UNIVERSIDADE ESTADUAL DO MARANHÃO

CENTRO DE CIÊNCIAS TECNOLÓGICAS

CURSO DE ENGENHARIA CIVIL

TARIK DE SOUSA MENEZES

IMPLEMENTAÇÃO DAS PLACAS DE KIRCHHOFF E DE MINDLIN UTILIZANDO O MÉTODO DOS ELEMENTOS FINITOS

São Luís
– 2017 –

TARIK DE SOUSA MENEZES

IMPLEMENTAÇÃO DAS PLACAS DE KIRCHHOFF E DE MINDLIN UTILIZANDO O MÉTODO DOS ELEMENTOS FINITOS

Monografia apresentada ao curso de Engenharia Civil da Universidade Estadual do Maranhão para o grau de bacharelado em Engenharia Civil.

Orientador: Prof. Me. Carlos César Pereira de Almeida

São Luís
– 2017 –

Menezes, Tarik de Sousa.
Implementação das placas de Kirchhoff e de Mindlin utilizando o método dos elementos finitos / Tarik de Sousa Menezes. – São Luís, 2017.
191f.

Monografia (Graduação) – Curso de Engenharia Civil, Universidade Estadual do Maranhão, 2017.

Orientador: Prof. Me. Carlos César Pereira de Almeida.

1. Análise estrutural. 2. Métodos numéricos. 3. Programação. I. Título.

CDU 624.04:004.4

TARIK DE SOUSA MENEZES

IMPLEMENTAÇÃO DAS PLACAS DE KIRCHHOFF E DE MINDLIN UTILIZANDO O MÉTODO DOS ELEMENTOS FINITOS

Monografia apresentada ao curso de Engenharia Civil da Universidade Estadual do Maranhão para o grau de bacharelado em Engenharia Civil.

Aprovado em 11 / 12 / 2017

BANCA EXAMINADORA

Profº. Me. Carlos César Pereira de Almeida (Orientador)

Mestre em Engenharia Civil

Universidade Estadual do Maranhão

Profº. Me. Clodoaldo Cesar Malheiros Ferreira

Mestre em Engenharia Civil

Universidade Estadual do Maranhão

Profº. Dr. Eduardo Aurélio Barros Aguiar

Doutorado em Engenharia Civil – Engenharia de Estruturas

Universidade Estadual do Maranhão

Resumo

Descreve o procedimento da implementação de um programa de computador para análise de placas finas e grossas com carregamento lateral. Destaca as hipóteses da formulação das equações diferenciais governantes. Enumera os passos para obtenção das matrizes de rigidez e dos vetores de cargas nodais equivalentes. Utiliza a teoria da elasticidade para obtenção das relações entre deslocamentos, deformações e esforços. Descreve elementos isoparamétricos e define suas funções de forma. Descreve o procedimento da quadratura de Gauss para integração numérica.

Palavras-chave: análise estrutural, lajes finas, lajes grossas, métodos numéricos, teoria das placas, teoria de placas de Kirchhoff, teoria de placas de Mindlin, quadratura de Gauss, programação, MATLAB.

Abstract

Describes the procedure of implementing a computer program for analysis of thin and thick plates with lateral loading. Highlights the assumptions for the formulation of the governing differential equations. It lists the steps to obtain the stiffness matrices and the vectors of equivalent nodal forces. It uses the theory of elasticity to obtain the relations between displacements, strains and stresses. Describes isoparametric elements and defines their shape functions. Describes the Gauss quadrature procedure for numerical integration.

Keywords*: structural analysis, thin slabs, thick slabs, numerical methods, plate theory, Kirchhoff plate theory, Mindlin plate theory, Gauss quadrature, programming, MATLAB.*

Sumário

Introdução

Com o advento da tecnologia e o avanço da engenharia, as ferramentas computacionais vêm ganhando cada vez mais espaço nas Engenharias por conta da praticidade e eficiência em suas utilizações diárias.

Os estudos teóricos desenvolvidos pelos pioneiros nas engenharias vêm cada vez mais sendo refinados e modelados para aplicação em computador e cada vez mais softwares especializados vêm sendo desenvolvidos com base nessas teorias.

Em Análise Estrutural, existem as chamadas Teoria das Placas de Kirchhoff, utilizada fundamentalmente para analisar o comportamento estrutural em lajes finas, e a Teoria das Placas de Reissner-Mindlin, para analisar o comportamento em lajes espessas.

O presente trabalho tem como objetivo principal a elaboração de um programa de computador, em linguagem MATLAB, que utiliza da formulação dessas teorias para análise de lajes de espessura fina e de espessura grossa. Para atingir esse objetivo, será feito uma explicação das teorias de placas de Kirchhoff e de Reissner-Mindlin, o procedimento para obtenção das matrizes de rigidez e dos vetores de cargas nodais equivalentes, uma breve descrição do elemento de referência utilizado neste trabalho, suas funções de forma e, por fim, a técnica de integração utilizada para obtenção dos itens anteriores.

1. Teoria clássica de placas finas e suas equações governantes

1.1 Teoria clássica dos pequenos deslocamentos

Uma análise de tensões matematicamente exata de uma placa fina – sujeita a cargas agindo perpendicularmente a sua superfície – requer solução das equações diferenciais tridimensionais da elasticidade. Entretanto, na maioria dos casos, tal aproximação encontraria dificuldades matemáticas insuportáveis. Ainda assim, para a vasta maioria das aplicações técnicas da teoria clássica das placas finas de Kirchhoff, produz resultados suficientemente precisos sem a necessidade de criar uma completa análise de tensões tridimensional. Em consequência, a teoria clássica de placas ocupa uma posição única neste assunto. É formulada em termos de deslocamentos transversais $w(x, y)$ para os quais as equações diferenciais governantes são de quarta ordem, requerendo apenas duas condições de contorno para serem satisfeitas em cada borda. As simplificações usadas na derivação da equação da placa são baseadas nas seguintes hipóteses:

1- O material é homogêneo, isotrópico e elástico-linear; isto é, segue a lei de Hooke;
2- A placa é plana inicialmente;
3- A superfície média da placa se mantém não tensionada;
4- A espessura *constante* da placa é pequena comparada com suas outras dimensões; isto é, a menor dimensão lateral da placa é pelo menos 10 vezes maior que sua espessura;
5- Os deslocamentos transversais w(x, y) são pequenos comparados à espessura da placa. Um deslocamento máximo de um décimo da espessura é considerado o limite da teoria dos pequenos deslocamentos;
6- As rotações da superfície deformada são pequenas comparadas com a unidade;
7- Seções consideradas normais à superfície média antes da deformação permanecem planas e normais à superfície média deformada. Esta hipótese representa uma extensão da hipótese de Bernoulli utilizada em vigas para placas;
8- O esforço normal na direção transversal à superfície da placa pode ser desprezado.

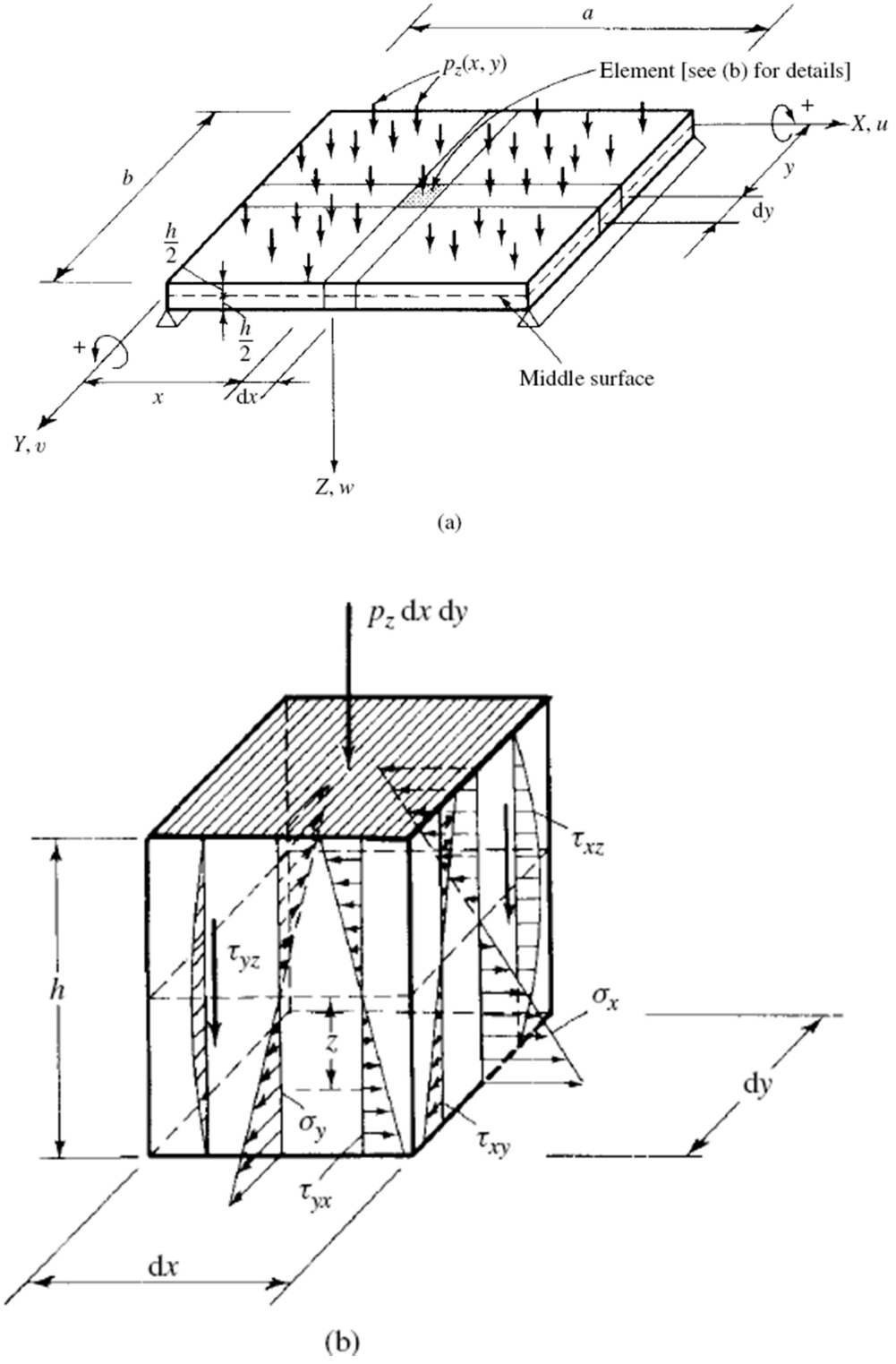
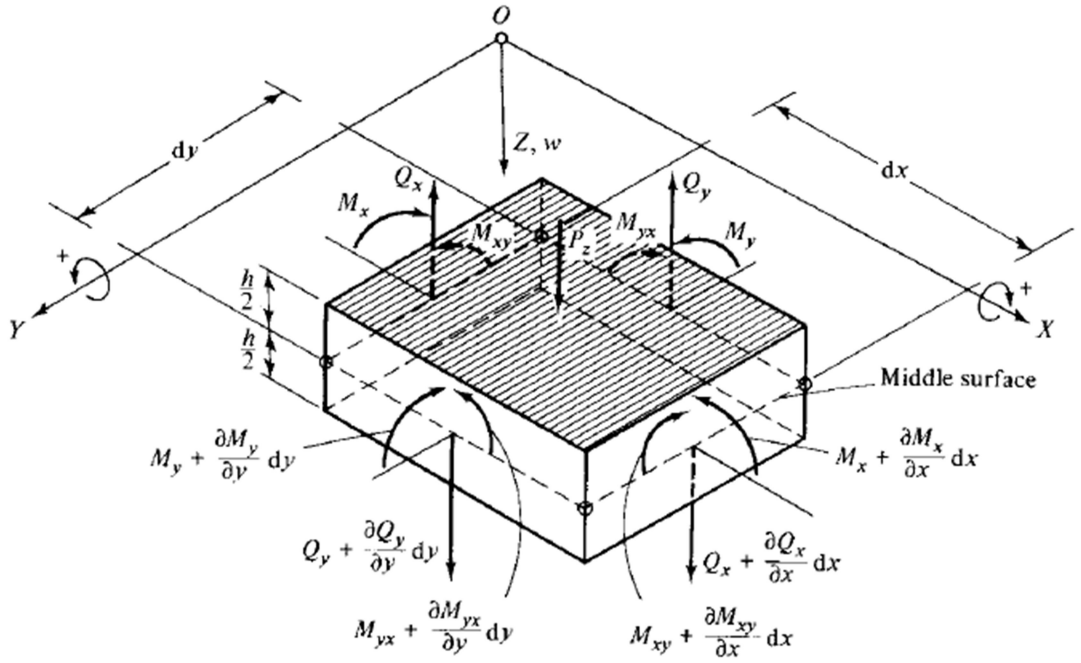


Figura 1.1-1 (a) Placa retangular carregada lateralmente. (b) Componentes de tensão num elemento de placa. Fonte: (Szilard, 2004)

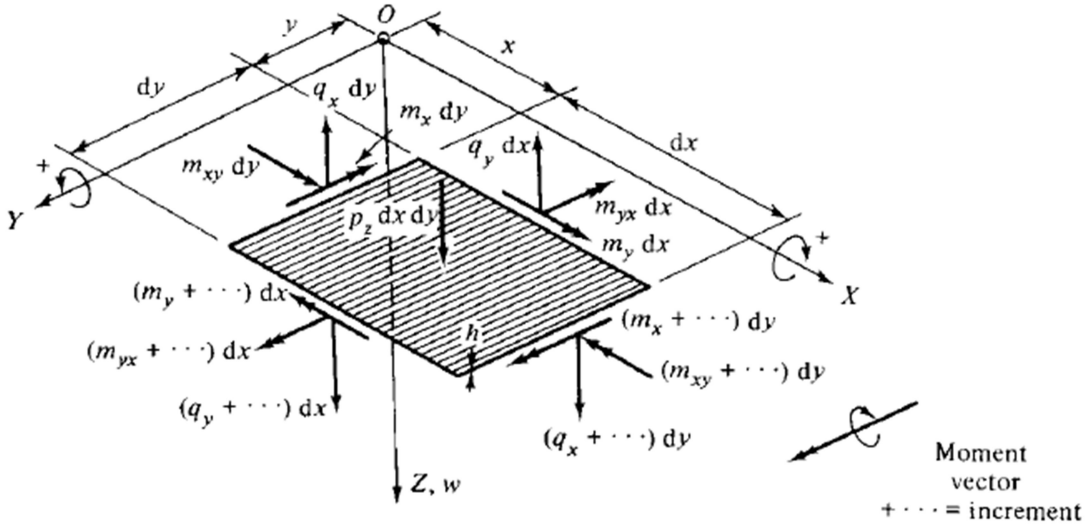
Com a ajuda dessas hipóteses, o problema de tensões elásticas, originalmente tridimensional, é reduzido a problemas de placas bidimensionais. Muitas dessas hipóteses são muito familiares àquelas equivalentes da teoria da viga elementar.

