r
1335         S(i,j)=S_(i,j)
1336         M(i,j)=M (i,j)
1337       End do
1338     End do
1339
1340
1341   ! Teste de simetria das matrizes de massa e rigidez
1342     Do i=1,r
1343       Do j=1,r
1344         if (S(i,j).ne.S(j,i)) Print*,"Matriz S não é simetrica. Erro no
              elemento S", i,j
1345         if (M(i,j).ne.M(i,j))  Print*,"Matriz M não é simetrica.",M(i,j),"e
              diferente de",M(j,i)
1346       End do
1347     End do
1348
1349
1350    End subroutine Redimensiona matrizes
1351
1352
1353      !-----------------------------------MAtrizes
          semelhantes----------------------------!
1354    Subroutine Calcula matriz semelhante(A,S,n)
1355      ! Transforma a Matriz S numa matriz semelhante a ela. S= A^T*S*A
1356
1357       Implicit none
1358       Integer :: n
1359       Double precision, dimension(n,n) :: A,S
1360
1361       ! Variáveis locais
1362       Double precision, dimension (n,n) :: M
1363
1364
1365       M=matmul(transpose(A),S)
1366       S=matmul(M,A)
1367
1368    End subroutine Calcula_matriz_semelhante
1369
1370
1371     !-------------- Subrotina GERAL para escrever uma matriz quadrada num
          arquivo-----------------------------------------!
1372    Subroutine Escreve Matriz em Arquivo (M,dimensao,unidade,nome arquivo,stat)
1373
1374     Implicit none
1375
1376     Double Precision :: M(dimensao,dimensao)
```

```
1379      Integer :: unidade,dimensao
1380
1381      ! Variaveis locais
1382      Integer:: i,j
1383
1384
1385      Open(unidade,file=nome arquivo,status=stat)
1386     ! Write(unidade,*),"Dimensao da matriz:",dimensao
1387      Write(unidade,*)
1388        Do i=1,dimensao
1389        !Write(unidade,*),"Coluna",i
1390          Do j=1,dimensao
1391            !Write(unidade,'(21E12.3)'),M(i,:)
1392            !Write(unidade,'(21f7.4)'),M(i,:)
1393            Write(unidade,*),M(i,j)
1394          End do
1395        End do
1396
1397        ! escrever em forma de matriz:
1398        Write(unidade,*)
1399        Write(unidade,*)
1400        Write(unidade,*)
1401        Write(unidade,*)
1402        Do i=1,dimensao
1403            Write(unidade,'(21f9.5)'),M(i,1:21)
1404            !Write(44,'(21f10.6)'),M(i,20:35)
1405        End do
1406
1407     Close(unidade)
1408
1409    End Subroutine Escreve matriz em arquivo
1410
1411
1412      !-----------------REGISTRO DAS MATRIZES DE RIGIDEZ E DE MASSA EM ARQUIVO
          .txt-------------------------------!
1413
1414    Subroutine Escreve_matrizes(r,M,S)
1415
1416      Implicit none
1417
1418      Integer:: r
1419      Double precision:: M(r,r), S(r,r)
1420
1421      !Variaveis locais
1422      Integer:: i,j
1423
1424
1425      Open(43, file=
          '/home/daniela/Documents/fortran/placa conforme nova/Entrada/Matriz rigid
          ez.txt', status='old')
1426      Open(44, file=
          '/home/daniela/Documents/fortran/placa conforme nova/Entrada/Matriz massa
          .txt', status='old')
1427      Write(43,*),"Dimensao da matriz:",r
1428      Write(44,*),"Dimensao da matriz:",r
1429      Do j=1,r
1430          ! Write(44,*)"Coluna",i
1431          Write(43,*)"Coluna",j
1432       Do i=1,r
1433          Write(43,*),S(i,j)
1434          Write(44,*),M(i,j)
1435          !Write(44,'(f23.12)'),M(i,j)
1436       End do
1437      End do
1438
```



```
1439      ! escrever em forma de matriz:
1440      Write(43,*)
1441      Write(43,*)
1442      Write(44,*)
1443      Write(44,*)
1444      Do i=11,21
1445          Write(43,'(21f10.6)'),S(i,:)
1446          Write(44,'(21f10.6)'),M(i,11:21)
1447       End do
1448
1449     End subroutine Escreve_matrizes
1450
1451
1452    !----------------IMPRESSÃO DOS AUTOVALORES E AUTOVETORES EM ARQUIVO
       .txt--------------------------------!
1453
1454   Subroutine Imprime_autovetores(r,w,S)
1455
1456    Implicit none
1457    Integer, intent(in) :: r
1458    Double precision, intent(in) :: w(r),S(r,r)
1459
1460    !Variaveis locais
1461    Integer :: i,j
1462
1463
       !Open(60,file='/home/daniela/Documents/fortran/placa_conforme_nova/Saida/A
       utovetores.txt', status='old')
1464    !Write(60,*),r
1465    !Write(*,*),"Primeiro autovalor:",w(1)
1466    !Write(60,*)
1467    !Do i=1,25
1468    !  Write(60,*) w(i)
1469    !End do
1470
1471    Open(70,file=
       '/home/daniela/Documents/fortran/placa_conforme_nova/Saida/Autovetores.txt
       ', status='old')
1472    Write(70,*),r
1473    Write(*,*),"Primeiro autovalor:",w(1)
1474    Write(70,*)
1475    Do i=1,25
1476      Write(70,*) w(i)
1477    End do
1478
1479     End subroutine Imprime_autovetores
1480
1481    !------------------------------ ARQUIVOS PARA O SCILAB (funcão
       fec)---------------------------------------!
1482    ! Só vale para o anzats linear. arrumar outra função do
       scilab-------------------------!
1483
1484   Subroutine Dados_plotagem(nedges,nnos,nele,x,y,S,T,r,fix)
1485
1486    Implicit none
1487    Integer, intent(in) :: nedges, nnos,r,nele, T(7,nele),fix(nnos)
1488    Double precision, intent(in) :: x(nnos), y(nnos), S(r,r)
1489
1490   ! Variáveis locais
1491    Integer :: i,j,k,l,p ! contadores
1492
1493    Open (50,file=
       '/home/daniela/Documents/fortran/placa conforme nova/Saida/coordxy02 scila
       b.txt', status='old')
1494    p=nnos-nedges                                  ! é o numero de
```



```
       vertices da malha, sem considerar os pontos medios dos triangulos
1495
1496    Do i=1,p
1497      Write(50,*) x(i), y(i)
1498    End do
1499
1500    Open (51,file=
       '/home/daniela/Documents/fortran/placa conforme nova/Saida/TO2 scilab.txt'
       , status='old')
1501    Do i=1,nele
1502      Write(51,*) T(1,i), T(2,i), T(3,i), T(4,i), T(5,i), T(6,i),T(7,i)
1503    End do
1504
1505    Open (52,file=
       '/home/daniela/Documents/fortran/placa conforme nova/Saida/autofunc02 scil
       ab.txt', status='old')
1506
1507    !l=1
1508    Do k=1,25 ! escreve 25 das r autofunções
1509      l=1
1510      Do i=1,p
1511        !se o vertice estava fixo, escreva 0.0, caso contrario, escreva o
          proximo elemento da matriz S
1512        if (fix(j).eq.1) then
1513          Write(52,*) 0.00
1514        else
1515          Write(52,*) S(l,k)
1516          l=l+1
1517        end if
1518      End do
1519    End do
1520
1521    Close(50)
1522    Close(51)
1523    Close(52)
1524
1525   End subroutine Dados_plotagem
1526
1527
```

B.3 Formulação de quinto grau - *confome 18*

```
1   PROGRAM Dani
2
3   !-------------------------------------------------------------------------------
    ------------------------------------------------------------!
4   !**.....................Elementos Finitos : Elemento triangular com
    aproximacão de quinto grau.....................................** !
5   !                              2ª variacão do elemento conforme de quinto
    grau:                                                        !
6   !                                                                              !
7   !    Neste caso, estamos impondo que o a derivada normal do polinomio de
    quinto grau (que a princípio tem grau 4) é um polinomio     !
8   ! cúbico. Dessa forma, as variáveis nodais dos vértices continuam
    determinando tal poliomio cubico, o que garante a continuidade    !
9   ! da derivada normal entre os elementos, caracterizando a conformidade
    desse elemento.                                              !
10  ! Ainda, tal restrição, faz com que a derivada normal seja derminada
    apenas pelas variáveis de vértices e portando,               !
11  ! ela deixa de ser uma variável nodal. Tal variacão do elemento de quinto
    grau passa agora a ter apenas 18 coeficientes,               !
12  ! ao invés de 21 e assim, esperamos que o efeito numérico causado por essa
    diminuicão e pela uniformidade com relacão à distribuicão !
13  ! das variáveis nodais sejam positivamente
    relevante.              !
14  !                                                                              !
15  !                                                                              !
16  !   O objetivo deste programa é resolver o problema de autovalor
    generalizado do tipo Ax=B(lambda)x resultante da             !
17  !  discretizacão (via elementos finitos) da integral variacional para o
    problema de autovalores do bilaplaciano,                     !
18  !  com condicão de contorno de Dirichlet e Newman homogênea. (Bordo
    fixo)                                                        !
19  !  O programa recebe informacões sobre a malha que é utilizada na
    aproximação/discretização e                                  !
20  !  monta as matrizes A e B, chamadas de matrizes de rigidez e massa,
    respectivamente.                                             !
21  !  Para a resolução efetiva do preblema de autovalor, utilizamos uma
    subrtotina do LAPACK: Linear Algebra Package.                !
22  !  Ao final, este programa prepara arquivos para a plotagem gráfica (via
    Scilab) das soluções aproximadas obtidas.                    !
23  !                                                                              !
24  !                                                                              !
25  !  *.............  Subrotinas externas utilizadas - LAPACK:
    ..............................*                                              !
26  !                                                                              !
27  !    - DSYGV(ITYPE, JOBZ, UPLO, N, A, LDA, B, LDB, W, WORK, lwork, info):
    dupla precisão. Resolve um problema                          !
28  !    de autovalor generalizado do tipo Ax=B(lambda)x, onde A, e B são
    matrizes simétricas e B é positiva definida                  !
29  !                                                                              !
30  !    - DGETFR() : para fatoracão
    PLU                                                                            !
31  !    - dgetri(): para a inversão de matrizes, a partir  da fatoracão LU
    obtida da funcão DGETFR                                      !
32  !    - dpotrf(): para a fatoracão de
    Cholesky                                                     !
```