E também a uma análise de tensões tridimensional mais exata. Extensivos testes de grandes e pequenas escalas provaram a validade dessas hipóteses simplificadoras.

Para placas retangulares o uso do sistema de coordenadas retangulares cartesianas é mais conveniente. As forças e tensões externas e internas e componentes de deflexão u, v e w são consideradas positivas quando apontarem para a mesma direção positiva dos eixos coordenados X, Y e Z. Na prática geral de engenharia, momentos positivos produzem tração nas fibras localizadas na parte inferior pertinente à seção. Esta convenção de sinais também é mantida para placas.



(a) Detailed illustration



(b) Schematic illustration

Figura 1.1-2. Forças internas e externas no elemento da superfície média. Fonte: (Szilard, 2004)

Para satisfazer o equilíbrio do elemento, forças e momentos negativos internos precisam agir nas bordas. O primeiro conjunto de forças internas se refere à direção da superfície normal pertinente à seção na qual a força age. O conjunto de flexões internas e de momentos torçores se refere às tensões nas quais são produzidas. Portanto, o momento fletor Mx, dessa forma, é causado por tensões normais σ_x e rotaciona em volta do eixo Y. Essas rotações são padrões na teoria da elasticidade, mas contradizem aquelas utilizadas nos métodos numéricos contemporâneos, onde Mx se refere a um momento rotacionando em volta do eixo X. Finalmente, deve-se notar que o segundo conjunto de tensões de cisalhamento indicam a direção na qual os vetores apontam.

1.2 Equação da placa no sistema de coordenadas cartesianas

a) Equilíbrio do elemento de placa:

Supondo que a placa é sujeita apenas a forças laterais, das seis equações fundamentais do equilíbrio, as três a seguir podem ser utilizadas:

- $\sum Mx = 0$; $\sum My = 0$ e $\sum Pz = 0$.

O comportamento da placa é em muitos aspectos análogo a uma grelha bidimensional. Portanto, a carga externa Pz é carregada por forças de cisalhamento transversais Qx e Qy e momentos fletores Mx e My. O desvio significativo da grelha bidimensional é a presença de momentos torçores Mxy e Myx. Na teoria das placas costuma-se lidar com forças internas e momentos por unidade de comprimento da superfície média. Para distinguir essas forças internas das resultantes mencionadas acima, as notações utilizadas serão: q_x, q_y, m_x, m_y, m_{xy} e m_{yx}.

O procedimento envolvido em armar a equação diferencial de equilíbrio é o seguinte:

1- Selecione um sistema de coordenadas conveniente e faça um esboço do elemento;
2- Mostre todas as forças internas e externas agindo no elemento;
3- Assinale forças internas positivas com incremento (q_x + ... , q_y + ... , etc) nos lados próximos;

4- Assinale forças internas negativas nos lados opostos;

5- Expresse os incrementos em séries de Taylor truncadas, da forma:

$$q_x + dq_x = q_x + \frac{\partial q_x}{\partial x} dx$$

$$m_y + dm_y = m_y + \frac{\partial m_y}{\partial y} dy$$

6- Expresse o equilíbrio das forças internas agindo no elemento.

$$\frac{\partial m_x}{\partial x} + \frac{\partial m_{yx}}{\partial y} = q_x \tag{1.1}$$

$$\frac{\partial m_y}{\partial y} + \frac{\partial m_{xy}}{\partial x} = q_y \tag{1.2}$$

$$\frac{\partial q_y}{\partial y} + \frac{\partial q_x}{\partial x} = -p_z \tag{1.3}$$

$$\frac{\partial^2 m_x}{\partial x^2} + \frac{\partial^2 m_y}{\partial y^2} + 2\frac{\partial^2 m_{xy}}{\partial x \partial y} = -p_z(x,y) \tag{1.4}$$

Os momentos fletores e torçores da equação (1.4) dependem das deformações, e as deformações são funções das componentes de deslocamento (u, v, w). Portanto, nos próximos passos, relações entre momentos internos e componentes de deslocamento são procuradas.

b) Relação entre tensão, deformação e deslocamento:

A hipótese de que o material seja elástico permite o uso da lei de Hooke bidimensional.

$$\sigma_x = E\varepsilon_x + \nu\sigma_y, \text{ e} \tag{1.5}$$

$$\sigma_y = E\varepsilon_y + \nu\sigma_x \tag{1.6}$$

as quais relacionam tensão e deformação num elemento de placa. Substituindo (1.6) em (1.5) temos:

$$\sigma_x = \frac{E}{1-\nu^2}\left(\varepsilon_x + \nu\,\varepsilon_y\right) \tag{1.7}$$

e de maneira similar:

$$\sigma_y = \frac{E}{1-\nu^2}\left(\varepsilon_y + \nu\,\varepsilon_x\right) \tag{1.8}$$

Os momentos torçores m_{xy} e m_{yx} produzem tensões de cisalhamento τ_{xy} e τ_{yx} no plano, nos quais são relacionadas com a deformação de cisalhamento γ pela relação de Hooke:

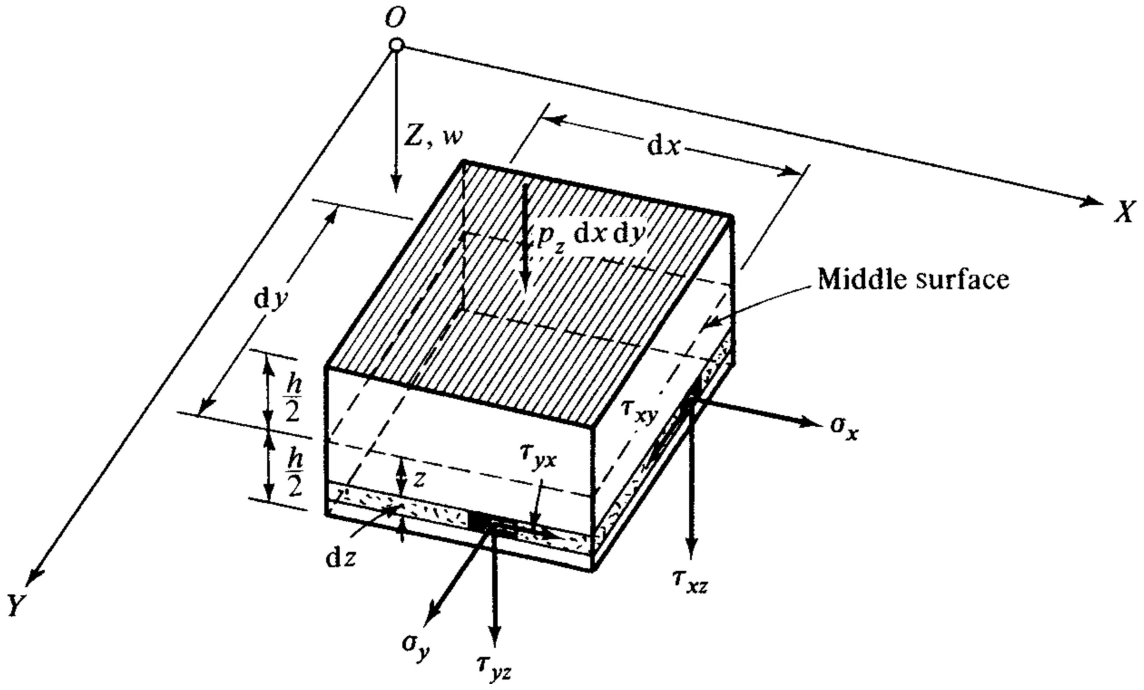


Figura 1.2-1. Tensões num elemento de placa. Fonte: (Szilard, 2004)

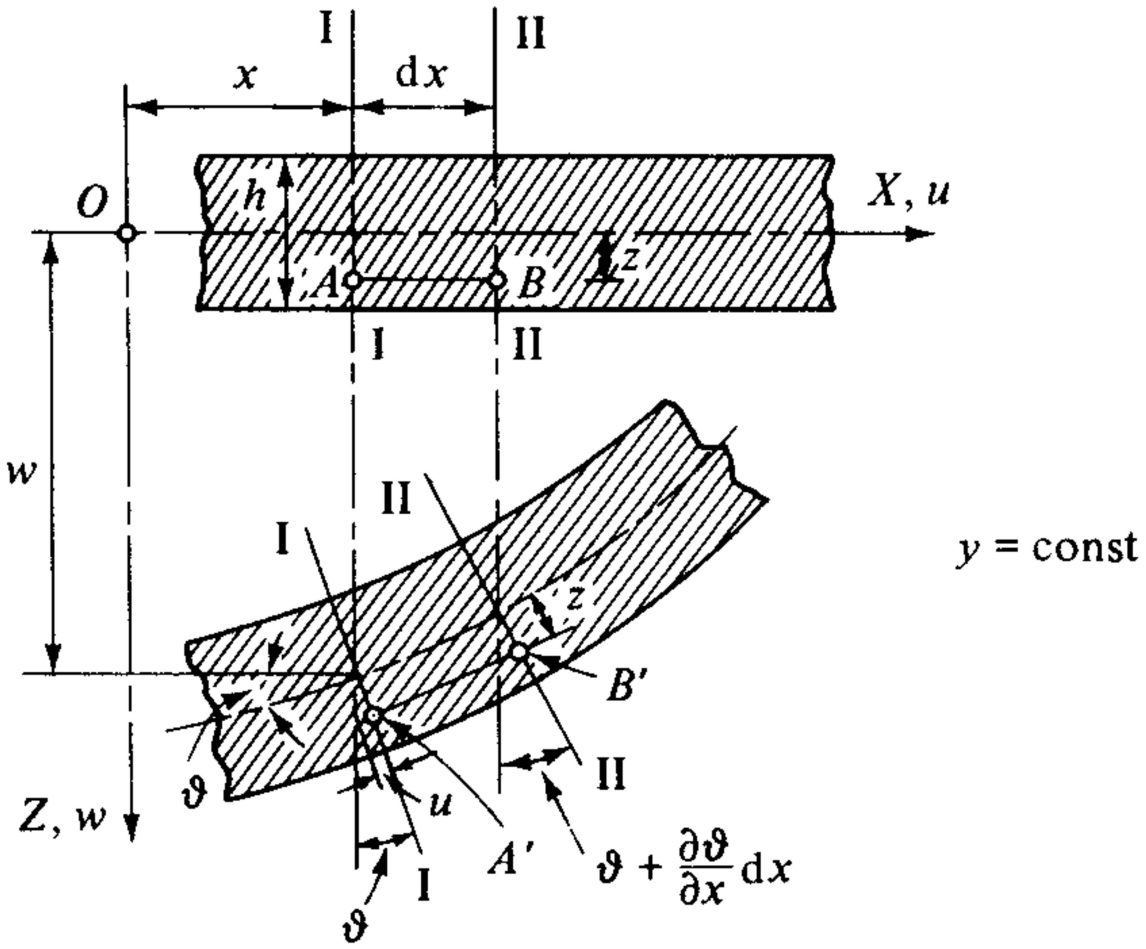


Figura 1.2-2. Seção antes e depois da deflexão. Fonte: (Szilard, 2004)

$$\tau_{xy} = G\gamma_{xy} = \frac{E}{2(1+\nu)}\gamma_{xy} = \tau_{yx} \tag{1.9}$$

Em seguida, consideramos a geometria da placa deformada para expressar as deformações em termos de coeficientes de deslocamento. Considerando uma seção em uma constante **y**, comparemos a seção antes e depois da deflexão (Fig.1.2-2). Utilizando as hipóteses 6 e 7, mostradas anteriormente, expressamos o ângulo de rotação na forma:

$$v = -\frac{\partial w}{\partial x} \text{ e } v + \ldots = v + \frac{\partial v}{\partial x}dx \tag{1.10}$$

Depois da deformação de uma fibra de tamanho $\overline{AB}$, localizada a uma distância **z** do plano médio da placa, se torna $\overline{A'B'}$. Utilizando a definição de deformação, escrevemos:

$$\varepsilon_x = \frac{\Delta dx}{dx} = \frac{\overline{A'B'} - \overline{AB}}{\overline{AB}} = \frac{\left[dx + z\left(\frac{dv}{dx}\right)\right] - dx}{dx} = z\frac{\partial v}{\partial x}$$

Substituindo a primeira das equações de (1.10) na equação acima, obtemos:

$$\varepsilon_x = -z\frac{\partial^2 w}{\partial x^2} \tag{1.11}$$

e de forma similar:

$$\varepsilon_y = -z\frac{\partial^2 w}{\partial y^2} \tag{1.12}$$

Agora determinando a distorção angular $\gamma_{xy} = \gamma' + \gamma''$ comparando um paralelogramo retangular, localizado a uma distância constante **z** a partir da superfície média, com sua forma deformada A'B'C'D' na superfície da placa defletida. De dois triângulos pequenos:

$$\gamma' = \frac{\partial v}{\partial x}, \text{ e} \tag{1.13}$$

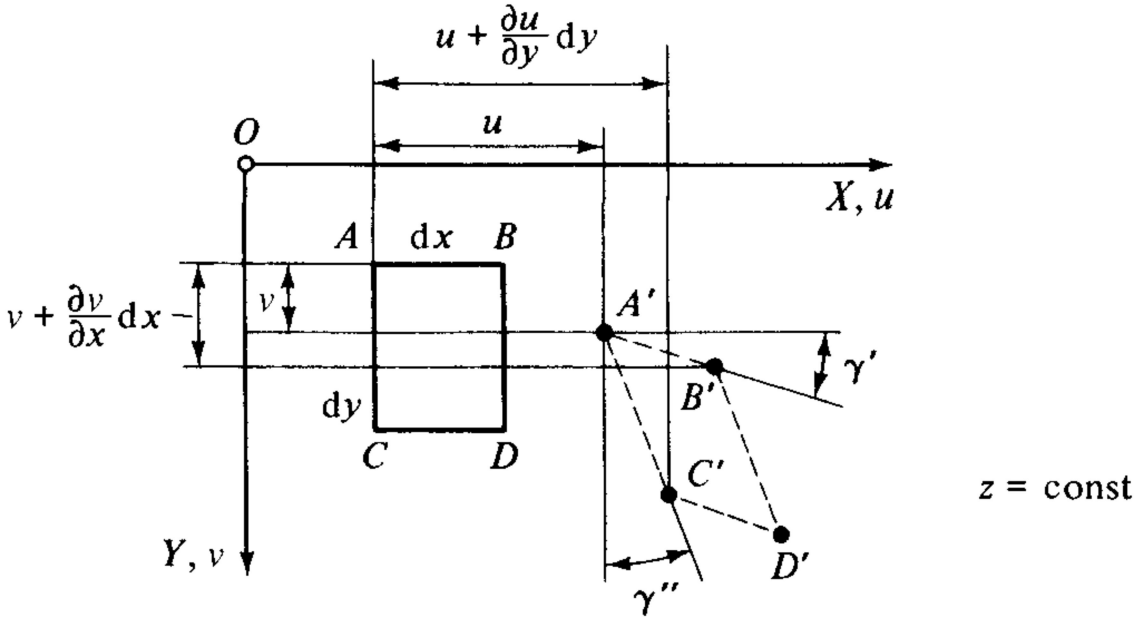


Figura 1.2-3. Distorção angular. Fonte: (Szilard, 2004)

$$\gamma'' = \frac{\partial u}{\partial y}, \text{ mas} \tag{1.14}$$

$$u = zv = -z\frac{\partial w}{\partial x}, \text{ similarmente} \tag{1.15}$$

$$v = -z\frac{\partial w}{\partial y}, \text{ consequentemente} \tag{1.16}$$

$$\gamma_{xy} = \gamma' + \gamma'' = -2z\frac{\partial^2 w}{\partial x \partial y} \tag{1.17}$$

As mudanças na curvatura da superfície média são definidas por:

$$\kappa_x = \frac{-\partial^2 w}{\partial x^2} \tag{1.18}$$

$$\kappa_y = -\frac{\partial^2 w}{\partial y^2} \tag{1.19}$$

$$\chi = -\frac{\partial^2 w}{\partial x \partial y} \tag{1.20}$$

onde χ representa o empenamento da placa.

c) Forças internas expressas em termos de w:

As componentes de tensão σ_x e σ_y produzem momentos fletores no elemento de placa de maneira similar àquelas da teoria da viga elementar. Portanto, por integração das componentes de tensões normais, os momentos fletores, que agem no elemento de placa, são obtidos:

$$m_x = \int_{-h/2}^{+h/2} \sigma_x z\, dz \tag{1.21}$$

$$m_y = \int_{-h/2}^{+h/2} \sigma_y z\, dz \tag{1.22}$$

De maneira similar, os momentos torçores produzidos pelas tensões de cisalhamento $\tau_{xy} = \tau_{yx}$ podem ser calculados de:

$$m_{xy} = \int_{-h/2}^{+h/2} \tau_{xy} z\, dz \tag{1.23}$$

$$m_{yx} = \int_{-h/2}^{+h/2} \tau_{yx} z \, dz \tag{1.24}$$

mas $\tau_{xy} = \tau_{yx}$, dessa forma: $m_{xy} = m_{yx}$.

Se substituirmos, (1.11) e (1.12) em (1.7) e (1.8) as tensões normais σ_x e σ_y são expressos em termos de deflexão lateral **w**. Portanto, podemos escrever:

$$\sigma_x = -\frac{Ez}{1-\nu^2}\left(\frac{\partial^2 w}{\partial x^2} + \nu \frac{\partial^2 w}{\partial y^2}\right) \tag{1.25}$$

$$\sigma_y = -\frac{Ez}{1-\nu^2}\left(\frac{\partial^2 w}{\partial y^2} + \nu \frac{\partial^2 w}{\partial x^2}\right) \tag{1.26}$$

Integrando as equações (1.21) e (1.22), e substituindo σ_x e σ_y pelas equações acima obtemos:

$$\begin{aligned} m_x &= -\frac{Eh^3}{12\left(1-\nu^2\right)}\left(\frac{\partial^2 w}{\partial x^2} + \nu \frac{\partial^2 w}{\partial y^2}\right) \\ &= -D\left(\frac{\partial^2 w}{\partial x^2} + \nu \frac{\partial^2 w}{\partial y^2}\right) = D\left(\kappa_x + \nu \kappa_y\right), \text{ e} \end{aligned} \tag{1.27}$$

$$m_y = -D\left(\frac{\partial^2 w}{\partial x^2} + \nu \frac{\partial^2 w}{\partial y^2}\right) = D\left(\kappa_y + \nu \kappa_x\right), \text{ onde} \tag{1.28}$$

$$\boxed{D = \frac{Eh^3}{12\left(1-\nu^2\right)}}, \text{ representa a rigidez à flexão da placa.} \tag{1.29}$$

De maneira similar, a expressão de momento torçor em termos de deflexões laterais é obtida da seguinte forma:

$$\begin{aligned} m_{xy} = m_{yx} &= \int_{-h/2}^{+h/2} \tau z \, dz = -2G \int_{-h/2}^{+h/2} \frac{\partial^2 w}{\partial x \partial y} z^2 \, dz \\ &= -\left(1-\nu\right) D \frac{\partial^2 w}{\partial x \partial y} = D\left(1-\nu\right)\chi \end{aligned} \tag{1.30}$$

A substituição das equações (1.27), (1.28) e (1.30) na equação (1.4) resulta na equação diferencial governante da placa submetida a carregamentos laterais distribuídos:

$$\boxed{\frac{\partial^4 w}{\partial x^4}+\frac{\partial^4 w}{\partial y^2}+2\frac{\partial^4 w}{\partial x^2 \partial y^2}=\frac{-p_z\left(x,y\right)}{D}}, \tag{1.31}$$

ou utilizando o operador laplaciano bidimensional:

$$\nabla^2\left(\bullet\right)=\frac{\partial^2\left(\bullet\right)}{\partial x^2}+\frac{\partial^2\left(\bullet\right)}{\partial y^2} \tag{1.32}$$

ou de forma mais condensada:

$$D\nabla^2\nabla^2 w\left(x,y\right)=p_z\left(x,y\right) \tag{1.33}$$

Essa equação diferencial de quarta ordem, não homogênea, elíptica com coeficientes constantes é geralmente chamada de equação biarmônica não-homogênea. A equação (1.31) é linear pelo fato de que as derivadas de $w(x,y)$ não têm expoentes maiores que um.

Podemos expressar também os esforços cortantes em termos de deflexões laterais. A substituição das equações (1.27), (1.28) e (1.30) nas equações (1.1) e (1.2) resulta:

$$\begin{aligned} q_x &= \frac{\partial m_x}{\partial x}+\frac{\partial m_{yx}}{\partial y}=-D\frac{\partial}{\partial x}\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}\right)=-D\frac{\partial}{\partial x}\nabla^2 w \\ q_y &= \frac{\partial m_y}{\partial y}+\frac{\partial m_{xy}}{\partial x}=-D\frac{\partial}{\partial y}\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}\right)=-D\frac{\partial}{\partial y}\nabla^2 w \end{aligned} \tag{1.34}$$

O problema da placa é considerado resolvido se uma expressão adequada para a superfície defletida da placa $w(x,y)$ é encontrada que satisfaça simultaneamente a equação diferencial de equilíbrio e as condições de contorno. Consequentemente, pode-se dizer que a solução dos problemas de placas é um caso específico de problemas de valor de contorno em física-matemática. (Szilard, 2004)

No sistema de coordenadas X, Y, a matriz de momentos internos em um ponto conhecido é escrito na forma:

$$M = \begin{bmatrix} m_x & m_{xy} \\ m_{yx} & m_y \end{bmatrix} \text{ onde } m_{xy} = m_{yx}. \tag{1.35}$$

Se desejarmos obter esses momentos em qualquer outro plano transversal, devemos utilizar uma transformação de coordenadas do sistema de coordenadas X, Y para X', Y'. Portanto:

$$M' = \begin{bmatrix} m'_x & m'_{xy} \\ m'_{yx} & m'_y \end{bmatrix} = RMR^T \tag{1.36}$$

onde R representa a matriz de rotação:

$$R = \begin{bmatrix} cos\phi & sen\phi \\ -sen\phi & cos\phi \end{bmatrix} = \begin{bmatrix} C & S \\ -S & C \end{bmatrix} \tag{1.37}$$

Dessa forma, obtemos:

$$M' = \begin{bmatrix} m_x C^2 + m_y S^2 + 2m_{xy} SC & m_{xy}\left(C^2 - S^2\right) + \left(m_y - m_x\right) SC \\ simétrico & m_x S^2 + m_y C^2 - 2m_{xy} SC \end{bmatrix} \tag{1.38}$$

Em resumo: foi mostrado que a montagem da equação diferencial de uma placa elástica seguiu necessariamente os seguintes passos:

1. Expressão de equilíbrio de forças externas e internas agindo num elemento de placa;
2. Relação entre tensões e deformações pela lei de Hooke;
3. Uso de certas identidades trigonométricas, obtidas através da forma da superfície defletida, o que permite expressar tensões em termos de componentes de deslocamento;
4. Expressão de forças internas em termos de tensões e deformações e finalmente em termos de componentes de deslocamento.

Problemas de placas, assim como todos os problemas da elasticidade de tensões bi e tridimensionais, são estaticamente indeterminados internamente; isto é, as três equações de equilíbrio (1.1), (1.2), e (1.4) possuem cinco variáveis: três momentos e duas forças cortantes. Para obter uma solução, equações da elasticidade adicionais

foram utilizadas. Desta forma, a equação diferencial governante de placas carregadas lateralmente (1.31), que contém apenas uma única variável foi definida.

1.3 Condições de contorno da teoria das placas de Kirchhoff

Uma solução exata da equação governante (1.31) precisa satisfazer simultaneamente a equação diferencial e as condições de contorno de qualquer problema de placa (Szilard, 2004). Já que a equação (1.31) é uma equação diferencial de quarta ordem, duas condições de contorno, seja para deslocamentos ou para forças internas, são necessárias para cada contorno. Na teoria das placas, três forças internas são consideradas: momento fletor, momento torçor e componentes de cisalhamento transverso. Similarmente, as componentes de deslocamento a serem usadas na formulação das condições de contorno são as deflexões laterais e as rotações. Condições de contorno de placas à flexão podem ser classificadas, em geral, em uma das definições a seguir:

a) Condições de contorno geométricas:

Certas condições geométricas fornecidas pela magnitude dos deslocamentos (translação e rotação) podem ser usadas para formular as condições de contorno na forma matemática. Nas bordas fixas, por exemplo, a deflexão e a rotação da placa defletida são zero. Portanto, escrevemos:

$$\begin{aligned} &\left(w\right)_x = 0,\ \left(\frac{\partial w}{\partial x}\right)_x = 0 \text{ em } x = 0, a; \text{ e} \\ &\left(w\right)_y = 0,\ \left(\frac{\partial w}{\partial y}\right)_y = 0 \text{ em } y = 0, b. \end{aligned} \tag{1.39}$$

Tais condições de contorno são chamadas “geométricas”.

b) Condições de contorno estáticas (bordas livres):

Para condições de contorno estáticas as forças nas bordas fornecem as expressões matemáticas procuradas. Em uma borda livre sem carregamento, por exemplo, podemos dizer que o momento na borda e o esforço cortante são zero, o que resulta:

$$\begin{aligned} &\left(m_x\right)_x = \left(v_x\right)_x = 0 \text{ em } x = 0, a; \text{ ou} \\ &\left(m_y\right)_y = \left(v_y\right)_y = 0 \text{ em } y = 0, b. \end{aligned} \tag{1.40}$$

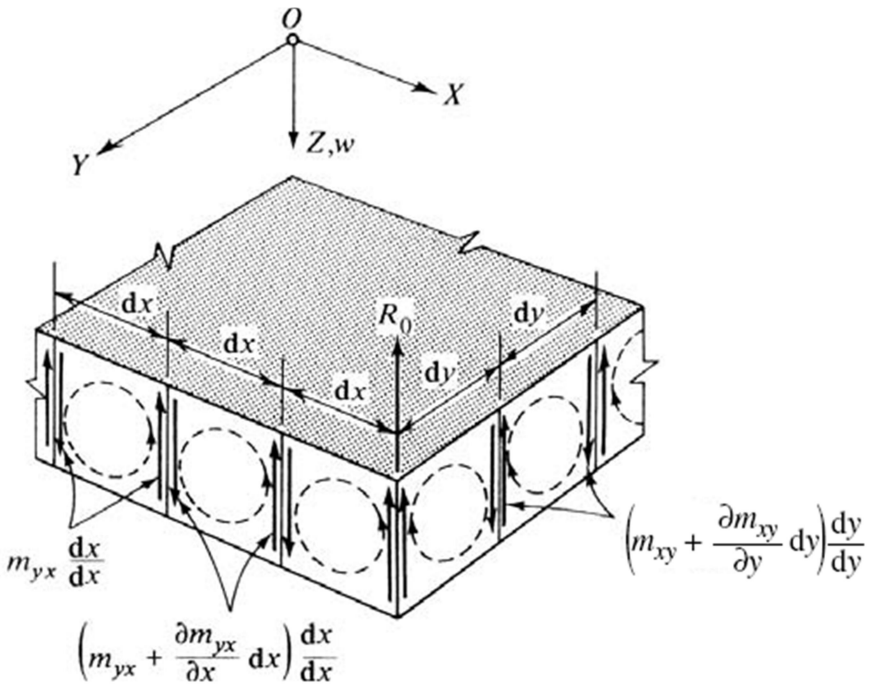


Figura 1.3-1. Efeito dos momentos torçores de borda. Fonte: (Szilard, 2004)