```
33  !                                                                              !
34  !-------------------------------------------------------------------------------
    ------------------------------------------------------------!
35
36  !'/home/daniela/Documents/Malha/triangle O2/Qua.5.ele' lab
37  !'/home/daniela/Documentos/Malha/triangle O2/Quar.1.ele' casa
38  !'/home/posmap/daniela/Documentos/Malha/triangle_O2/Quar.1.ele' redeIME
39
40    Implicit none
41
42   !-------------- Variáveis globais da
    modelagem------------------------------!
43
44    Integer :: nele,nnos,nospe,nedges    ! Parâmetros da malha:  #de
    elementos triangulares, #nós, #nós por elemento, #arestas,
    respectivamente.
45                                         !São lidos nos arquivos externos
                                           .ele, .node e .edge
46
47    Integer :: nv vertice,nvertices      ! #variáveis nodais por vértice de
    triângulo, #nós da malha que são vértices de triângulos.
48
49    Integer :: dimensao           ! dimensao inicial das matrizes de massa e
    rigidez globais
50
51    Integer, allocatable, dimension (:,:) :: T, edge   !matriz que contém
    parâmetros da malha: #de cada elemento triangular e seus
                                              !respectivos vertices segundo
                                              a numeração global da malha
53
54    Integer, allocatable, dimension (:) :: fix, bordo   ! vetor que armazena
    a informacão sobre o nós que ficarão fixos e sobre
55                                         ! os nós do bordo (marca 1 se o nó
                                           está fixo ou é de bordo e marca 0
                                           caso contrario), respectivamente.
56
57    Integer :: r                         ! dimensão das matrizes de massa e
    rigidez depois das condicões de contornono serem
    contabilizadas.
58
59    Integer :: bordo fixo=1      !Condicão de contorno: bordo fixo recebe os
    valores 1, se o bordo for fixo ou 0, caso contário.
60
61    Integer :: nv fix          !nv fix é a quantidade de vertices fixos,
    definido pela condição de contorno
62
63
64    Double precision, allocatable, dimension (:) :: x,y ! Vetores que
    armazenam as coordenadas (x,y) de cada nó: (x(i), y(i))
65                                                       ! são as coordenadas
                                                         do i-ésimo nó.
66
67    Double Precision, allocatable, dimension (:,:) :: S, M   ! Vetores que
    representam as Matrizes de Rigidez e de Massa globais, do problema de
    autovalor.
68
69    Double precision, allocatable, dimension (:,:) :: normal ! vetor que
    guarda a lista global de arestas e de vetores
    normais
70
71    Integer:: ierr    ! controle da alocacão dinâmica de memória
72    !Integer:: i,j,k           ! Contadores
73    Integer:: contabilizou cc ! Indica se a condicão de contorno foi
    realizada (assume valor 1 se sim e  0 se não)
```



```
78      Double precision, allocatable, dimension(:) :: w, work  !w é um vetor
        que recebe os r autovalores
79      Character :: jobz='V', uplo='U'
80      Integer :: info, itype=1, lda, ldb, lwork
81
82
83
     !---------------------------Início-----------------------------------------
     ------------------------------!
84
85      nv vertice = 6 ! # variaveis por vertice
86      nospe=3 ! # nós por elemento.
87      !Vamos ler apenas as informacões dos 3 vértices de cada elemento
88
89    !-- Leitura de arquivos .edge, .node e .ele: Fornecem os parâmetros da
      malha que serão usados para a alocacão da memória para alumas variáveis
      ----!
90
91      !Leitura do arquivo .ele gerado pelo Triangle. Malha com ordem 2
92      Open(40,file='/home/daniela/Documents/Malha/triangle_O2/quadrado.5.ele',
        status='old')
93      Read(40,*) nele
94
95      !Leitura do arquivo .node gerado pelo Triangle
96      Open (41, file =
        '/home/daniela/Documents/Malha/triangle O2/quadrado.5.node', status='old')
97      Read (41,*) nnos
98
99      !Leitura do arquivo .edge gerado pelo Triangle - malha de ordem 2 !
100     Open (42, file=
        '/home/daniela/Documents/Malha/triangle O2/quadrado.5.edge', status='old')
101     Read(42,*) nedges
102
103
     !-------------------------------------------------------------------------------
     ----------------------------------------!
104
105     nvertices= nnos-nedges
106     dimensao=6*nvertices
107
108   !--------------- Alocando as variaveis
      alocáveis-------------------------------!
109     allocate (T(1:4,1:nele), stat = ierr)
110     if (ierr/=0) then
111       print*,"Erro de alocacão do variavel T"
112       stop
113     end if
114
115     allocate (bordo(1:nvertices),fix(1:nvertices), stat = ierr)
116     if (ierr/=0) then
117       print*,"Erro de alocacão do variavel bordo"
118       stop
119     end if
120
121     allocate (x(1:nvertices), y(1:nvertices), stat = ierr)
122     if (ierr/=0) then
123       print*,"Erro de alocacão das variaveis x e y"
124       stop
125     end if
126
127      allocate (edge(3,nedges), normal (2,nedges), stat = ierr)
128      if (ierr/=0) then
129        print*,"Erro de alocacão do variavel edge e normal"
130        stop
131     end if
```



```
     ----------------------------------------------!
134
135     print*,"nvertices,nele,nedges",nvertices,nele,nedges
136
137     Call Le arquivos(nele, nvertices, nedges, x, y, bordo, T, edge)
138
139     Call Define_condicoes_contorno(nvertices,fix,bordo,dimensao, bordo_fixo,r,
        nv fix) ! Varredura na malha, só diz respeito aos nós, e não às variáveis
140
141     !A partir daqui, o valor de r é conhecido e posso alocar as matrizes S e M
142     allocate (M(1:r,1:r), S(1:r,1:r), stat = ierr)
143     if (ierr/=0) then
144       print*,"Erro de alocacão das variaveis S e M"
145       stop
146     end if
147
148     contabilizou_cc = 0
149
150     Call Define vetores normais(nedges,nnos,edge,normal, x,y)
151
152     ! ----------FUNÇÃO PRINCIPAL: MONTAGEM DAS MATRIZES DE MASSA E RIGIDEZ
        PARA O PROBLEMA DE AUTOVALOR----------------!
153     !-----Monta matrizes de rigidez e massa, locais e globais, bem como
        todas as matrizes auxiliares necessárias-------!
154
155     print*,"Entrando em monta matriz"
156     Call Monta_matrizes (nele, nospe,x, y, fix, T, S, M,r,contabilizou_cc,
        nv vertice, dimensao,nvertices,nv fix, nedges,edge,normal)
157
158     print*,"Saí de monta matriz"
159
160     !---- Registro em arquivo das matrizes de Rigidez e de Massa que serão
        usadas em dsygv ------------------------------!
161
162     !Open(43,
        file='/home/daniela/Documentos/fortran/placa_conforme_18_coef/Matriz_rigid
        ez.txt', status='old')
163     !Open(44,
        file='/home/daniela/Documentos/fortran/placa conforme 18 coef/Matriz massa
        .txt', status='old')
164
165     !Call Escreve Matrizes(r,M,S)
166
167
     !-------------------------------------------------------------------------------
     ----------------------------------------------!
168
169     print*,"Dimensão das matrizes em dsygv:",r
170     lda=r
171     ldb=r
172     lwork=max(1,3*r-1)
173
174     print*,"lda,ldb,lwork",lda,ldb,lwork
175
176     allocate (w(1:r), work(1:lwork), stat = ierr)
177      if (ierr/=0) then
178       print*,"Erro de alocacão das variáveis w e work"
179       stop
180     end if
181
182   !------------------ Resolve o problema generalizado de autovalores do tipo
    1: itype=1, computa os autovetores e autovetores jobz=V ----!
183     print*,"entrei em DSYGV"
184
185     Call dsygv (itype, jobz, uplo, r, S, lda, M, ldb, w, work, lwork, info)
```