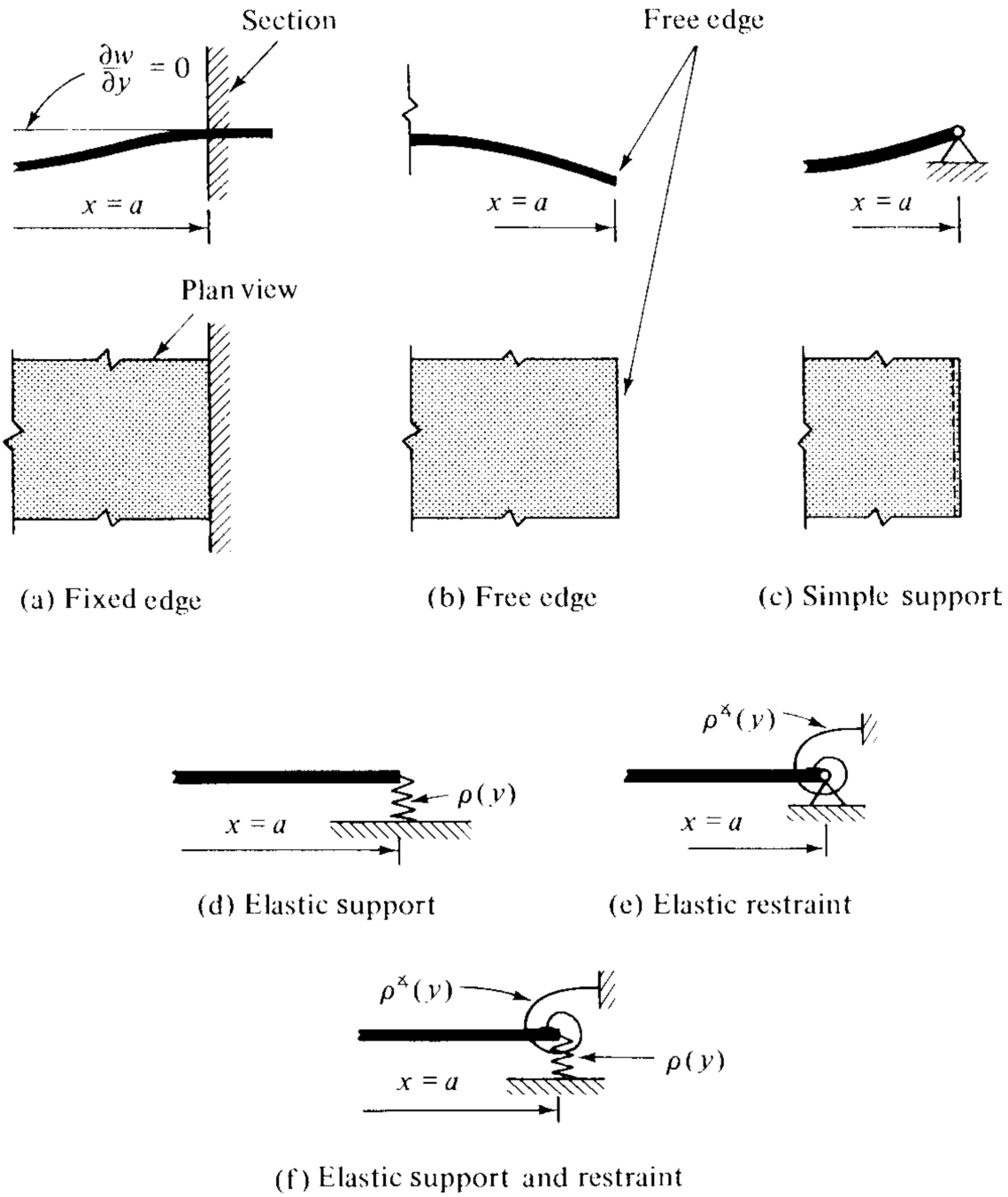


Figura 1.3-2. Esquematização de algumas condições de contorno. Fonte: (Szilard, 2004)

A força cortante na borda da placa consiste de dois termos, isto é, cisalhamento transverso e o efeito do momento torçor. Considerando as normais das bordas das placas nas direções X e Y, respectivamente, as forças verticais de borda por unidade de comprimento podem ser escritas como:

$$v_x = q_x + \frac{\partial m_{xy}}{\partial y} = -D\left[\frac{\partial^3 w}{\partial x^3} + (2-\nu)\frac{\partial^3 w}{\partial x \partial y^2}\right] \text{ em } x = 0, a,$$
$$v_y = q_y + \frac{\partial m_{yx}}{\partial x} = -D\left[\frac{\partial^3 w}{\partial y^3} + (2-\nu)\frac{\partial^3 w}{\partial x^2 \partial y}\right] \text{ em y} = 0, b. \tag{1.41}$$

onde q_x e q_y são as forças cortantes laterais (eq. (1.34)). Os termos secundários, $\partial m_{xy}/\partial y$ e $\partial m_{yx}/\partial x$, na equação (1.41) representam forças de cisalhamento adicionais nas bordas, produzidas pelos momentos torçores $m_{xy} = m_{yx}$.

c) Condições de contorno mistas:

Uma borda simplesmente apoiada (Figura 1.3-2 (c)) representa condições de contorno mistas. Já que a deflexão e o momento fletor ao longo do contorno são zero, formulação para esse tipo de condição de contorno envolve afirmações em relação a forças e deslocamentos. Portanto:

$$(w)_x = 0,\ (m_x)_x = \left(\frac{\partial^2 w}{\partial x^2} + \nu\frac{\partial^2 w}{\partial y^2}\right)_x = 0 \text{ em } x = 0, a,$$
$$(w)_y = 0,\ (m_y)_y = \left(\frac{\partial^2 w}{\partial y^2} + \nu\frac{\partial^2 w}{\partial x^2}\right)_y = 0 \text{ em y} = 0, b. \tag{1.42}$$

Nos cantos de placas retangulares, as ações dos momentos torçores, discutidas anteriormente, se somam ao invés de se anularem (pois $m_{xy} = m_{yx}$), produzindo uma força de canto adicional:

$$R_0 = 2m_{xy} = -2D(1-\nu)\frac{\partial^2 w}{\partial x \partial y} \tag{1.43}$$

Se nenhum ancoramento for fornecido, essas forças podem levantar os cantos. Já que essa condição é geralmente indesejável, deve ser evitada fixando-se as bordas de placas simplesmente apoiadas. Para lajes de concreto armado, quando o levantamento dos cantos é evitado, armaduras especiais de canto são necessárias para eliminar falhas locais.

Quando duas bordas adjacentes são fixadas, a força de canto adicional é zero ($R_0 = 0$), já que não existe momentos torçores ao longo dessas bordas. O caso da interseção das bordas livres é similar (Szilard, 2004).

Se a solução de um determinado problema de placa $w(x, y)$ fosse obtida por um dos métodos analíticos ou numéricos, a determinação das forças de canto podem pedir considerações especiais (Szilard, 2004).

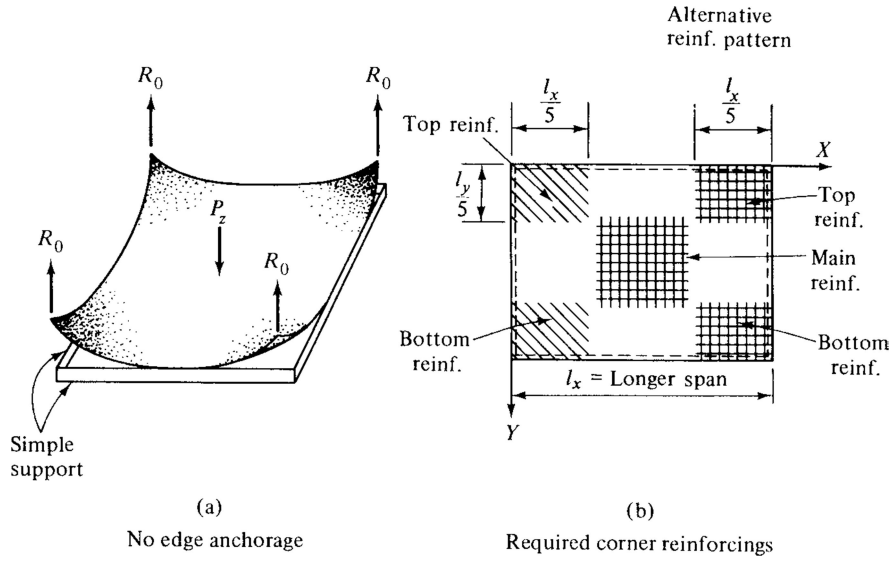


Figura 1.3-3. Levantamento dos cantos. Fonte: (Szilard, 2004)

2. Elemento finito ACM para placas de espessura fina

Obtendo as equações básicas que governam o comportamento de uma placa com espessura fina, podemos proceder na obtenção da expressão da matriz de rigidez, $[k^{(e)}]$, de um elemento de quatro nós denominado ACM (Adini-Clough-Melosh), e o vetor de forças externas nodais $\{f^{(e)}\}$, de tal maneira que se obedeça (Chaves & Minguez, 2010):

$$\left\{f^{(e)}\right\} = \left[k^{(e)}\right].\left\{f^{(e)}\right\} \tag{2.1}$$

2.1 Obtenção da matriz de rigidez do elemento

Os passos para obtenção da matriz de rigidez $[k^{(e)}]$ do elemento ACM para resolução de problemas de placas de espessura fina são (Chaves & Minguez, 2010):

Passo 1: Identificação do problema.

Para qualquer tipo de elemento selecionado, o primeiro passo é a seleção do sistema de coordenadas e a identificação tanto do número de nós do elemento (n_{ne}) quanto do número de graus de liberdade por nó (n_{gn}).

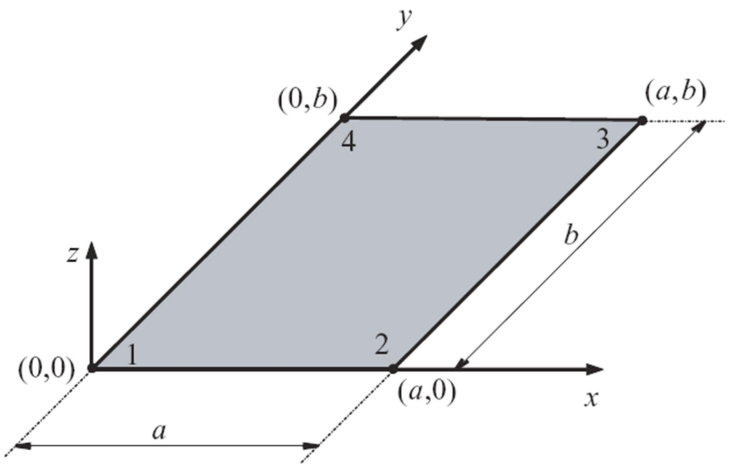


Figura 2.1-1. Elemento finito ACM para o estudo de placas finas. Fonte: (Chaves & Minguez, 2010)

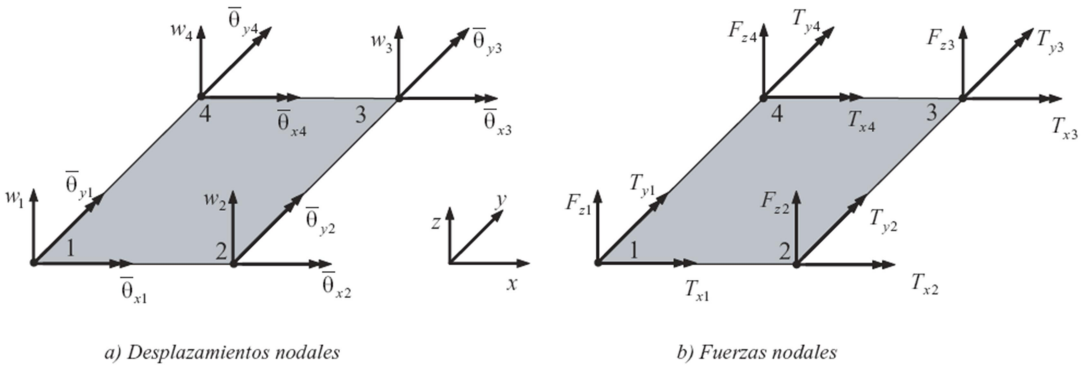


Figura 2.1-2. Componentes dos vetores de deslocamentos e de forças externas nodais do elemento de quatro nós (ACM). Fonte: (Chaves & Minguez, 2010)

Passo 2: Seleção da função de deslocamentos $\delta(x, y)$.

Dado que o elemento ACM possui 12 graus de liberdade no total, a função na qual se representa o campo de deslocamentos precisa ter 12 coeficientes associados aos 12 termos do polinômio de Pascal. Assim, o deslocamento vertical no elemento em função das coordenadas **x** e **y** fica na forma:

$$w = \begin{bmatrix} 1 & x & y & x^2 & xy & y^2 & x^3 & x^2y & xy^2 & y^3 & x^3y & xy^3 \end{bmatrix} . \{\alpha\} \qquad (2.2)$$

onde $\{\alpha\} = \{\alpha_1, \alpha_2, \dots, \alpha_{12}\}^T$ é um vetor com 12 constantes a determinar.