```
186
187     Print*, "info=",info    ! Controle da função dsyqv: info=0 indica
        sucesso nos calculos
188     print*,"primeiro autovalor", w(1)
189     if (info.ne.0) then
190       print*,"Problema de autovalor não resolvido"
191       stop
192     end if
193
194     ! Impressão dos autovalores e autovetores em arquivo
195     Call Imprime_autovetores(r,w,S)
196
197     print*,"entrei em dados plotagem"
198
199     ! Prepara arquivos para plotar gráficos no scilab
200     Call Dados_plotagem(nvertices,nele,x,y,S,T,r,fix)
201
202     print*,"Enfim, acabou!"
203
204     END PROGRAM Dani
205
206     !---------------------------Subrotinas
        Utilizadas-----------------------------------------------!
207
208     !---------------------------------LEITURA DE
        ARQUIVOS-------------------------------------------------!
209
210   Subroutine Le_arquivos (nele, nvertices, nedges, x, y, bordo, T, edge)
211
212     Implicit none
213     Integer, intent(in) :: nele, nvertices, nedges
214     Integer, intent(out) :: T(1:4,1:nele)  ! T é matriz (nospe+1)x(nele) que
        na coluna j guarda o numero
215                                            !do triangulo i, e seus vértices
                                               (i 1, i 2, i 3)
216                                            !nas linhas sequintes, segundo a
                                               numeração global da malha
217
218     Integer, intent(out) :: bordo(1:nvertices), edge(1:3,1:nedges)    !As
        posições desse vetor de inteiros recebem valores 0 ou 1, conforme o
        vértice for ou não for de fronteira, respectivamente
219
220     Double precision, intent(out) :: x(1:nvertices), y(1:nvertices)
221     !Integer, intent(out) :: edge(1:3,1:nedges)      ! matriz que guarda o
        numero da aresta da malha e os pontos final e inicial dessas,
222                                                     !nessa ordem, nas linhas
                                                        1,2 e 3 respectivamente
223   ! Variáveis locais
224     Integer:: i
225     Double precision :: c,b
226
227   ! Leitura do arquivo .ele gerado pelo Triangle. Malha com ordem 2
228     Do j=1,nele
229       Read(40,*) T(1,j),T(2,i),T(3,j),T(4,j)
230     End do
231     Close(40)
232
233     !Leitura do arquivo .node gerado pelo Triangle
234     Do j=1, nvertices
235       Read (41,*) c, x(j), y(j),b, bordo(j)  ! Nas malhas triangular
          tri isosceles e retangulos, temos so quatro parametros
236     End Do
237     Close (41)
238
239     !Leitura do arquivo .edge gerado pelo Triangle
```



```
242     End do
243     Close(42)
244
245   End subroutine Le_arquivos
246
247   !--------------------------------DEFINIÇÃO DOS VETORES
      NORMAIS-----------------------------------------------!
248
249   Subroutine  Define vetores normais(nedges,nnos,edge,normal, x,y)
250
251    Implicit none
252    Integer, intent(in) :: nedges, nnos
253    Double precision, intent(in) :: x(1:nnos),y(1:nnos)
254    Integer, intent(in):: edge(1:3,1:nedges)
255
256    Double precision, intent(out) :: normal(1:2,1:nedges)  ! matriz de vetores
       normais.
257                                                 !Cada coluna i tem as
                                                    corrdenadas do i-ésimo vetor
                                                    normal: (normal(1,i),normal(2,i))
258
259    !Variáveis locais
260
261    Integer :: i, ierr
262    Double precision, allocatable, dimension(:) :: norma
263    Double precision, allocatable, dimension(:,:) :: normal unitaria
264
265
266    allocate(normal unitaria(1:2,1:nedges), norma(1:nedges), stat=ierr)
267    if (ierr/=0) then
268      print*, "Erro na alocacao da variavel normal unitaria e norma"
269      stop
270    end if
271
272    !---Definindo as coordenadas x e y do i-ésimo vetor normal:
       [normal(1,i);normal(2,i)]-----!
273    Do i=1,nedges
274      !A aresta i tem pto inicial edge(2,i) e pto final edge(3,i)
275      !As coordenadas desses pontos são [x(edge(2,i)); y(edge(2,i))] e
         [x(edge(2,i)) ; y(edge(2,i))]
276      !As coordenadas dos vetores normais sao definidas pela rotação da
         aresta i, por um angulo de 90 graus no sentido anti-horário
277
278      if(edge(2,i) .LT. edge(3,i)) then
279        normal(1,i)=y(edge(2,i))-y(edge(3,i))
280        normal(2,i)=x(edge(3,i))-x(edge(2,i))
281      end if
282
283      if (edge(2,i) .GT. edge(3,i)) then
284        normal(1,i)=(y(edge(3,i))-y(edge(2,i)))
285        normal(2,i)=x(edge(2,i))-x(edge(3,i))
286      end if
287    end do
288
289   !------------- Tornando a normal unitária---------!
290    Do i=1,nedges
291      norma(i) = sqrt(normal(1,i)**2 + normal(2,i)**2)
292
293      normal_unitaria(1,i)=normal(1,i)/norma(i)
294      normal unitaria(2,i)=normal(2,i)/norma(i)
295
296      normal(1,i) = normal unitaria(1,i)
297      normal(2,i) = normal unitaria(2,i)
298    End do
299
```



```
301
302
303   !-----------------------------DEFINIÇÃO DAS CONDIÇÕES DE
      CONTORNO-----------------------------------------!
304
305    !Esta subrotina define as condições de contorno que serao utilizadas.
306    !Por enquanto, define quais nós da malha permanecerao fixos e guarda a
       informação no vetor fix.
307    !Também conta o numero de vertices fixo, e guarda em nv fix
308
309   Subroutine Define_condicoes contorno(nvertices,fix,bordo,dimensao,
      bordo fixo,r,nv fix)
310
311    Implicit none
312
313    Integer, intent(in) :: nvertices, bordo(nvertices),bordo fixo, dimensao
314    Integer, intent(out) :: r, nv_fix, fix(nvertices)
315
316    !Variávies locais
317    Integer :: i
318
319    if (bordo fixo.eq.1) then
320      print*,"BORDO FIXO!"
321      fix(1:nvertices)=bordo(1:nvertices)
322    end if
323
324    nv_fix=0
325
326    !Laco no numero de vertices de triangulos. Verificacão da quantidade dos
       que estão fixos.
327    Do i=1,nvertices
328      if ((fix(i).eq. 1) ) then  ! se o vertice i esta fixo,
329        nv_fix=nv_fix+1
330      end if
331    End do
332
333    print*,"nv fix,nvertices,dimensao",nv fix,nvertices,dimensao
334    r=dimensao-3*nv fix   ! dimensao das matrizes S e M globais, após
       contabilizar as condições de contorno.
335
336   End Subroutine Define condicoes contorno
337
338   !-------------------------------------------------------------------------
      ---------------------------------!
339
340   !-------------------------MONTAGEM DAS MATRIZES DE MASSA E RIGIDEZ
      GLOBAIS-----------------------------------!
341
342   Subroutine Monta_matrizes(nele,nospe,x,y,fix,T,S,M,r,contabilizou_cc,
      nv_vertice,dimensao,nvertices,nv fix,nedges,edge,normal)
343    Implicit none
344
345    Integer, intent(in) :: nele, nospe, nv fix , nedges
346    Integer, intent(in) :: T(1:4,1:nele), fix(1:nvertices) , edge(3,nedges)
347    Double precision, intent(in) ::  x(1:nvertices), y(1:nvertices),normal(2,
       nedges)
348    Double precision, intent(out) :: S(1:r,1:r), M(1:r,1:r)
349    Integer, intent(out) :: contabilizou cc
350    Integer, intent(in) :: nv_vertice, nvertices, dimensao,r   ! dimensao é a
       dimensao das matrizes de massa de e de rigidez, antes da contabilizacão
       das condições de contorno
351
352
353    !---Variáveis locais-----!
354
```



```
       e massa elementares
357    Double Precision, Dimension(1:21,1:21) :: P   ! Matrizes auxiliares na
       montagem de Mi.
358    Double Precision, Dimension(1:18,1:18) :: F
359    Double Precision, Dimension(1:21,1:18) :: C
360    Double precision, dimension(1:10,1:10) :: P1 ! Matrizes auxiliares na
       montagem de Si.
361    Double precision, allocatable, dimension (:,:) :: S , M  ! matrizes
       auxiliares para a montagem de M e S (globais)
362    Integer :: pot(1:21,1:2)  ! Matriz que vai guardar as potencias de x e y
       para uso na montagem das matrizes P,P1, E e E1.
363                              ! A potencia do x está na coluna 1 e a do y, na
                                 coluna 2
364
365    Integer, allocatable, dimension(:) :: vi  ! Vetor auxiliar que guarda os
       vertices do iésimo triangulo segundo a numeração global da malha.
366    Integer :: n_i(3), ui(1:18)     ! Vetor auxiliar que guarda as posições
       globais de cada variável nodal do i-esimo triangulo.
367                              ! Será usado na montagem das matrizes globais
368
369    Integer :: i,j,k, ierr ! contadores, controle de alocação de memória
370
371    Double precision, parameter :: sigma = 0.3D0
372
373    allocate(S (1:dimensao,1:dimensao), M (1:dimensao,1:dimensao), vi(1:nospe),
       stat=ierr)
374    if(ierr/=0) then
375      print*, "Erro na alocação das variáveis M ,S , e vi"
376      stop
377    end if
378
379    print*,"Dimensao inicial da matriz global:",dimensao
380    !Open(102,
       file='/home/daniela/Documentos/fortran/placa conforme grau5/Matriz massa el
       ementar.txt', status='old')
381    !Open(103,
       file='/home/daniela/Documentos/fortran/placa conforme grau5/Espectro.txt',
       status='old')
382
383
384    !Inicializacão de S  e M  com zero
385    S (1:dimensao, 1:dimensao)= 0.0D0
386    M_(1:dimensao, 1:dimensao)= 0.0D0
387
388    !Montagem da matriz pot (potências de x e y)
389    k=1
390    Do i=0,5   ! 5 é  grau da aproximação
391       Do i=0,i
392         !print*,"k", k
393        pot(k,1)=i-i
394        pot(k,2)=i
395        k=k+1
396      End do
397    End do
398
399    !Montagem da Matriz das integrais P e P1. Essas matrizes sao constantes,
       não depende da geometria da malha.
400    Call Monta_matriz_P (P1,pot,10) ! para ser usada em Si
401    Call Monta_matriz_P (P,pot,21)  ! para ser usada em Mi
402    !Call Escreve Matriz em Arquivo (P,21,60,"Matriz P.txt","old")
403
404
405
406    !------------------------LACO NO NúMERO DE TRIANGULOS
       ---------------------------------!
```

```
411       Do j=1,nospe
412         vi(j)=T(i+1,i)  !Os vertices do triangulo i são: T(2,i), T(3,i), T(4,i)
413       End Do
414
415       Ji=((x(vi(2)))-x(vi(1)))*(y(vi(3))-y(vi(1))) - (x(vi(3))-x(vi(1)))*(y(vi(2
          ))-y(vi(1)))  ! Determinante da matriz de mudanca de coordenadas
416
417       !--------Definição do vetor n_i que guarda o indice global do vetor
          normal, nas posições locais 1,2 ou 3:
418       !Vamos estabelecer uma numeracão local para os vetores normais globais.
          cada elemento triangular i tem 3 vetores normais: n_i(j), j=1,2,3
419       j=1
420       Do while (j.LE.nedges)
421         if( ((vi(1) .eq. edge(2,j)) .and. (vi(2) .eq. edge(3,j))) .or. ((vi(2)
          .eq. edge(2,j)) .and. (vi(1) .eq. edge(3,j)))) then
422           n_i(3)=j      !n_i(3) é vetor oposto ao vertice 3 e é o j-ésimo
              vetor normal da tabela normal
423           j=nedges+1
424         else
425           j=j+1
426         end if
427       End do
428
429       j=1
430       Do while (j.LE.nedges)
431         if( ((vi(2) .eq. edge(2,j)) .and. (vi(3) .eq. edge(3,j))) .or. ((vi(3)
          .eq. edge(2,j)) .and. (vi(2) .eq. edge(3,j)))) then
432           n_i(1)=j      !n_i(1) é é vetor oposto ao vertice 1 e é o j-ésimo
              vetor normal da tabela normal (tal tabela segue a indexacão da
              lista de arestas)
433           j=nedges+1
434         else
435           j=j+1
436         end if
437       End do
438
439       j=1
440       Do while (j.LE.nedges)
441         if( ((vi(1) .eq. edge(2,j)) .and. (vi(3) .eq. edge(3,j))) .or. ((vi(3)
          .eq. edge(2,j)) .and. (vi(1) .eq. edge(3,j)))) then
442           n_i(2)=j !n_i(2) é é vetor oposto ao vertice 2 e é o j-ésimo vetor
              normal da tabela normal
443           j=nedges+1
444         else
445           j=j+1
446         end if
447       End do
448
449       ! O triangulo i,tem vetores normais, n_i(j), j=1,2,3 e estes tem
          coordenadas a(j),b(j), j=1,2,3
450       Do j=1,3
451         a(j)=normal(1,n_i(j))
452         b(j)=normal(2,n_i(j))
453       End do
454
455       Call Monta_Var_nodais (vi,ui,nvertices,nv_vertice) ! Monta Vetor que
          guarda o indice global de cada variavel nodal do triangulo Ti, em suas
          posicões 1,2,...,21
456
457       Call Monta matriz C (C,Ji,a,b,nvertices,x,y,vi,nospe,i)  ! Montagem da
          matriz C, que relaciona as variáveis nodais (ũ1,ũ2,...,ũ18) com os
          parâmetros (a1, a2,..., a21) da aproximacão. a=C*u
458
459       Call Monta matriz F (F,18,nvertices,nospe,vi,x,y)  ! Matriz que
          relaciona o vetores de variáveis nodais u e ũ.
```



```
462
463         Call Monta matriz Si (Ji,P1,C,F,Si,sigma,x,y,vi,nvertices,nospe) ! Matriz
             de rigidez elementar
464
465       !----Montagem das matrizes de rigidez e massa globais S  e M  a partir
          de Si e Mi: ----------!
466         Call Monta matrizes globais(S ,M ,Si,Mi,ui,dimensao)
467
468       End Do  !---------------------- FIM DO LACO (em i) NO NUMERO DE
          TRIANGULOS-----------------------------!
469
470       !close(102) ! Fechando o arquivo de testes
471
472       Call Contabiliza_condicoes_de_contorno (S_,M_,fix,nv_fix,r,dimensao,
          nvertices,contabilizou cc)
473
474       Call Redimensiona_matrizes(dimensao,r,M,S,M_,S_) ! S_ e M_ são passadas
          para as matrizes S e M, de dimensão r
475
476
477     End Subroutine Monta matrizes
478
479     !---------------------------------Subrotinas usadas em Monta Matriz
          ------------------------------------!
480
481     Subroutine Monta matriz C (C,Ji,a,b,nvertices,x,y,vi,nospe,k)
482
483       ! Essa é a matriz que relaciona coeficientes da aproximação polinomial
          com as variáveis nodais.
484       ! Nesse programa, C é matriz 18x21, pois temos polinomio de grau 5, com
          21 coeficientes, e 18 variáveis nodais que o determinarão.
485       ! As 18 variáveis nodais são suficientes pois esses polinomios são tais
          que sua derivada normal nas arestas do triangulo padrão
486       !são polinimios cúbicos. Fizemos essa imposição no triangulo padrão, onde
          fazemos a aproximacão polinomial.
487       !dessa forma, C passa a ser constante em relação aos parâmetros da malha.
488       !
489
490       Implicit none
491       Integer, intent(in) :: k,nospe,nvertices,vi(nospe)
492       Double precision, intent(in) :: Ji, a(3),b(3),x(nvertices),y(nvertices)
493       Double precision, intent(out) :: C(1:21,1:18)
494
495       !Variaveis locais
496       Integer:: i
497       Double precision, dimension (1:21,1:21) :: M_chapeu, inv_M_chapeu
498       Double precision, dimension (1:21,1:18) :: inclusao ! Representa a
          aplicacão linear inclusão de R^18 em R^18 x R^21
499       Double precision :: a (3), b (3) !imagens das coordenadas do vetor normal
          (a,b) pela transformação DL
500       Double precision :: x (3), y (3)
501
502
503       x (3)=x(vi(3))-x(vi(1))
504       x (2)=x(vi(2))-x(vi(1))
505       y_(3)=y(vi(3))-y(vi(1))
506       y (2)=y(vi(2))-y(vi(1))
507
508       ! Coordenadas das imagens dos vetores normais no plano \xi \eta
509       Do i=1,3
510         a (i)=(y (3)*a(i)-y (2)*b(i))/Ji
511         b (i)=(-x (3)*a(i)+x (2)*b(i))/Ji
512       End do
513
514       !Open(104,
```



```
          file='/home/daniela/Documentos/fortran/placa_conforme_18coef/Entrada/M_chap
          eu.txt', status='old')
515
516       !Do j=1,18
517       !  Read(104,*) M chapeu(i,:)
518       !End do
519
520       ! Montagem da matriz M chapeu, que deverá ser invertida em seguida.
521       M chapeu(1:21,1:21)=0.0D0
522
523       Do j=1,3
524         M chapeu(j,1)= 1.0D0
525         M chapeu(j+3,2)= 1.0D0
526         M_chapeu(j+6,3)= 1.0D0
527         M chapeu(j+9,4)= 2.0D0
528         M chapeu(j+12,5)= 1.0D0
529         M_chapeu(j+15,6)= 2.0D0
530     End do
531
532     M_chapeu(2,2)= 1.0D0
533     M chapeu(2,4)= 1.0D0
534     M chapeu(2,7)= 1.0D0
535     M_chapeu(2,11)= 1.0D0
536     M chapeu(2,16)= 1.0D0
537
538     M chapeu(3,3)= 1.0D0
539     M_chapeu(3,6)= 1.0D0
540     M chapeu(3,10)= 1.0D0
541     M chapeu(3,15)= 1.0D0
542     M_chapeu(3,21)= 1.0D0
543
544     M chapeu(5,4)= 2.0D0
545     M_chapeu(5,7)= 3.0D0
546     M chapeu(5,11)= 4.0D0
547     M chapeu(5,16)= 5.0D0
548
549     M chapeu(6,5)= 1.0D0
550     M chapeu(6,9)= 1.0D0
551     M chapeu(6,14)= 1.0D0
552     M chapeu(6,20)= 1.0D0
553
554     M chapeu(8,5)= 1.0D0
555     M_chapeu(8,8)= 1.0D0
556     M chapeu(8,12)= 1.0D0
557     M chapeu(8,17)= 1.0D0
558
559     M chapeu(9,6)= 2.0D0
560     M chapeu(9,10)= 3.0D0
561     M_chapeu(9,15)= 4.0D0
562     M chapeu(9,21)= 5.0D0
563
564     M_chapeu(11,7)= 6.0D0
565     M chapeu(11,11)= 12.0D0
566     M chapeu(11,16)= 20.0D0
567
568     M_chapeu(12,8)= 2.0D0
569     M chapeu(12,13)= 2.0D0
570     M chapeu(12,19)= 2.0D0
571
572     M chapeu(14,8)= 2.0D0
573     M chapeu(14,12)= 3.0D0
574     M_chapeu(14,17)= 4.0D0
575
576     M chapeu(15,9)= 2.0D0
577     M_chapeu(15,14)= 3.0D0
```