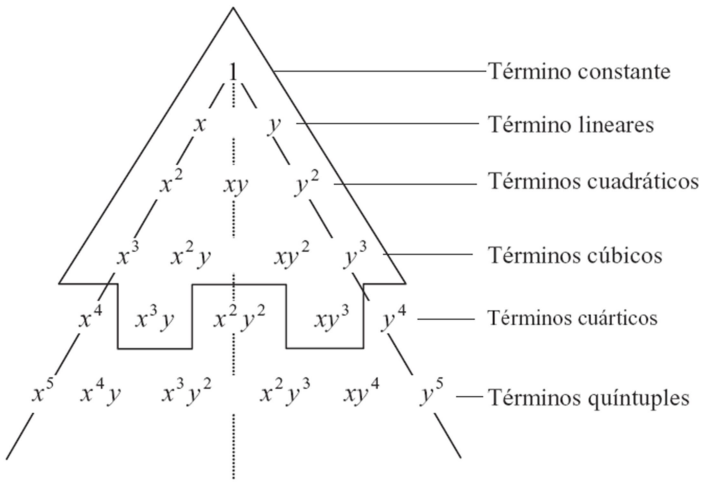


Figura 2.1-3. Polinômio de Pascal em duas dimensões. Fonte: (Chaves & Minguez, 2010)

Uma vez definido o deslocamento vertical no elemento, as rotações são obtidas empregando a expressão:

$$\{\delta\} = \begin{Bmatrix} w \\ \overline{\theta_x} = -\theta_y = -\left(-\dfrac{\partial w}{\partial y}\right) \\ \overline{\theta_y} = \theta_x = -\dfrac{\partial w}{\partial x} \end{Bmatrix} \tag{2.3}$$

com os quais se obtém:

$$\begin{aligned} \overline{\theta_x} &= \frac{\partial w}{\partial x} = \alpha_3 + \alpha_5 x + 2\alpha_6 y + \alpha_8 x^2 + 2\alpha_9 xy + 3\alpha_{10} y^2 + \alpha_{11} x^3 + 3\alpha_{12} xy^2 \\ \overline{\theta_y} &= -\frac{\partial w}{\partial y} = -\alpha_2 - 2\alpha_4 x - \alpha_5 y - 3\alpha_7 x^2 - 2\alpha_8 xy - \alpha_9 y^2 - 3\alpha_{11} x^2 y - \alpha_{12} y^3 \end{aligned} \tag{2.4}$$

Dispondo as expressões (2.2) e (2.4) em forma matricial, o vetor de deslocamentos $\{\delta(x, y)\}$ fica na forma:

$$\underbrace{\begin{Bmatrix} w \\ \overline{\theta_x} \\ \overline{\theta_y} \end{Bmatrix}}_{\{\delta(x,y)\}} = \underbrace{\begin{bmatrix} 1 & x & y & x^2 & xy & y^2 & x^3 & x^2y & xy^2 & y^3 & x^3y & xy^3 \\ 0 & 0 & 1 & 0 & x & 2y & 0 & x^2 & 2xy & 3y^2 & x^3 & 3xy^2 \\ 0 & -1 & 0 & -2x & -y & 0 & -3x^2 & -2xy & -y^2 & 0 & -3x^2y & -y^3 \end{bmatrix}}_{[X]} \{\alpha\}$$

$$\{\delta(x,y)\} = [X].\{\alpha\} \tag{2.5}$$

Passo 3: Representação dos deslocamentos dentro do elemento.

Nesta etapa, pretende-se representar os deslocamentos dentro do elemento, $\{\delta(x,y)\}$, em função dos deslocamentos nodais, $\{\delta^{(e)}\}$, isto é, que os parâmetros $\{\alpha\}$ da equação (2.5) permanecerão em função dos deslocamentos nodais:

$$\{\delta^{(e)}\} = [A].\{\alpha\} \tag{2.6}$$

A matriz [A] é obtida partindo de que a solução adotada (2.5) é válida para todos os pontos do elemento e particularizando a função dos deslocamentos nas coordenadas dos quatro nós do elemento ACM, com o que se obtém:

$$[A] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & a & 0 & a^2 & 0 & 0 & a^3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & a & 0 & 0 & a^2 & 0 & 0 & a^3 & 0 \\ 0 & -1 & 0 & -2a & 0 & 0 & -3a^2 & 0 & 0 & 0 & 0 & 0 \\ 1 & a & b & a^2 & ab & b^2 & a^3 & a^2b & ab^2 & b^3 & a^3b & ab^3 \\ 0 & 0 & 1 & 0 & a & 2b & 0 & a^2 & 2ab & 3b^2 & a^3 & 3ab^2 \\ 0 & -1 & 0 & -2a & -b & 0 & -3a^2 & -2ab & -b^2 & 0 & -3a^2b & -b^3 \\ 1 & 0 & b & 0 & 0 & b^2 & 0 & 0 & 0 & b^3 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2b & 0 & 0 & 0 & 3b^2 & 0 & 0 \\ 0 & -1 & 0 & 0 & -b & 0 & 0 & 0 & -b^2 & 0 & 0 & -b^3 \end{bmatrix} \tag{2.7}$$

A forma inversa da expressão (2.6) é:

$$\{\alpha\} = [A]^{-1}.\{\delta^{(e)}\} \tag{2.8}$$

Substituindo (2.8) em (2.5), pode-se obter a expressão do deslocamento dentro do elemento em função dos deslocamentos nodais:

$$\left\{\delta(x,y)\right\} = \left[X\right].\left[A\right]^{-1}.\left\{\delta^{(e)}\right\} \tag{2.9}$$

$$\left\{\delta(x,y)\right\} = \left[N\right].\left\{\delta^{(e)}\right\}, \text{ onde } \left[N\right] = \left[X\right].\left[A\right]^{-1}. \tag{2.10}$$

Passo 4: Estabelecimento da relação deformação-deslocamento ($\{\varepsilon(x,y)\} - \delta^{(e)}$) dentro do elemento.

Dado que as equações da elasticidade interveem nas deformações, é necessário relacionar, para um elemento dado, as deformações $\{\varepsilon(x,y)\}$ com os deslocamentos nodais $\{\delta^{(e)}\}$. As deformações segundo a teoria de Kirchhoff para placas com espessura fina estão relacionadas a seguir:

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = -z \begin{Bmatrix} \dfrac{\partial^2 w}{\partial x^2} \\ \dfrac{\partial^2 w}{\partial y^2} \\ 2\dfrac{\partial^2 w}{\partial x \partial y} \end{Bmatrix} = -z.\left\{\hat{\varepsilon}\right\} \tag{2.11}$$

Substituindo o campo de deslocamentos (w) adotado (eq. (2.2)), em (2.11) obtém-se o vetor de deformações independentes:

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = -z \begin{Bmatrix} \left(2\alpha_4 + 6\alpha_7 x + 2\alpha_8 y + 6\alpha_{11} xy\right) \\ \left(2\alpha_6 + 2\alpha_9 x + 6\alpha_{10} y + 6\alpha_{12} xy\right) \\ 2\left(\alpha_5 + 2\alpha_8 x + 2\alpha_9 y + 3\alpha_{11} x^2 + 3\alpha_{12} y^2\right) \end{Bmatrix} \tag{2.12}$$

que expresso na forma matricial fica:

$$\left\{\varepsilon(x,y)\right\} = -z\begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 0 & 6x & 2y & 0 & 0 & 6xy & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2x & 6y & 0 & 6xy \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 4x & 4y & 0 & 6x^2 & 6y^2 \end{bmatrix}.\{\alpha\} = -z.\left[G\right].\{\alpha\} \tag{2.13}$$

Substituindo (2.8) na expressão (2.13) obtém-se:

$$\left\{\varepsilon(x,y)\right\} = -z\left[G\right]\left[A\right]^{-1}\left\{\delta^{(e)}\right\} = \left[B\right]\left\{\delta^{(e)}\right\}, \tag{2.14}$$

expressão que nos permite relacionar as deformações dentro do elemento em função dos deslocamentos nodais.

Também define-se o seguinte vetor que será útil na hora de definir os momentos:

$$\left\{\hat{\varepsilon}\right\} = \begin{Bmatrix} \dfrac{\partial^2 w}{\partial x} \\ \dfrac{\partial^2 w}{\partial y^2} \\ 2\dfrac{\partial^2 w}{\partial x \partial y} \end{Bmatrix} = \left[G\right]\left[A\right]^{-1}\left\{\delta^{(e)}\right\} \tag{2.15}$$

Passo 5: Estabelecimento da relação momentos fletores – deslocamentos nodais ($\{M^{(e)}\} - \delta^{(e)}$) dentro do elemento.

Partindo da expressão de $\{\hat{\varepsilon}\}$ dada por (2.15) e das relações conhecidas entre momentos fletores e torçores da teoria das placas:

$$\left\{M\right\} = \begin{Bmatrix} Mx \\ My \\ Mxy \end{Bmatrix} = -D\underbrace{\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \dfrac{1-\nu}{2} \end{bmatrix}}_{\left[Dk\right]}\underbrace{\begin{Bmatrix} \dfrac{\partial^2 w}{\partial x^2} \\ \dfrac{\partial^2 w}{\partial y^2} \\ 2\dfrac{\partial^2 w}{\partial x \partial y} \end{Bmatrix}}_{\left\{\hat{\varepsilon}\right\}} \tag{2.16}$$

$$\left\{M\right\} = -\left[Dk\right]\left\{\hat{\varepsilon}\right\} = -\left[Dk\right]\left[G\right]\left[A\right]^{-1}\left\{\delta^{(e)}\right\} \tag{2.17}$$

onde $[Dk]$ é a matriz constitutiva generalizada à flexão e $\{\hat{\varepsilon}\}$ é o vetor de deformações generalizadas à flexão ou vetor de curvaturas.

Passo 6: Estabelecimento da relação forças – deslocamentos nodais.

O último passo para obter a matriz de rigidez do elemento ACM consiste em estabelecer a relação entre a força $\{f^{(e)}\}$ e o deslocamento nodal $\{\delta^{(e)}\}$. Essa relação vem dada pela matriz de rigidez do elemento $\{k^{(e)}\}$.

Aplicando o Princípio dos Trabalhos Virtuais, supõem-se uns deslocamentos virtuais $\{\bar{\delta}^{(e)}\}$ nos nós do elemento de forma que o trabalho externo total, dado pela expressão:

$$W_{ext} = \left\{\bar{\delta}^{(e)}\right\}^T \left\{f^{(e)}\right\} \tag{2.18}$$

seja igual ao trabalho interno total. Este último vem dado pela integral no volume do elemento:

$$\int_V \left\{\bar{\varepsilon}\right\}^T \{\sigma\}\, dV \tag{2.19}$$

onde $\{\bar{\varepsilon}\}$ é o vetor de deformações virtuais e $\{\sigma\}$ é o vetor tensor de tensões.

Se a expressão (2.14) é válida, também será válida para o vetor de deformações virtuais:

$$\left\{\bar{\varepsilon}(x,y)\right\} = -z\left\{\bar{\hat{\varepsilon}}(x,y)\right\} \text{ com } \left\{\bar{\hat{\varepsilon}}(x,y)\right\} = \left[G\right]\left[A\right]^{-1}\left\{\delta^{(e)}\right\} \tag{2.20}$$

Substituindo a expressão anterior no trabalho interno (2.19) obtemos:

$$\begin{aligned} &\int_V -z\left\{\delta^{(e)}\right\}^T \left[A\right]^{-T}\left[G\right]^T \{\sigma\}\, dV \\ &\int_A \left[\left\{\delta^{(e)}\right\}^T \left[A\right]^{-T}\left[G\right]^T \left(\int_z -z\{\sigma\}\, dz\right)\right] dA \end{aligned} \tag{2.21}$$

Levando em conta que:

$$Mx = \int_{-\frac{t}{2}}^{\frac{t}{2}} z\sigma_x dz$$

$$My = \int_{-\frac{t}{2}}^{\frac{t}{2}} z\sigma_y dz \tag{2.22}$$

$$Mxy = \int_{-\frac{t}{2}}^{\frac{t}{2}} z\tau_{xy} dz$$

e a partir de (2.17), podemos dizer que:

$$\int_z -z\{\sigma\}dz = -\{M\} = -\left(-\left[Dk\right]\{\hat{\varepsilon}\}\right) = \left[Dk\right]\left[G\right]\left[A\right]^{-1}\left\{\delta^{(e)}\right\} \tag{2.23}$$

resultando que o trabalho interno vem dado por:

$$\mathrm{W}_{int} = \left\{\bar{\delta}^{(e)}\right\}^T \int_A \left[\left[A\right]^{-T}\left[G\right]^T\left[Dk\right]\left[G\right]\left[A\right]^{-1}\right] dA \left\{\delta^{(e)}\right\} \tag{2.24}$$

Igualando o trabalho externo W_{ext} (2.18) com o trabalho interno W_{int} (2.24), se chega à seguinte expressão:

$$\left\{\bar{\delta}^{(e)}\right\}^T \left\{f^{(e)}\right\} = \left\{\bar{\delta}^{(e)}\right\}^T \int_0^b \int_0^a \left[\left[A\right]^{-T}\left[G\right]^T\left[Dk\right]\left[G\right]\left[A\right]^{-1}\right] dxdy \left\{\delta^{(e)}\right\} \tag{2.25}$$

que simplificando fica:

$$\left\{f^{(e)}\right\} = \int_0^b \int_0^a \left[A\right]^{-T}\left[G\right]^T\left[Dk\right]\left[G\right]\left[A\right]^{-1} dxdy \left\{\delta^{(e)}\right\} \tag{2.26}$$

Identificando os termos desta expressão com a equação (2.1) deduz-se que a matriz de rigidez do elemento ACM é igual a:

$$\left[k^{(e)}\right] = \int_0^b \int_0^a \left[A\right]^{-T}\left[G\right]^T\left[Dk\right]\left[G\right]\left[A\right]^{-1} dxdy \tag{2.27}$$

A expressão analítica da matriz de rigidez pode ser obtida por integração numérica utilizando dois pontos de Gauss por dimensão (Chaves & Minguez, 2010).