```
578     M_chapeu(15,20)= 4.0D0
579
580     M chapeu(17,9)= 2.0D0
581     M_chapeu(17,13)= 2.0D0
582     M chapeu(17,18)= 2.0D0
583
584     M_chapeu(18,10)= 6.0D0
585     M chapeu(18,15)= 12.0D0
586     M chapeu(18,21)= 20.0D0
587
588     !Parte 3x21 não constante
589
590     M chapeu(19,16)=5.0D0*a (3)
591     M_chapeu(19,17)=b_(3)
592
593     M chapeu(20,20)=a (2)
594     M_chapeu(20,21)=5.0D0*b_(2)
595
596     M chapeu(21,16)=5.0D0*a (1)
597     M_chapeu(21,17)= -4.0D0*a_(1)+b_(1)
598     M chapeu(21,18)=3.0D0*a (1)-2.0D0*b (1)
599     M chapeu(21,19)=-2.0D0*a (1)+3.0D0*a (1)
600     M_chapeu(21,20)=a_(1)-4.0D0*b_(1)
601     M chapeu(21,21)=5.0D0*b (1)
602
603     Call Inverte matriz (M chapeu,inv M chapeu,21,k)
604
605     !Montagem da matriz inclusão
606     inclusao(1:21,1:18)=0.0D0
607
608     Do i=1,18
609       inclusao(i,i)=1.0D0
610     End do
611
612     C=matmul(inv M chapeu,inclusao)
613
614   ! Close(104)
615
616   End Subroutine Monta matriz C
617
618   Subroutine Monta matriz F(F,n,nvertices,nospe,vi,x,y)
619
620     !F é matriz 18x18 que relaciona as variaveis nodais do triangulo padrão
        ũ com aquelas do triangulo arbitrário u
621     !ũ=Fu
622
623     Implicit none
624     Double precision, intent(out) :: F(1:n,1:n)
625     Double precision, intent(in) :: x(1:nvertices), y(1:nvertices)
626     Integer, intent(in) :: vi(1:nospe)
627     Integer, intent(in) :: n, nvertices, nospe
628
629     ! Variaveis locais
630     Integer :: j
631     Double precision :: x (1:3), y (1:3)
632
633     x (3)=x(vi(3))-x(vi(1))
634     x (2)=x(vi(2))-x(vi(1))
635     y_(3)=y(vi(3))-y(vi(1))
636     y (2)=y(vi(2))-y(vi(1))
637
638     F(1:n,1:n)=0.0D0
639
640     Do j=1,3
641       F(j,j)=1.0D0
```

```
642       End do
643
644       Do j=1,3
645         F(3+j,3+j)=x_(2)
646         F(6+i,6+i)=y (3)
647
648         F(3+j,6+j)=y (2)
649         F(6+j,3+j)=x_(3)
650
651         F(9+j,9+j)=x_(2)**2
652         F(9+j,12+j)=2*x_(2)*y_(2)
653         F(9+i,15+i)=y (2)**2
654
655         F(12+j,9+j)=x_(2)*x_(3)
656         F(12+i,12+i)=x (2)*y (3)+x (3)*y (2)
657         F(12+i,15+i)=y (2)*y (3)
658
659         F(15+i,9+i)=x_(3)**2
660         F(15+i,12+j)=2*x (3)*y (3)
661         F(15+i,15+i)=y (3)**2
662       End do
663
664     End Subroutine Monta matriz F
665
666
667      !-------Montagem da matriz das integrais
668     Subroutine Monta_matriz_P(P,pot,n)
669
670      Implicit none
671      Double precision, intent(out) :: P(1:n,1:n)
672      Integer, intent(in) :: pot (1:21,1:2)
673
674      ! Variaveis locais
675      Integer :: n,k,i,j
676      Double precision :: numerador,denominador ! var auxiliares, pois Fat é
         fc inteira, mas o quociente tava deve ser um numero real
677
678      !n é a dimensao de P, e depende do grau do polinomio que sera integrado.
679      !n=21 se grau_aprox =5
680      !n=10 se grau_aprox =3 . Essas sao as duas possibilidades nessa subrotina
681
682      Do k=1,n
683        Do i=1,n
684
685           numerador=((Fat(pot(k,1)+pot(i,1)))*(Fat(pot(k,2)+pot(i,2))))
686           denominador=(Fat(pot(k,1)+pot(i,1)+pot(k,2)+pot(i,2)+2))
687           P(k,i)=numerador/denominador
688
689        End do
690      End do
691
692     ! Teste de simetria
693      if (n==21) then
694        Do i=1,21
695          Do j=1,21
696            if (P(i,j) .ne. P(j,i))       print*,i,j,"Matriz P nao é simetrica"
697          End do
698        End do
699      End if
700
701      if (n==10) then
702        Do i=1,10
703          Do j=1,10
704              if (P(i,j) .ne. P(i,j))       print*,i,j,"Matriz P nao é simetrica"
705         End do
```



```
706       End do
707      end if
708
709     CONTAINS
710     Integer Function Fat(n)  !Função que calcula o fatorial de um numero
        inteiro >= 0
711
712      Implicit none
713      Integer, intent (in) :: n
714      Integer :: produto, i
715
716      produto=1
717      Do i=1,n
718        produto=produto*i
719      End do
720      Fat=produto
721
722     End Function Fat
723
724     End Subroutine Monta matriz P
725
726      !---Montagem da matriz de massa elementar  Mi-----!
727     Subroutine Monta matriz Mi (i,Ji,P,C,F,Mi)
728
729      Implicit none
730      Double precision, intent(in) :: P(1:21,1:21),C(1:21,1:18), F(1:18,1:18)
731      Double precision, intent(out), dimension(1:18,1:18) :: Mi
732      Double precision, intent(in) :: Ji
733      Integer, intent(in) :: i
734
735      !Variaveis locais
736      Integer :: j,k
737      Double precision :: A(1:18,1:18),Aux(1:21,1:21),Aux out(1:18,1:18),
         Aux simetrica(1:18,1:18)
738
739
740      Aux(1:21,1:21)=P(1:21,1:21) ! Matriz auxiliar para que P não se modifique
         nesta função e possa ser usada por outras
741
742      Call Calcula_matriz_semelhante(C,21,18,Aux,21,C,21,18,A)  ! A =
         C^T*Aux_in*C   , A tem dim 18
743      Call Calcula_matriz_semelhante(F,18,18,A,18,F,18,18,Aux out)  !
         Aux_out=F^T*A*F , Aux_out tem dimensão 18
744
745      !Simetrizando a matriz Aux out
746      Aux_simetrica=0.5D0*(transpose(Aux_out)+Aux_out)
747
748      !Mi(1:18,1:18)=Ji*Aux out(1:18,1:18)
749     Mi(1:18,1:18)=Ji*Aux simetrica(1:18,1:18)
750
751      !Teste de simetria
752     Do j=1,18
753       Do k=1,18
754
755         !if (Mi(j,k) .ne. Mi(k,j))      print*,i,j,"Matriz Mi nao é simetrica"
756          if (abs (Mi(j,k)-Mi(k,j)) >= 1.D-7 ) Print*,"Matriz Mi não é
             simetrica. Diferença :", (Mi(j,k)-Mi(k,j))
757       End do
758     End do
759
760     End Subroutine Monta matriz Mi
761
762      !----Montagem da matriz de rigidez elementar Si---!
763
764     Subroutine Monta_matriz_Si (Ji,P,C,F,Si,sigma,x,y,vi,nvertices,nospe) !
```



```
765
766      Implicit none
767
768      Double precision, intent(in) :: Ji, sigma
769      Double precision, intent(in) :: F(1:18,1:18),C(1:21,1:18), P(1:10,1:10)
770      Double precision, intent(in) :: x(1:nvertices), y(1:nvertices)
771      Integer, intent(in) :: nvertices,nospe, vi(1:nospe)
772
773      Double precision, intent(out) :: Si (1:18,1:18)
774
775      !Variaveis locais
776      Integer :: i,j
777      Double Precision, dimension (1:10,1:21) :: Dxx,Dyy,Dxy ! ! Matrizes do
         operador derivação
778      Double Precision, dimension (1:18,1:18) :: I 1,I 2,I 3,I 4,I 5,I 6 !
         matrizes auxiliares, parcelas da integral
779      Double precision :: A(1:21,1:21),A_(1:21,1:21),G(1:21,1:21),B(1:18,1:18),
         Aux out(1:18,1:18)
780      Double precision :: a1,a2,a3,a4,a5,a6,f1,f2,f3
781
782      Double Precision, dimension (1:10,1:10) :: Aux
783
784
785      !Definição de Dxx, Dyy e Dxy, as matrizes de derivação
786      Dxx=0.0D0
787      Dyy=0.0D0
788      Dxy=0.0D0
789
790      Dxx(1,4)=2.0D0
791      Dxx(2,7)=6.0D0
792      Dxx(4,11)=12.D0
793      Dxx(7,16)=20.D0
794
795      Dxx(3,8)=Dxx(1,4)
796      Dxx(6,13)=Dxx(1,4)
797      Dxx(10,19)=Dxx(1,4)
798      Dxx(5,12)=Dxx(2,7)
799      Dxx(9,18)= Dxx(2,7)
800      Dxx(8,17)=Dxx(4,11)
801
802      Dxy(1,5)=1.0D0
803      Dxy(2,8)=2.0D0
804      Dxy(4,12)=3.0D0
805      Dxy(5,13)=4.0D0
806      Dxy(8,18)=6.0D0
807
808      Dxy(3,9)=Dxy(2,8)
809      Dxy(6,14)=Dxy(4,12)
810      Dxy(10,20)=Dxy(5,13)
811      Dxy(7,17)=Dxy(5,13)
812      Dxy(9,19)=Dxy(8,18)
813
814      Dyy(1,6)=2.0D0
815      Dyy(3,10)=6.0D0
816      Dyy(6,15)=12.0D0
817      Dyy(10,21)=20.0D0
818
819      Dyy(2,9)=Dyy(1,6)
820      Dyy(4,13)=Dyy(1,6)
821      Dyy(7,18)=Dyy(1,6)
822      Dyy(5,14)=Dyy(3,10)
823      Dyy(8,19)= Dyy(3,10)
824      Dyy(9,20)=Dyy(6,15)
825
826      Aux(1:10,1:10)=P(1:10,1:10)    ! P é de dim 10
```