2.2 Vetor de forças nodais

Para o vetor de forças nodais externas $\{f^{(e)}\}$, considera-se apenas uma carga uniformemente distribuída. Neste caso em particular, o vetor de forças nodais equivalente $\{f^{(e)}\}$ vem dado por (Chaves & Minguez, 2010):

$$\left\{f^{(e)}\right\} = \int_A \left[N\right]^T \left\{P\right\} dA \tag{2.28}$$

onde $\{P\}$ é o vetor com as forças no elemento.

Considerando a expressão (2.10) pode-se dizer que:

$$\begin{aligned} \left\{f^{(e)}\right\} &= \int_A \left[N\right]^T \left\{P\right\} dA \\ &= \int_A \left(\left[X\right]\left[A\right]^{-1}\right)^T \left\{P\right\} dA \\ &= \int_A \left[A\right]^{-T} \left[X\right]^T \left\{P\right\} dA \end{aligned} \tag{2.29}$$

Sendo a matriz $[A]^{-1}$ constante em um elemento de placa, conclui-se que:

$$\left\{f^{(e)}\right\} = \left[A\right]^{-T} \int_A \left[X\right]^T \left\{P\right\} dA \tag{2.30}$$

O vetor $\{P\}$ depende do tipo de carga que se adota no elemento, podendo ser variável ou não dentro do mesmo. Neste caso o vetor de cargas é dado por:

$$\left\{P\right\} = \begin{Bmatrix} q \\ 0 \\ 0 \end{Bmatrix} \tag{2.31}$$

Tendo em conta as expressões de $[X]$ (2.5), $\{P\}$ (2.31) e a matriz $[A]^{-1}$, obtém-se o vetor de forças nodais externas:

$$\left\{f^{(e)}\right\}=\begin{Bmatrix}F_{x1}\\T_{x1}\\T_{y1}\\F_{x2}\\T_{x2}\\T_{y2}\\F_{x3}\\T_{x3}\\T_{y3}\\F_{x4}\\T_{x4}\\T_{y4}\end{Bmatrix}=\frac{abq}{4}\begin{Bmatrix}1\\\frac{1}{6}b\\-\frac{1}{6}a\\1\\\frac{1}{6}b\\\frac{1}{6}a\\1\\-\frac{1}{6}b\\\frac{1}{6}a\\1\\-\frac{1}{6}b\\-\frac{1}{6}a\end{Bmatrix} \tag{2.32}$$

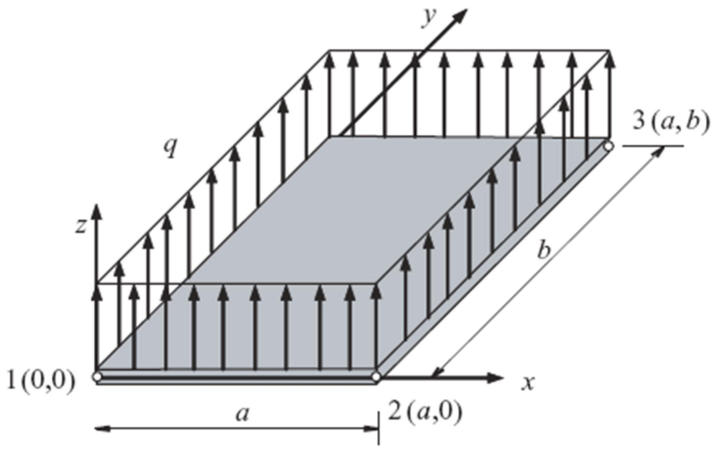


Figura 2.1-4. Carga distribuída considerada no elemento finito de placa. Fonte: (Chaves & Minguez, 2010)

3. Placas grossas, teoria de Reissner-Mindlin:

Os elementos de placa de Kirchhoff são restringidos apenas a situações de placas finas (espessura / média dos lados $\leq$ 0,10). Além disso, a condição de continuidade C^1 para elementos de Kirchhoff causa severas dificuldades para derivar

um campo de deflexões conformante (Oñate, 2013). Esses problemas podem ser resolvidos utilizando a formulação de placa devido a Reissner e Mindlin.

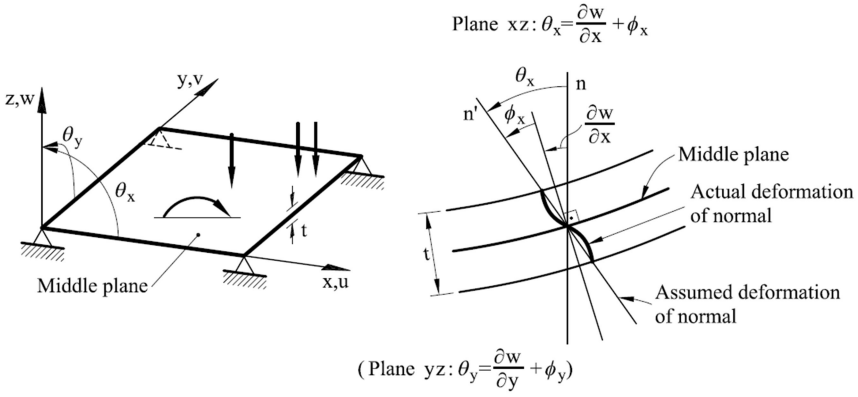


Figura 3-1. Convenção de sinais para os deslocamentos e as rotações da normal. Fonte: (Oñate, 2013)

A tão chamada teoria de placas de Mindlin assume que as normais da placa não são mantidas ortogonais ao plano médio após a deflexão, permitindo, então, efeitos de deformação transversa do cisalhamento. Esta hipótese é análoga àquela feita para rotação da transversa da seção transversal na teoria da viga de Timoshenko. Isto nos permite utilizar elementos contínuos C^0. Infelizmente, algumas dificuldades surgem quando elementos de Reissner-Mindlin são utilizados para situações para placas finas devido à influência excessiva dos termos de deformação do cisalhamento transverso. O efeito "shear locking" é análogo ao apresentado quando elementos de viga de Timoshenko são aplicados a vigas esbeltas. Eliminação do *shear locking* é possível com integração reduzida, interpolações ligadas ou campos de deformação-tensão impostos.

Elementos de placa Reissner-Mindlin podem ser tomados como ponto de partida para derivação de elementos de placa fina contínua C^0 contornando adequadamente a deformação transversa do cisalhamento para zero em pontos selecionados do elemento.

A teoria de Reissner-Mindlin é muito adequada para estudar composições de placas laminadas para os quais efeitos de deformação por cisalhamento são importantes, e também pode ser estendida para análise de cascas (Oñate, 2013).

A simplicidade dos elementos de placa de Reissner-Mindlin e sua versatilidade para análise de placas grossas e finas com materiais homogêneos e compostos contribuíram para sua popularidade em aplicações práticas (Oñate, 2013).

3.1 Teoria das placas de Reissner-Mindlin

A teoria da flexão de placas compartilha as três primeiras hipóteses da teoria das placas de Kirchhoff. A hipótese da rotação da normal é diferente e lê-se o seguinte:

"Uma linha reta normal à superfície não-defletida do plano médio permanece reta mas não necessariamente ortogonal ao plano médio após a deflexão".

Esta hipótese tem analogia com a hipótese da rotação da seção transversal nas vigas de Timoshenko e, de fato, existem muitas características comuns entre ambas as teorias das placas e das vigas.

3.1.1 Campo de deslocamento

O campo de deslocamento tridimensional é expresso em termos das variáveis cinemáticas do plano médio w, θ_x e θ_y como:

$$\begin{aligned} u(x,y,z) &= -z\theta_x(x,y) \\ v(x,y,z) &= -z\theta_y(x,y) \\ w(x,y,z) &= w(x,y) \end{aligned} \tag{3.1}$$

onde θ_x e θ_y são os ângulos definindo a rotação do vetor normal. A equação (3.1) é idêntica à equação da teoria de Kirchhoff. Aqui o plano médio é tomado também com o plano de referência ($z = 0$). O vetor de deslocamento é:

$$u = \begin{bmatrix} w & \theta_x & \theta_y \end{bmatrix}^T \tag{3.2}$$

A hipótese anterior da rotação nos permite expressar a rotação da normal no plano xz:

$$\theta_x = \frac{\partial w}{\partial x} + \phi_x \tag{3.3}$$

similarmente para o plano yz:

$$\theta_y = \frac{\partial w}{\partial y} + \phi_y \tag{3.4}$$

A rotação da normal em cada um dos dois planos verticais xz e yz é obtida como a soma de termos: 1) a deflexão adequada do plano médio da placa, e 2) uma rotação adicional ϕ resultante da falta de ortogonalidade da normal com o plano médio

após a deformação. Consequentemente, as rotações θ_x e θ_y não podem ser computadas em termos da deflexão apenas e, portanto, são tratadas como *variáveis independentes*. Essa é uma diferença essencial entre as teorias de Kirchhoff e Reissner-Mindlin.

A hipótese de normais retas é uma aproximação da verdadeira cinemática das placas que na realidade as normais da placa são distorcidas durante a deformação. Claramente este efeito é mais importante em placas grossas. Os ângulos θ_x e θ_y podem ser interpretados como a rotação da linha reta representando a deformação "média" da normal (Oñate, 2013).

3.1.2 Campos de tensão e deformação

Substituindo o campo de deslocamento (3.1) na expressão para tensões num sólido tridimensional resulta:

$$\begin{aligned}
\varepsilon_x &= \frac{\partial u}{\partial x} = -z\frac{\partial \theta_x}{\partial x} \\
\varepsilon_y &= \frac{\partial v}{\partial y} = -z\frac{\partial \theta_y}{\partial y} \\
\varepsilon_z &= \frac{\partial w}{\partial z} = 0 \\
\gamma_{xy} &= \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = -z\left(\frac{\partial \theta_x}{\partial y} + \frac{\partial \theta_y}{\partial x}\right) \\
\gamma_{xz} &= \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} = -\theta_x + \frac{\partial w}{\partial x} = -\phi_x \\
\gamma_{yz} &= \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} = -\theta_y + \frac{\partial w}{\partial y} = -\phi_y
\end{aligned} \tag{3.5}$$

A não-ortogonalidade do vetor normal induz *deformações de cisalhamento transverso não-nulos* γ_{xz} e γ_{yz}, nos quais coincidem (em valor absoluto) com as rotações ϕ_x e ϕ_y. Essas deformações são independentes da coordenada **z**.

Admitindo zero nas deformações de cisalhamento transverso retornamos à condição de ortogonalidade da normal da teoria das placas de Kirchhoff já que $\theta_x = \frac{\partial w}{\partial x}$ e $\theta_y = \frac{\partial w}{\partial y}$.

O vetor de deformações contendo as tensões significantes é, portanto;

$$\varepsilon = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \\ \cdots \\ \gamma_{xz} \\ \gamma_{yz} \end{Bmatrix} = \begin{Bmatrix} -z\dfrac{\partial \theta_x}{\partial x} \\ -z\dfrac{\partial \theta_y}{\partial y} \\ -z\left(\dfrac{\partial \theta_x}{\partial y} + \dfrac{\partial \theta_y}{\partial x}\right) \\ \cdots \\ -\theta_x + \dfrac{\partial w}{\partial x} \\ -\theta_y + \dfrac{\partial w}{\partial y} \end{Bmatrix} = \begin{Bmatrix} \varepsilon_b \\ \cdots \\ \varepsilon_s \end{Bmatrix} \tag{3.6}$$

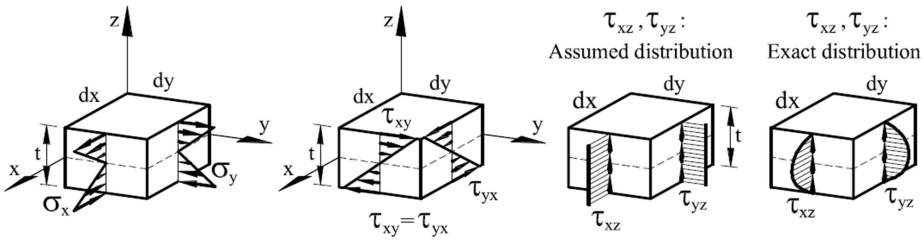


Figura 3.1.2-1. Convenções de sinais para tensões no plano σ_x, σ_y, τ_{xy} e as tensões de cisalhamento transverso τ_{xz} e τ_{yz}. Fonte: (Oñate, 2013)

onde os vetores ε_b e ε_s contêm as deformações de flexão e cisalhamento transverso, respectivamente. O vetor de deformações da eq. (3.6) pode ser expresso como:

$$\varepsilon = S\hat{\varepsilon} \tag{3.7}$$

onde

$$\hat{\varepsilon} = \begin{Bmatrix} \hat{\varepsilon}_b \\ \hat{\varepsilon}_s \end{Bmatrix} \tag{3.8}$$

é o vetor de deformações generalizado e $\hat{\varepsilon}_b$ e $\hat{\varepsilon}_s$ são o vetor de deformações por flexão generalizado e o vetor de deformações por cisalhamento transverso generalizado, respectivamente. A matriz **S** de transformação de deformação na eq. (3.7) é:

$$S=\begin{bmatrix} -z & 0 & 0 & 0 & 0 \\ 0 & -z & 0 & 0 & 0 \\ 0 & 0 & -z & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

O vetor de tensões é:

$$\sigma=\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \\ \cdots \\ \tau_{xz} \\ \tau_{yz} \end{Bmatrix}=\begin{Bmatrix} \sigma_b \\ \cdots \\ \sigma_s \end{Bmatrix} \tag{3.10}$$

onde σ_b e σ_s são as tensões devido à flexão pura e aos efeitos do cisalhamento transverso, respectivamente. Na teoria das placas σ_z foi omitido da eq. (3.10) pela hipótese de tensão no plano ($\sigma_z = 0$).

3.1.3 Relação tensão-deformação

Consideraremos a seguir apenas as condições do material ser homogêneo e isotérmico. Isso é consistente com a hipótese da teoria da flexão da placa, já que não há tensões ou forças axiais introduzidas durante a deformação da placa. Efeitos axiais aparecem para placas laminadas compostas ou quando efeitos térmicos são levados em consideração.