```
827
828      ! Matriz da forma quadratica da Integral do quadrado de w xx
829      Call Calcula matriz semelhante(Dxx,10,21,Aux,10,Dxx,10,21,A) ! A tem dim
         21,   A=Dxx^T*Aux*Dxx A,nA,mA,S,nS,B,nB,mB,produto
830      Call Calcula matriz semelhante(C,21,18,A,21,C,21,18,B) ! B tem dim 18,
         B=C^T*A*C
831      Call Calcula matriz semelhante(F,18,18,B,18,F,18,18,Aux_out) ! Aux_out
         tem dim 18, Aux out=F^T*B*F
832
833      I_1=0.5D0*(transpose(Aux_out)+Aux_out)
834
835      ! Matriz da forma quadratica da Integral do quadrado de wyy
836      Call Calcula matriz semelhante(Dyy,10,21,Aux,10,Dyy,10,21,A) ! A tem dim
         21, A=Dxx^T*Aux*Dxx
837      Call Calcula matriz semelhante(C,21,18,A,21,C,21,18,B) ! B tem dim 18,
         B=C^T*A*C
838      Call Calcula matriz semelhante(F,18,18,B,18,F,18,18,Aux_out) ! Aux_out
         tem dim 18, Calcula Aux out=F^T*B*F
839
840      I_6=0.5D0*(transpose(Aux_out)+Aux_out)
841
842      ! Matriz da forma quadratica da Integral do quadrado de wxy
843      Call Calcula matriz semelhante(Dxy,10,21,Aux,10,Dxy,10,21,A) ! A tem dim
         21, A=Dxx^T*Aux*Dxx
844      Call Calcula matriz semelhante(C,21,18,A,21,C,21,18,B) ! B tem dim 18,
         B=C^T*A*C
845      Call Calcula matriz semelhante(F,18,18,B,18,F,18,18,Aux_out) ! Aux_out
         tem dim 18, Calcula Aux out=F^T*B*F
846
847      I_4=0.5D0*(transpose(Aux_out)+Aux_out)
848
849
850
851      !------------Matriezes de Formas Bilineares-----------------------!
852      !Para simetrizar as formas bilineares, utilizaremos a matriz simétrica
         0.5*[(Dxx^T)*P*Dyy + (Dyy^T)*P*Dxx], que representa a mesma forma bilinear
853
854     ! Matriz da forma bilinear da Integral de wyy*wxx
855      Call Calcula matriz semelhante(Dxx,10,21,Aux,10,Dyy,10,21,A) ! A tem dim
         21, A=Dxx^T*Aux*Dxx
856      Call Calcula matriz semelhante(Dyy,10,21,Aux,10,Dxx,10,21,A ) ! A  tem
         dim 21, A =Dxx^T*Aux*Dxx
857      G=0.5D0*(A+A )
858      Call Calcula matriz semelhante(C,21,18,G,21,C,21,18,B) ! B tem dim 18,
         B=C^T*G*C
859      Call Calcula matriz semelhante(F,18,18,B,18,F,18,18,Aux_out) ! Aux_out
         tem dim 18, Calcula Aux out=F^T*B*F
860
861      I_3=0.5D0*(transpose(Aux_out)+Aux_out)
862
863
864      ! Matriz da forma bilinear da Integral de wxy*wxx
865      Call Calcula matriz semelhante(Dxx,10,21,Aux,10,Dxy,10,21,A) ! A tem dim
         21, A=Dxx^T*Aux*Dxx
866      Call Calcula matriz semelhante(Dxy,10,21,Aux,10,Dxx,10,21,A ) ! A  tem
         dim 21, A =Dxx^T*Aux*Dxx
867      G=0.5D0*(A+A )
868      Call Calcula matriz semelhante(C,21,18,G,21,C,21,18,B) ! B tem dim 18,
         B=C^T*G*C
869      Call Calcula matriz semelhante(F,18,18,B,18,F,18,18,Aux out) ! Aux out
         tem dim 18, Calcula Aux out=F^T*B*F
870
871      I 2=0.5D0*(transpose(Aux out)+Aux out)
872
873      ! Matriz da forma bilinear da Integral de wyy*wxy
```

```
877     Call Calcula_matriz_semelhante(C,21,18,G,21,C,21,18,B) ! B tem dim 18, B=C^T*G*C
878     Call Calcula_matriz semelhante(F,18,18,B,18,F,18,18,Aux out) ! Aux out tem dim 18, Calcula Aux_out=F^T*B*F
879
880     I 5=0.5D0*(transpose(Aux out)+Aux out)
881
882     ! Fim da montagem das matrizes auxiliares
883
884     ! Definição dos coeficientes das parcelas para a montagem de Si:
885
886     f1=(x(vi(3))-x(vi(1)))**2 + (y(vi(3)) -y(vi(1)))**2
887     f2=(x(vi(2)) - x(vi(1)))*(x(vi(3))- x(vi(1))) +(y(vi(2)) - y(vi(1)))*(y(vi(3)) - y(vi(1)))
888     f3=(x(vi(2))-x(vi(1)))**2 + (y(vi(2)) -y(vi(1)))**2
889
890     a1=(f1**2)/Ji**3
891     a2=-4.0D0*f2*f1/Ji**3
892     a3=2.0D0*(f2**2 + sigma*(Ji**2))/Ji**3
893     a4=(4.0D0*f2**2+2.0D0*(1.0D0-sigma)*Ji**2)/Ji**3
894     a5=-4.0D0*f2*f3/Ji**3
895     a6=f3**2/Ji**3
896
897     Si=a1*I_1+a2*I_2+a3*I_3+a4*I_4+a5*I_5+a6*I_6
898
899   ! Teste de simetria das matrizes de rigidez
900     Do i=1,18
901       Do j=1,18
902         if (Si(i,j) .ne. Si(j,i))       print*,i,j,"Matriz Si nao é simetrica"
903       !if (abs (Mi(i,j)-Mi(j,i)) >= 0.0000001 ) Print*,"Matriz Mi não é simetrica. Diferenca no elemento",i,j
904       End do
905     End do
906
907
908     End Subroutine Monta matriz Si
909
910
911   !----------------------Monta vetor de variaveis nodais ----------------------------!
912
913   !Essa subrotina monta um vetor ui, para cada triangulo i, análogo ao vetor vi que relaciona os
914   !indices locais e globais dos nós; mas ui considera todas as variáveis incidentes em cada nó,
915   !Dessa forma a dimensão de ui é 18, o numero de variáveis totais por elemento.
916   !As variáves estão aqui ordenadas como nas matrizes elementares: u1,u2,u3,ux1,ux2,ux3,uy1,uy2,uy3,...,uyy1,uyy2,uyy3.
917
918
919   Subroutine Monta Var nodais (vi,ui,nvertices,nv vertice)
920   Implicit none
921
922   ! Variáveis fantoches (Dummy)
923   Integer, intent(out) :: ui(1:18)
924   Integer, intent(in) :: vi(1:3), nvertices,nv vertice
925
926   ! Variávies locais
927   Integer:: l,i,k
928
929   Do k=1,3*nv vertice
930       l=1+int((k-1)/3)
931       i=1+mod((k-1),3)
932       ui(k)=vi(j) + nvertices*(l-1)
```



```
936
937   End Subroutine Monta Var nodais
938
939
940   Subroutine Monta matrizes globais(S ,M ,Si,Mi,ui,dimensao)
941
942     Implicit none
943
944     !Variáveis Dumy
945     Integer, intent(in) :: dimensao
946     Double Precision, intent(inout) :: S_(1:dimensao,1:dimensao),M_(1:dimensao,1:dimensao) ! Essas matrizes entram e saem dessa rotina a cada elemento triangular
947     Double Precision, intent(in) :: Si(1:18,1:18),Mi(1:18,1:18)
948     Integer, intent(in) :: ui(1:18)
949
950     !Variáveis locais
951     Integer ::l,k
952
953     Do l=1,18
954       Do k=1,18
955         S (ui(k),ui(l))= S (ui(k),ui(l)) + Si(k,l)
956         M (ui(k),ui(l))= M (ui(k),ui(l)) + Mi(k,l)
957       End Do
958     End Do
959
960   End Subroutine Monta_matrizes_globais
961     !----------CONTABILIZAR NA MONTAGEM DAS MATRIZES, AS CONDICÕES DE CONTORNO-----------------------------------!
962
963   Subroutine Contabiliza condicoes de contorno (S ,M ,fix,nv fix,r,dimensao, nvertices,contabilizou cc)
964     Implicit none
965
966     Integer, intent(in) :: fix(nvertices),dimensao,nvertices,r,nv fix
967     Integer, intent(out) :: contabilizou cc
968     Double precision, intent(inout) :: M_(1:dimensao,1:dimensao), S_(1:dimensao,1:dimensao)
969
970     !Variáveis locais
971     Integer:: i,j,k,n, ierr  !n é a dimensão do vetor eliminar
972
973     Integer, allocatable, dimension(:) :: eliminar
974
975     !------------------------Condição de bordo fixo -------------------------------------------!
976
977       ! Utilizando a info dos vértices que estão fixos na montagem das matrizes: As linhas e colunas relativas aos vértices que estão
978       ! fixos são deletadas/anuladas, uma vez que o valor da função nesses pontos ja é conhecido.
979       ! Nesse caso, que temos 6 variáveis nodais por vertice, temos que excluir/anular as linhas e colunas relativas as variáveis que temos certeza que serão nulas:
980           ! o valor de w e de suas primeiras derivadas: w,wx,wy
981           ! Nesse caso, para cada nó fixo temos que anular as linhas da matriz que correspondem aos valores
982           ! de w,wx,wy nesse vertice, i.e., para cada vertice i que for vertice de triangulo, temos 3 linhas e colunas para excluir.
983
984
985     print*,"Contabilizando as condicoes de contorno"
986     print*,"nv fix,nvertices,dimensao,r",nv fix,nvertices,dimensao,r
987
988     n=3*nv_fix
```