Começando pela equação constitutiva da elasticidade tridimensional e utilizando a hipótese de tensão no plano ($\sigma_z = 0$), podemos encontrar uma relação entre as tensões não-nulas e as deformações. Para um material ortotrópico com eixos de ortotropia 1,2,3 com $3 = z$ e satisfazendo a condição de anisotropia plana (ou seja, o plano 1,2 é um plano de simetria do material) podemos escrever:

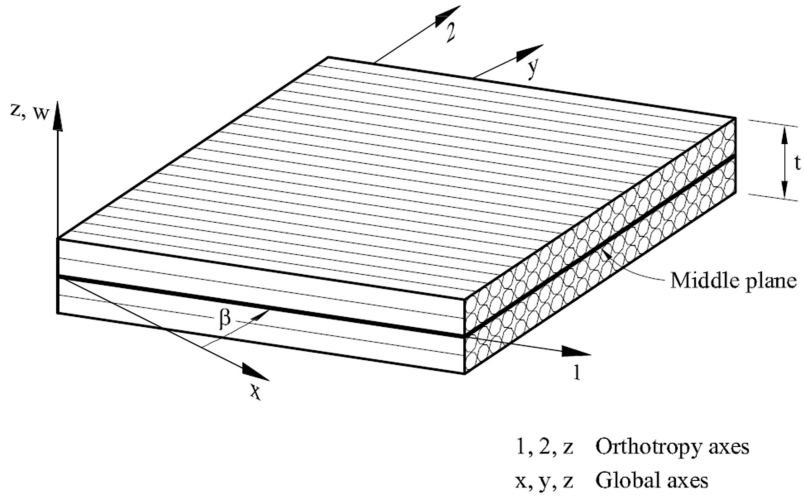


Figura 3.1.3-1. Material ortotrópico no plano 1-2 e isotrópico no plano 2-z. Fonte: (Oñate, 2013)

$$\sigma_I = \begin{Bmatrix} \sigma_1 \\ \sigma_2 \end{Bmatrix} = \begin{bmatrix} & & & 0 & 0 \\ & \underbrace{D_1}_{3\times 3} & & 0 & 0 \\ & & & 0 & 0 \\ 0 & 0 & 0 & & \\ 0 & 0 & 0 & \underbrace{D_2}_{2\times 2} & \end{bmatrix} \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{Bmatrix} = D_1 \varepsilon_1, \text{ onde} \tag{3.11}$$

$$\begin{aligned} \sigma_1 &= \begin{bmatrix} \sigma_1 & \sigma_2 & \tau_{12} \end{bmatrix}^T, \ \sigma_2 = \begin{bmatrix} \tau_{13} & \tau_{23} \end{bmatrix}^T \\ \varepsilon_1 &= \begin{bmatrix} \varepsilon_1 & \varepsilon_2 & \gamma_{12} \end{bmatrix}^T, \ \varepsilon_2 = \begin{bmatrix} \gamma_{13} & \gamma_{23} \end{bmatrix}^T \end{aligned} \tag{3.12}$$

são as tensões de flexão e cisalhamento transverso e deformações nos eixos principais de ortotropia e:

$$\begin{aligned} D_1 &= \frac{1}{1-\nu_{12}\nu_{21}} \begin{bmatrix} E_1 & \nu_{21}E_1 & 0 \\ \nu_{12}E_1 & E_2 & 0 \\ 0 & 0 & \left(1-\nu_{12}\nu_{21}\right)G_{12} \end{bmatrix} \\ D_2 &= \begin{bmatrix} G_{13} & 0 \\ 0 & G_{23} \end{bmatrix} \end{aligned} \tag{3.13}$$

Se existe isotropia na direção 1 (isto é, no plano 2-z) para compostos de fibra com fibras na direção 1 cobertas por uma matriz, então $G_{13} = G_{12}$.

As seguintes relações seguem:

$$\begin{aligned}
&\varepsilon_1 = T_1\varepsilon_b,\ \varepsilon_1 = T_1\varepsilon_b,\ \text{com}\\
&\sigma_b = T_1^T\sigma_1,\ \sigma_b = T_2{}^T\sigma_2\\
&T_1 = \begin{bmatrix} C^2 & S^2 & CS \\ S^2 & C^2 & -CS \\ -2CS & 2CS & C^2 - S^2 \end{bmatrix},\ T_2 = \begin{bmatrix} C & S \\ -S & C \end{bmatrix},
\end{aligned} \tag{3.14}$$

onde $C = cos\beta$, $S = sen\beta$ e β é o ângulo entre os eixos 1 e **x** e $\varepsilon_b, \varepsilon_s$ são definidos na equação (3.6).

A relação inversa pode ser obtida simplesmente substituindo β por $-\beta$ nas equações (3.14), isto é:

$$\begin{aligned}
&\varepsilon_b = \overline{T_1}\varepsilon_1,\ \varepsilon_s = \overline{T_2}\varepsilon_2,\ \text{com}\\
&\sigma_1 = \overline{T_1}^T\sigma_b,\ \sigma_2 = \overline{T_2}^T\sigma_s\\
&\overline{T_1} = \begin{bmatrix} C^2 & S^2 & -CS \\ S^2 & C^2 & CS \\ 2CS & -2CS & C^2 - S^2 \end{bmatrix},\ \overline{T_2} = \begin{bmatrix} C & -S \\ S & C \end{bmatrix}
\end{aligned} \tag{3.15}$$

Combinando as equações (3.11) e (3.14) resulta nas equações constitutivas para as tensões de flexão e de cisalhamento transverso nos eixos globais:

$$\begin{aligned}
&\sigma_b = D_b\varepsilon_b,\ \sigma_b = \overline{D_s}\varepsilon_s,\ \text{ou}\\
&\sigma = \begin{Bmatrix} \sigma_b \\ \sigma_s \end{Bmatrix} = D\begin{Bmatrix} \varepsilon_b \\ \varepsilon_s \end{Bmatrix} = D\varepsilon \text{ com D}= \begin{bmatrix} D_b & 0 \\ 0 & \overline{D_s} \end{bmatrix} \text{ e}\\
&D_b = T_1^T D_1 T_1,\ \overline{D_s} = T_2{}^T D_2 T_2
\end{aligned} \tag{3.16}$$

Para o caso de material isotrópico basta igualar $\beta = 0$, o resultado fica:

$$\begin{aligned}
D_b &= \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \dfrac{1-\nu}{2} \end{bmatrix}\\
D_s &= G\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\end{aligned} \tag{3.17}$$

Substituindo a equação (3.7) na equação (3.16) fornece a relação entre as tensões num ponto através da espessura e as deformações generalizadas.

$$\sigma = \begin{Bmatrix} \sigma_b \\ \sigma_s \end{Bmatrix} = DS\hat{\varepsilon} = D \begin{Bmatrix} -z\varepsilon_b \\ \hat{\varepsilon}_s \end{Bmatrix} \tag{3.18}$$

A equação (3.18) mostra que a distribuição das tensões à flexão "no plano" σ_x, σ_y e τ_{xy} varia linearmente com **z**, enquanto que as tensões de cisalhamento transverso τ_{xz} e τ_{yz} são constantes ao longo da espessura.

A distribuição "exata" da espessura das tensões de cisalhamento transverso obtida da teoria da elasticidade tridimensional não é uniforme e somem nas superfícies acima e abaixo da placa. Para material isotrópico a distribuição é parabólica. Esse problema, no qual também apareceu na teoria da viga de Timoshenko, pode ser resolvido escalando o trabalho interno associado às tensões de cisalhamento transverso de forma que coincida com o trabalho interno exato obtido da teoria da elasticidade tridimensional. Dessa forma, o trabalho de deformação total é computado exatamente, embora as tensões de cisalhamento transverso não tenham uma distribuição localmente correta na espessura (Oñate, 2013).

Na prática, isso implica a modificação da relação constitutiva do cisalhamento da primeira equação (3.16) como:

$$\sigma_s = D_s \varepsilon_s \text{ com } D_s = \begin{bmatrix} k_{11}\overline{D_{s_{11}}} & k_{12}\overline{D_{s_{12}}} \\ k_{12}\overline{D_{s_{12}}} & k_{22}\overline{D_{s_{22}}} \end{bmatrix}, \tag{3.19}$$

onde $D_{s_{ij}}$ são as componentes de $\overline{\overline{D_s}}$ da terceira equação (3.16) e k_{ij} são os parâmetros de correção do cisalhamento. Seus cômputos seguem um procedimento análogo ao explicado para vigas de Timoshenko. Para uma placa isotrópica $k_{12} = 0$ e $k_{11} = k_{22} = 5/6$, para vigas retangulares.

3.1.4 Tensões resultantes e matriz constitutiva generalizada

O vetor de tensões resultantes num ponto do plano médio da placa é definido como:

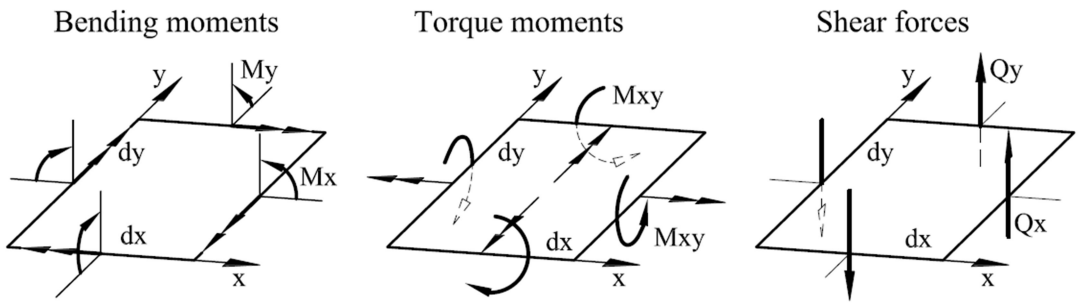


Figura 3.1.4-1. Convenção de sinais das tensões resultantes numa placa. Fonte: (Oñate, 2013)

$$\hat{\sigma}=\begin{Bmatrix}\hat{\sigma}_b\\ \cdots\\ \hat{\sigma}_s\end{Bmatrix}=\begin{Bmatrix}Mx\\ My\\ Mxy\\ \cdots\\ Qx\\ Qy\end{Bmatrix}=\int_{-\frac{t}{2}}^{+\frac{t}{2}}\begin{Bmatrix}-z\sigma_x\\ -z\sigma_y\\ -z\tau_{xy}\\ \cdots\\ \tau_{xz}\\ \tau_{yz}\end{Bmatrix}dz=$$
$$=\int_{-\frac{t}{2}}^{+\frac{t}{2}}\begin{Bmatrix}-z\sigma_b\\ \cdots\\ \sigma_s\end{Bmatrix}dz=\int_{-\frac{t}{2}}^{+\frac{t}{2}}S^T\sigma dz \tag{3.20}$$

onde S é a matriz de transformação da equação (3.9) e:

$$\hat{D}=\int_{-\frac{t}{2}}^{+\frac{t}{2}}S^T DS\,dz=\int_{-\frac{t}{2}}^{+\frac{t}{2}}\begin{bmatrix}z^2D_b & 0\\ 0 & D_s\end{bmatrix}dz=\begin{bmatrix}\hat{D}_b & 0\\ 0 & \hat{D}_s\end{bmatrix} \tag{3.21}$$

é a matriz constitutiva generalizada com D_b e D_s dados nas equações (3.16) e (3.19), respectivamente.

Para material isotrópico:

$$\hat{D}_b=\frac{t^3}{12}D_b$$
$$\hat{D}_s=tD_s=\frac{5}{6}tGI_2 \tag{3.22}$$

onde D_b é dado pela equação (3.17) e I_2 é a matriz identidade 2×2.

A equação (3.20) relaciona as tensões resultantes e as deformações generalizadas em qualquer ponto da superfície da placa. As tensões ao longo da

espessura podem ser obtidas a partir das deformações generalizadas combinando as equações (3.7) e (3.16).

$$\sigma = DS\hat{\varepsilon} \tag{3.23}$$

3.1.5 Princípio dos trabalhos virtuais

Consideremos uma placa carregada por carregamentos verticais distribuídos **t** e cargas pontuais p_i. A expressão do trabalho virtual é escrita como:

$$\iiint_V \delta\varepsilon^T \sigma \, dV = \iint_A \delta u^T t \, dA + \sum_i \delta u_i{}^T p_i, \text{ onde} \tag{3.24}$$

$$\begin{aligned} \delta u &= \left[\delta w, \delta\theta_x, \delta\theta_y \right]^T \\ t &= \left[f_z,\ m_x,\ m_y \right]^T \\ p_i &= \left[P_{z_i}, M_{x_i}, M_{y_i} \right]^T \end{aligned} \tag{3.25}$$

e δu é o vetor de deslocamento virtual, $\delta\varepsilon$ é o vetor de deformação virtual, f_z, m_x e m_y são o carregamento vertical distribuído e os momentos distribuídos agindo nos planos **xz** e **yz**, respectivamente. P_{z_i}, M_{x_i} e M_{y_i} são o carregamento pontual vertical e os momentos fletores concentrados agindo num ponto **i**, respectivamente.