```
991     if (ierr/=0) then
992       print*,"Erro de alocacão do variavel eliminar"
993       stop
994     end if
995
996     !Montagem do vetor eliminar, das linhas e colunas que serao eliminadas. Esse vetor tem tamanho nv_fix*3
997     k=1
998     Do i=1,nvertices
999       if (fix(i).eq. 1)  then  ! se o vertice i esta fixo,
1000        Do j=0,2
1001          eliminar(k+nv fix*j)=nvertices*j+i   ! quardo em eliminar, a variável que será eliminada, nas posicões das linhas que serão eliminadas.
1002          ! print*,"posicao",k+nv fix*j,"do vetor eliminar recebe",nvertices*j+i
1003        End do
1004        k=k+1
1005      end if
1006    End do
1007
1008
1009    Do i=n,1,-1 ! n=3*nv_fix    ! laço decrescente no numero de variaveis, para o redimensionamento da matriz
1010      if (eliminar(i).ne.dimensao) then          ! se eliminar(i) nao for a ultima linha/coluna então faremos o deslocamento das linhas e colunas abaixo e a direita de eliminar(i) até dimensao-1.
1011       !print*,"Vertice",eliminar(i),"esta fixo."    ! Deslocamos a matriz uma unidade para cima e para a esquerda, respect.
1012        Do j=eliminar(i),dimensao-1
1013          S (i,:)=S (i+1,:)
1014          S_(:,j)=S_(:,j+1)
1015          M (i,:)=M (i+1,:)
1016          M (:,i)=M (:,i+1)
1017        End do
1018      End if
1019    End do
1020
1021    contabilizou cc=1
1022
1023  End Subroutine Contabiliza condicoes de contorno
1024
1025    !-----Redimensiona  matrizes---
1026  Subroutine Redimensiona matrizes(dimensao,r,M,S,M ,S )
1027    Implicit none
1028
1029    Integer, intent(in) :: dimensao,r
1030    Double precision, intent (in) :: M_(1:dimensao,1:dimensao),S_(1:dimensao,1:dimensao)
1031    Double precision, intent (out) :: M(1:r,1:r), S(1:r,1:r)
1032
1033    !Variaveis locais
1034    Integer :: i,j
1035
1036  print*,"Redimensionando as matrizes, de dim",dimensao,"para",r
1037  Do j=1,r
1038    Do i=1,r
1039      S(i,j)=S_(i,j)
1040      M(i,j)=M (i,j)
1041    End do
1042  End do
1043
1044  ! Teste de simetria das matrizes de massa e rigidez globais
1045  Do i=1,r
```



```
1049    End do
1050  End do
1051
1052
1053  End subroutine Redimensiona matrizes
1054
1055
1056    !--------------------- Calcula Matrizes "semelhantes" -------------------!
1057  Subroutine Calcula matriz semelhante(A,nA,mA,S,nS,B,nB,mB,produto)
1058    ! Retorna em produto uma matriz "semelhante" a Matriz S.  produto= A^T*S*B
1059    ! Essa funcão é mais qeral pois permite o fazer o produto a esquerda e a direita com matrizes diferentes e não necessariamente quadradas.
1060    ! A tem dimensão nAxmA, S é quadrada e tem dimensão nS e B tem dimensão nBxmB.
1061
1062    Implicit none
1063    Integer, intent(in) :: nS, nA, mA, nB, mB
1064    Double precision, intent(in) :: A(nA,mA), S(nS,nS), B(nB,mB)
1065    Double precision, intent(out), dimension(1:mA,1:mB) :: produto
1066
1067    ! Variáveis locais
1068    Double precision :: M1(mA,nS),M2(mA,mB)
1069
1070    !print*, "dimensão de A",nA,mA ,"dimensão de S", nS, "Dimensao de B",nB,mB
1071    if(nA==nS) then
1072     M1=matmul(transpose(A),S)
1073    ! print*, "Tamanho de M1", size(M1)
1074    else
1075     print*,"Impossível multiplicar matrizes, dimensões não batem"
1076     stop
1077    end if
1078    if(nS==nB) then
1079     M2=matmul(M1,B)
1080     !print*, "Tamanho de M2", size(M2)
1081    else
1082     print*,"Impossível multiplicar matrizes, dimensões não batem"
1083     stop
1084    end if
1085    produto(1:mA,1:mB)=M2(1:mA,1:mB)
1086
1087  End subroutine Calcula matriz semelhante
1088
1089
1090  !-------------- Subrotina GERAL para escrever uma matriz quadrada num arquivo-----------------------------------------!
1091  Subroutine Escreve_Matriz_em_Arquivo (M,dimensao,unidade,nome_arquivo,sta)
1092
1093  Implicit none
1094
1095    Double Precision, intent(in) :: M(1:dimensao,1:dimensao)
1096    Character (len=20) :: nome arquivo
1097    Character (len=3) :: sta ! status do arquivo: "old" or "new"
1098    Integer :: unidade, dimensao ! unidade onde o arquivo será aberto.
1099
1100    ! Variaveis locais
1101    Integer:: i
1102
1103    Open(unidade,file=nome arquivo,status=sta)
1104    Write(unidade,*) "Matriz de dimensao:",dimensao
1105    Write(unidade,*)
1106
1107    Do i=1,dimensao
1108      Write(unidade,*),M(i,:)
1109    End do
1110
```

```
1112
1113    End Subroutine Escreve matriz em arquivo
1114
1115
1116    !----------------REGISTRO DAS MATRIZES DE RIGIDEZ E DE MASSA EM ARQUIVO
        .txt-------------------------------!
1117
1118    Subroutine Escreve_matrizes(r,M,S)
1119
1120      Implicit none
1121
1122      Integer, intent(in) :: r
1123      Double precision, intent(in) :: M(r,r), S(r,r)
1124
1125      !Variaveis locais
1126      Integer:: i
1127
1128      Open(43, file=
         '/home/daniela/Documents/fortran/placa conforme 18coef/Saida/Matriz rigid
         ez.txt', status='old')
1129      Open(44, file=
         '/home/daniela/Documents/fortran/placa conforme 18coef/saida/Matriz massa
         .txt', status='old')
1130      Write(43,*),r
1131      Write(44,*),r
1132      Do i=1,r
1133        Write(43,*),S(j,:)
1134        Write(44,*),M(i,:)
1135      End do
1136
1137    End subroutine Escreve matrizes
1138
1139     !----------------IMPRESSÃO DOS AUTOVALORES E AUTOVETORES EM ARQUIVO
         .txt-------------------------------!
1140
1141    Subroutine Imprime autovetores(r,w,S)
1142
1143      Implicit none
1144      Integer, intent(in):: r
1145      Double precision, intent(in) :: w(1:r),S(1:r,1:r)
1146
1147      !Variaveis locais
1148      Integer :: i,j
1149
1150      Open(45,file=
         '/home/daniela/Documents/fortran/placa_conforme_18coef/Saida/Autovetores.
         txt',status='old')
1151      Write(45,*),r
1152
1153      Do i=1,25   ! Vou escrever só os 25 primeiros, do total de r autovalores
1154        Write(45,*) w(i)
1155      End do
1156      Write(*,*),"Primeiro autovalor:",w(1)
1157      !Do i=1,r
1158      !  Write(45,*)
1159      !  Write(45,*)"Autovalor",i,":", w(i)
1160
1161      !   Write(45,*)
1162      ! Write(45,*)"autovetor associado:"
1163      ! Do j=1, r
1164      !    Write(45,'(f30.24)'),S(j,i)
1165      ! End do
1166      !End do
1167
1168
```



```
1169    End subroutine Imprime_autovetores
1170
1171      !------------------------------ ARQUIVOS PARA O SCILAB (função
          fec)---------------------------------------!
1172      ! A função FEC só vale para quando queremos plotar o valor de u nos
          vértices do triangulo-------------------------!
1173
1174    Subroutine Dados_plotagem(nvertices,nele,x,y,S,T,r,fix)
1175
1176      Implicit none
1177      Integer, intent(in) :: nvertices,r,nele, T(1:4,1:nele),fix(1:nvertices)
1178      Double precision, intent(in) :: x(1:nvertices), y(1:nvertices),S(1:r,1:r)
1179
1180    ! Variáveis locais
1181      Integer :: i,j,k,l ! contadores
1182
1183      Open (50,file=
         '/home/daniela/Documents/fortran/placa conforme 18coef/Saida/coordxy02 sci
         lab.txt', status='old')
1184
1185      Do i=1,nvertices
1186        Write(50,*) x(i), y(i)
1187      End do
1188
1189      Open (51,file=
         '/home/daniela/Documents/fortran/placa conforme 18coef/Saida/TO2 scilab.tx
         t', status='old')
1190      Do i=1,nele
1191        Write(51,*) T(1,i), T(2,i), T(3,i), T(4,i), T(4,i) ! Repeti o último,
          pois o arquivo T02 deve apresentar cinco dados, a informação do
          triangulo mais uma falq para o Scilab
1192      End do
1193
1194
1195      Open (52,file=
         '/home/daniela/Documents/fortran/placa conforme 18coef/Saida/autofuncO2 sc
         ilab.txt', status='old')
1196
1197      Do k=1,25   ! Escrever 25 dos r autovetores
1198        l=1
1199        Do i=1,nvertices
1200          !se o vertice estava fixo, escreva 0.0, caso contrario, escreva o
            proximo elemento de S
1201          if (fix(i).eq.1) then
1202            Write(52,*) 0.00
1203          else
1204            write(52,*) S(l,k)
1205            l=l+1
1206          end if
1207
1208        End do
1209      End do
1210
1211        Close(50)
1212        Close(51)
1213        Close(52)
1214
1215    End subroutine Dados_plotagem
1216
1217    Subroutine Espectro (A,n,i,simetrica)
1218     !Calcula os autovalores de uma matriz real
1219     !Se a matriz for simetrica, usa a função dsyev
1220     !Se a matriz for arbitrartia, usa a função dgeev
1221
1222     Implicit none
```



```
1223
1224      !Variáveis dummy
1225      Integer, intent(in) :: i,n
1226      Double precision, intent(in) :: A (1:n,1:n)
1227      Character*1, intent(in) :: simetrica ! se simetrica='S', então é
          simetrica, se for 'N', não é simetrica
1228
1229      !Variáveis locais
1230      Integer :: j
1231      Integer :: lda, lwork, info, ierr
1232      Double precision, allocatable, dimension (:) :: work
1233      Double precision, allocatable, dimension (:) :: norma
1234      Double precision :: maior modulo, menor modulo
1235
1236
1237      !Variáveis locais só para a função dsyev
1238
1239      Double Precision, allocatable, dimension (:) :: w
1240      Character*1 :: jobz, uplo='U'
1241
1242      !Variáveis para a função dgeev
1243
1244      Integer :: ldvr,ldvl
1245      Double Precision, allocatable, dimension (:,:) :: vl,vr
1246      Double Precision, allocatable, dimension (:) :: wr,wi
1247      Character*1 :: jobvl, jobvr
1248
1249      lda=n
1250
1251      if(simetrica =='S') then ! Se a matriz A é simetrica
1252        lwork = max(1,3*n-1) ! é a dimensão do vetor work
1253        jobz='N' ! N=calcula apenas os autovalores; V=calcula autovalores e
          autovetores
1254
1255        !Alocar as variaveis alocaveis
1256        allocate (work(1:lwork), w(1:n), stat = ierr)
1257        if (ierr/=0) then
1258          print*,"Erro de alocacão das variáveis work e w"
1259          stop
1260        end if
1261
1262        !dsyev (jobz, uplo, n, A, lda, w, work, lwork, info)
1263        Call dsyev (jobz, uplo, n, A, lda, w, work, lwork, info) ! calcula os
          autovalores (e opcionalmente os autovetores) de uma matriz symetrica
1264        print*, "info espectro=", info
1265
1266        !Imprimir os 5 primeiros autovalores, se existirem valores negativos,
          estes aparecerão primeiro, pois os autovalores aparecem em ordem
          crescente
1267        !write(103,*),"Matriz simetrica, primeiros autovalores reais"
1268        write(103,*)
1269        Do j=1,n
1270          !write(103, '(I4,5E20.8)') i, w(1),w(2),w(3),w(4),w(5)
1271          write(103, '(E20.8)')  w(i)
1272        End do
1273      end if
1274
1275      if(simetrica =='N') then ! Se a matriz A não é simetrica
1276        ldvl=n
1277        ldvr=n
1278        lwork = max(1,3*n) ! é a dimensão do vetor work
1279        jobvl='N' ! N=não calcula autovetores à esquerda
1280        jobvr='N' ! N=não calcula autovetores à direita
1281
1282        !Alocar as variaveis alocaveis
```



```
1284        if (ierr/=0) then
1285          print*,"Erro de alocacão das variáveis do caso não simetrico"
1286          stop
1287        end if
1288
1289        Call dgeev(jobvl, jobvr, n, A, lda, wr, wi, vl, ldvl,vr,ldvr, work, lwork
            , info)
1290        print*, "info espectro=", info
1291
1292        !Imprimir os 5 primeiros autovalores, se existirem valores negativos,
            estes aparecerão primeiro, pois os autovalores aparecem em ordem
            crescente
1293        !print*,"Matriz genérica, primeiros autovalores: parte real e parte
            imaginária"
1294        write(*,*)
1295
1296        !Cálculo do módulo de cada autovalor complexo
1297        Do i=1,n
1298         norma(i) = sqrt(wi(i)**2 + wr(i)**2)
1299        End do
1300
1301        maior modulo=maxval(norma)
1302        menor_modulo=minval(norma)
1303
1304        !Do i=1,n
1305        !  write(103,'(I3,2E18.8,3x,2E18.8,3x,2E18.8,3x,2E18.8)')
          i,wr(1),wi(1),wr(2),wi(2),wr(3),wi(3),wr(4),wi(4),wr(5),wi(5)
1306        !   write(103,'(2E18.8)') wr(i),wi(i)
1307           write(103, '(I3,f18.8,2x,f18.8)') i, menor modulo, maior modulo
1308        !End do
1309
1310      end if
1311
1312      End subroutine
1313
1314    Subroutine Cholesky (A,n,i)
1315
1316    ! Calcula a decomposição de Cholesky para matrizes simétricas e definida
      positivas.
1317    ! Estamos uando essa funcão como criterio para decidir sobre a
      positividade definida de uma matriz simétrica.
1318    ! Por se tratar de um teste, essa funcão não retorna a fatoracão de
      Cholesky da matriz A, ele apenas testa indica se foi possivel fazer a
      fatoracão
1319
1320      Implicit none
1321
1322      Double Precision, intent(in) :: A(1:n,1:n)
1323      Integer, intent(in) :: n, i
1324
1325      !Variáveis locais
1326      Double precision :: B(1:n,1:n)
1327      Character :: uplo='L'
1328      Integer :: lda, info
1329
1330      lda=n
1331      B(1:n,1:n)=A(1:n,1:n)
1332
1333      Call dpotrf(uplo, n, B, lda, info)   ! função de calucula a fatoração de
          cholesky
1334      if (info==0) then
1335        write(102,*),"Exito! matriz", i, "  é positiva definida"
1336        !Do k=1,n
1337         ! write(*,'(21f10.4)') B(k,:)
1338        !End do
```

Referências Bibliográficas

[1] Ciarlet, P. G.: *The Finite Element Method for Elliptic Problems, V4.* North-Holland Publishing Company, 1978. 19, 21, 28, 31, 36, 39

[2] Courant, R. e D. Hilbert: *Methods of Mathematical Physics, V1, edição 1.* Interscience Publishers, Inc., New York, 1953. 5, 8, 14, 94

[3] Evans, L. C.: *Partial Differential Equations, edição 2.* American Mathematical Society, 2010. 21, 31

[4] Felippa, C. A.: *Kirchhoff Plates: Field Equations.* http://www.colorado.edu/engineering/CAS/courses.d/AFEM.d/AFEM.Ch20.d/AFEM.Ch20.pdf. Último acesso em 19/10/2016. xi, 2, 5

[5] Fish, J. e T. Belytschko: *Um Primeiro Curso em Elementos Finitos.* LTC - Livros Técnicos e Científicos Editora S.A., 2009.

[6] Gross, D., W. Hauger, W. W. J. Schröder e J. Bonet: *Engineering Mechanics 2: Mechanics of Materials.* Springer-Verlag Berlin Heidelberg, 2011. 2, 3

[7] Grossmann, C., H. Roos e M. Stynes: *Numerical Treatment of Partial Differential Equations.* Springer-Verlag Berlin Heidelberg, 2005. 19, 28

[8] Leissa, A. W.: *Vibration of Plates.* Nacional Aeronautics and Space Administration, 1969. xii, 77, 78, 79, 82, 83, 92

[9] Müller, T.: *Numerical Chladni figures.* European Journal of Physics, 34:1067–1074, 2013. 79

[10] N. Katz, A.G.Peano, M. R.: *Nodal Variables for Complete Conforming Finite Element for arbitrary polimonial order.* Computacional and Math Aplications, 4:85–112, 1978. 44

[11] Provenzano, L.: *Eigenvalues of harmonic and poly-harmonic operators subject to mass density perturbations.* Tese de Doutoramento, Universit'a degli Studi di Padova, 2012. (Capítulo 2). Disponível em http://www.math.unipd.it/~proz/file/thesis.pdf. Último acesso em 20/12/2016. 28

[12] Rayleigh, J. W. S.: *The Theory of Sound, V1.* Dover Publications, 1945. xii, 3, 88

[13] Rossin, T. D.: *Chladni's law for vibrating plates.* American Journal of Physics, 50:271–274, 1982. xi, 5

[14] Sayas, F. J.: *A gentle introducrtion to the Finite Element Method*. 2008.

[15] Schwarz, H.: *Finite Element Methods.* Academic Press, 1988. 5, 72

[16] Shewchuk, J. R.: *Triangle: A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator.* https://www.cs.cmu.edu/~quake/triangle.html. Último acesso em 19/10/2016. 77

[17] Terzopoulos, D.: *Multi-Level Reconstruction of Visual Surfaces: Variational Principles and Finite Element Representations.* Massachusetts Institute of Technology Artificial Intelligence Laboratory, 1982.

[18] Zienkiewicz, O. C.: *Finite Element Method.* Butterworth-Heinemann, 2000. 72