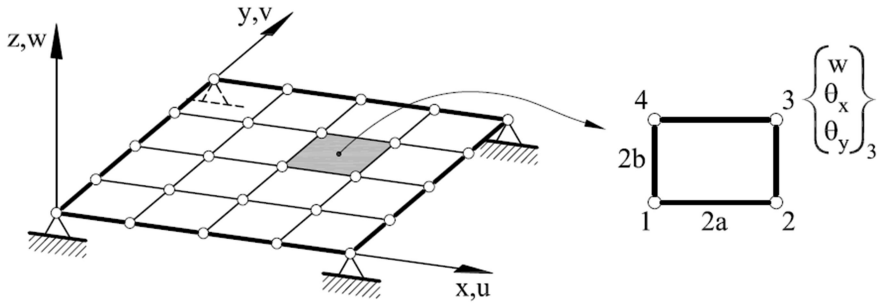
O trabalho virtual interno sobre o domínio da placa pode ser expresso em termos das variáveis do plano médio (isto é, tensões resultantes e deformações generalizadas) utilizando eqs. (3.7) e (3.20) como segue:

$$\iiint_V \delta\varepsilon^T \sigma \, dV = \iint_A \delta\hat{\varepsilon}^T \left[\int_{-\frac{t}{2}}^{\frac{t}{2}} S^T \sigma \, dz \right] dA = \iint_A \delta\hat{\varepsilon}^T \hat{\sigma} \, dA \tag{3.26}$$

O PTV é finalmente escrito em termos de integrais ao longo da superfície como:

$$\iint_A \delta\hat{\varepsilon}^T \hat{\sigma}^T \, dA = \iint_A \delta u^T t \, dA + \sum_i \delta u_i{}^T p_i \tag{3.27}$$

As integrais acima contêm derivadas de deslocamento apenas até a primeira ordem. Isso permite utilizar os elementos de continuidade C^0 os quais são mais simples do que os elementos de placa de Kirchhoff.



Às vezes é interessante dividir o trabalho interno virtual em termos de contribuições à flexão e ao cisalhamento transverso utilizando as eqs. (3.8) e (3.20). O PTV é então escrito como:

$$\iint_A \left[\delta\hat{\varepsilon}_b{}^T \hat{\sigma}_b + \delta\hat{\varepsilon}_s{}^T \hat{\sigma}_s \right] dA = \iint_A u^T t \, dA + \sum_i \delta u_i{}^T p_i \tag{3.28}$$

3.2 Formulação em elementos finitos

3.2.1 Discretização do campo de deslocamentos

O plano médio da placa é discretizado em uma malha de elementos de n nós. A deflexão e as duas rotações são *variáveis independentes* e o campo de deslocamento é interpolado na forma padrão C^0; isto é:

$$u = \begin{Bmatrix} w \\ \theta_x \\ \theta_y \end{Bmatrix} = \sum_{i=1}^{n} \begin{Bmatrix} N_i w_i \\ N_i \theta_{x_i} \\ N_i \theta_{y_i} \end{Bmatrix} = \begin{bmatrix} N_1 & 0 & 0 & \vdots & & \vdots & N_n & 0 & 0 \\ 0 & N_1 & 0 & \vdots & \cdots & \vdots & 0 & N_n & 0 \\ 0 & 0 & N_1 & \vdots & & \vdots & 0 & 0 & N_n \end{bmatrix} \begin{Bmatrix} w_1 \\ \theta_{x_1} \\ \theta_{y_1} \\ \cdots \\ \vdots \\ \cdots \\ w_n \\ \theta_{x_n} \\ \theta_{y_n} \end{Bmatrix} = \left[N_1, N_2, \cdots, N_n \right] = \begin{Bmatrix} a_1^{(e)} \\ \vdots \\ a_n^{(e)} \end{Bmatrix} = N a^{(e)} \tag{3.29}$$

onde

$$N=\left[N_1,N_2,\cdots,N_n\right],\quad a^{(e)}=\begin{Bmatrix}a_1^{(e)}\\a_2^{(e)}\\\vdots\\a_n^{(e)}\end{Bmatrix}$$
$$N_i=\begin{bmatrix}N_i & 0 & 0\\0 & N_i & 0\\0 & 0 & N_i\end{bmatrix},\ a_i^{(e)}=\begin{Bmatrix}w\\\theta_{x_i}\\\theta_{y_i}\end{Bmatrix} \tag{3.30}$$

são a matriz de função de forma e o vetor de deslocamento para o elemento e um no **i**, respectivamente.

3.2.2 Discretização das deformações generalizadas e campo de tensões resultante

As deformações generalizadas são expressas em termos de deslocamentos nodais (utilizando eqs. (3.8), (3.9) e (3.29)) como:

$$\hat{\varepsilon}=\begin{Bmatrix}\hat{\varepsilon}_b\\\cdots\\\hat{\varepsilon}_s\end{Bmatrix}=\begin{Bmatrix}\dfrac{\partial\theta_x}{\partial x}\\\dfrac{\partial\theta_y}{\partial y}\\\left(\dfrac{\partial\theta_x}{\partial y}+\dfrac{\partial\theta_y}{\partial x}\right)\\\cdots\\\dfrac{\partial w}{\partial x}-\theta_x\\\dfrac{\partial w}{\partial y}-\theta_y\end{Bmatrix}=\sum_{i=1}^{n}\begin{Bmatrix}\dfrac{\partial N_i}{\partial x}\theta_{x_i}\\\dfrac{\partial N_i}{\partial y}\theta_{y_i}\\\left(\dfrac{\partial N_i}{\partial y}\theta_{x_i}+\dfrac{\partial N_i}{\partial x}\theta_{y_i}\right)\\\cdots\\\dfrac{\partial N_i}{\partial x}w_i-N_i\theta_{x_i}\\\dfrac{\partial N_i}{\partial y}w_i-N_i\theta_{y_i}\end{Bmatrix}=\sum_{i=1}^{n}\begin{Bmatrix}B_{b_i}\\B_{s_i}\end{Bmatrix}a_i^{(e)}=\left[B_1,\cdots,B_n\right]\begin{Bmatrix}a_1^{(e)}\\\vdots\\a_n^{(e)}\end{Bmatrix}=Ba^{(e)} \tag{3.31}$$

onde B e B_i são as matrizes de deformação generalizada para o elemento e um nó i, respectivamente. Da eq. (3.31) deduzimos:

$$B_i = \begin{Bmatrix} B_{b_i} \\ \cdots \\ B_{s_i} \end{Bmatrix} \text{ com } B_{b_i} = \begin{bmatrix} 0 & \frac{\partial N_i}{\partial x} & 0 \\ 0 & 0 & \frac{\partial N_i}{\partial y} \\ 0 & \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} \end{bmatrix}, B_{s_i} = \begin{bmatrix} \frac{\partial N_i}{\partial x} & -N_i & 0 \\ \frac{\partial N_i}{\partial y} & 0 & -N_i \end{bmatrix} \quad (3.32)$$

B_{b_i} e B_{s_i} são as matrizes de deformação por flexão e cisalhamento transverso associados com o i-gésimo nó, respectivamente.

As tensões resultantes são expressas em termos de deslocamentos nodais usando eqs (3.22) e (3.32) como:

$$\hat{\sigma} = \hat{D}Ba^{(e)} \text{ e } \hat{\sigma}_b = \hat{D}_b B_b a^{(e)},\ \hat{\sigma}_s = \hat{D}_s B_s a^{(e)} \text{ com} \quad (3.33)$$

$$B_b = \left[B_{b_1}, B_{b_2}, \cdots, B_{b_n} \right],\ B_s = \left[B_{s_1}, B_{s_2}, \cdots, B_{s_n} \right] \quad (3.34)$$

3.2.3 Derivação das equações de equilíbrio para o elemento

O PTV para um único elemento sob cargas distribuídas é dado por:

$$\iint_{A^{(e)}} \delta\hat{\varepsilon}^T \hat{\sigma} dA = \iint_{A^{(e)}} \delta u^T t dA + \left[\delta a^{(e)} \right]^T q^{(e)} \quad (3.35)$$

onde, como de costume, $\delta a^{(e)}$ é o vetor de deformações nodais virtuais e o último termo do 2º membro é o trabalho virtual das forças nodais equilibrantes $q^{(e)}$, com:

$$\delta a^{(e)} = \begin{Bmatrix} \delta a_1^{(e)} \\ \vdots \\ \delta a_n^{(e)} \end{Bmatrix},\ \delta a_i^{(e)} = \begin{Bmatrix} \delta w_i \\ \delta\theta_{x_i} \\ \delta\theta_{y_i} \end{Bmatrix},\ q^{(e)} = \begin{Bmatrix} q_1^{(e)} \\ \vdots \\ q_n^{(e)} \end{Bmatrix},\ q_i^{(e)} = \begin{Bmatrix} F_{z_i} \\ M_{x_i} \\ M_{y_i} \end{Bmatrix} \quad (3.36)$$

Substituindo eq. (3.33) em (3.35) e usando as eqs. (3.29) e (3.31) fornece a expressão discreta de equilíbrio padrão para o elemento:

$$k^{(e)} a^{(e)} - f^{(e)} = q^{(e)}, \text{ onde} \quad (3.37)$$

$$k_{ij} = \iint_{A^{(e)}} B_i^T \hat{D} B_j dA \quad (3.38)$$

$$f_i^{(e)} = \iint_{A^{(e)}} N_i \left[f_z, m_x, m_y \right]^T dA \tag{3.39}$$

são a matriz de rigidez do elemento conectando nós i e j e o vetor de forças nodais equivalentes devido a um carregamento vertical distribuído f_z e momentos fletores distribuídos m_x e m_y.

O sistema global de equações $ka = f$ é obtido a partir da montagem das contribuições do elemento para a matriz de rigidez global **k** e o vetor de forças nodais equivalentes **f** de maneira usual.

A matriz de rigidez do elemento pode ser dividida nas contribuições de flexão e cisalhamento transverso usando as equações (3.21) e (3.32) como segue:

$$\begin{aligned} k_{ij} &= \iint_{A^{(e)}} \left[{B_{b_i}}^T, {B_{s_i}}^T \right]^T \hat{D} \begin{Bmatrix} B_{b_j} \\ B_{s_j} \end{Bmatrix} dA \\ &= \int_{A^{(e)}} \left({B_{b_i}}^T \hat{D}_b B_{b_j} + {B_{s_i}}^T \hat{D}_s B_{s_j} \right) dA = {k_{b_{ij}}}^{(e)} + {k_{s_{ij}}}^{(e)}, \text{ onde} \end{aligned} \tag{3.40}$$

$${k_{b_{ij}}}^{(e)} = \iint_{A^{(e)}} {B_{b_i}}^T \hat{D}_b B_{b_j} dA \text{ e } {k_{s_{ij}}}^{(e)} = \iint_{A^{(e)}} {B_{s_i}}^T \hat{D}_s B_{s_j} dA \tag{3.41}$$

A divisão acima fornece uma maneira mais econômica para a computação da matriz de rigidez do elemento. Também ajuda na explicação do comportamento do elemento para situações de placas finas.

Diferentemente dos elementos de placa de Kirchhoff, carregamentos verticais e momentos fletores contribuem aos termos de $f_i^{(e)}$ de maneira desacoplada, isto é, carregamentos verticais não introduzem componentes de momento fletor em $f_i^{(e)}$. Isso é devido à interpolação independente para a deflexão e as rotações.

O peso próprio é tratado como um carregamento distribuído vertical. Para a gravidade **g** atuando na direção oposta do eixo global **z**, $f_z = -\rho g t$ e:

$$f_i^{(e)} = -\iint_{A^{(e)}} N_i \rho g t \left[1,0,0 \right]^T dA \tag{3.42}$$

onde ρ e **t** são a densidade do material e a espessura da placa, respectivamente. Finalmente, o vetor de forças nodais equivalentes devido a uma carga externa vertical

pontual P_{z_j} e momentos fletores concentrados M_{x_j} e M_{y_j} agindo num nó com número global **j** é:

$$p_j = \left[P_{z_j}, M_{x_j}, M_{y_j} \right]^T \tag{3.43}$$

Como sempre forças concentradas agindo nos nós são montadas diretamente no vetor global **f**.

As reações nos nós prescritos são computados a posteriori uma vez que deslocamentos nodais já foram encontrados.

3.2.4 Integração numérica

As integrais que aparecem na matriz de rigidez do elemento e o vetor de forças nodais equivalentes são tipicamente computados por uma quadratura de Gauss.

Da eq (3.40) deduzimos:

$$\begin{aligned} k_{ij} = &\sum_{p=1}^{nbp}\sum_{q=1}^{nbq}\left[B_{b_i}{}^T \hat{D}_b B_{b_j} \left|J^e\right| \right]_{p,q} W_p W_q \\ &+\sum_{p=1}^{nsp}\sum_{q=1}^{nsq}\left[B_{s_i}{}^T \hat{D}_s B_{s_j} \left|J^e\right| \right]_{p,q} W_p W_q \end{aligned} \tag{3.44}$$

onde $|J^e|$ é o determinante jacobiano, (nbp, nbq) e (nsp, nsq) são os pontos de integração para as matrizes de rigidez da flexão e do cisalhamento transverso, respectivamente e W_p, W_q são os pesos correspondentes. Eq. (3.11) nos permite usar uma regra de integração seletiva para as matrizes de rigidez da flexão e do cisalhamento transverso.

A quadratura de Gauss para o vetor de forças nodais equivalentes da eq. (3.39) é:

$$\sum_{p=1}^{np}\sum_{q=1}^{nq}\left[N_i \left[f_z, m_x, m_y \right]^T \left|J^e\right| \right]_{p,q} W_p W_q \tag{3.45}$$

3.2.5 Condições de contorno

As condições de contorno padrão são:

- Ponto de apoio: $w_i = 0$

- Eixo de simetria (da geometria e do carregamento): $\theta_n = 0$, onde **n** é a direção ortogonal ao eixo de simetria.
- Lado engastado: $w = \theta_x = \theta_y = 0$.
- Lado simplesmente apoiado:
  - Apoio rígido: $w = \theta_s = 0$, onde **s** é a direção do lado.
  - Apoio elástico: $w = 0$.

Lembrando que na teoria de Kirchhoff o contorno de **w** ao longo de uma direção **s** automaticamente especifica $\theta_s = 0$. Isso não é o caso para a teoria de Reissner-Mindlin onde ambos **w** e θ_s devem ser prescritos independentemente.

Em placas com cantos, a condição de apoio deformável introduz uma camada de contorno de ordem **t** adjacente ao lado prescrito para as tensões resultantes Q_s, Q_n e M_{ns} onde **n** é a direção ortogonal ao lado. A condição de apoio rígido é recomendada nesses casos, já que a captura da camada de contorno requer uma discretização muito fina próxima ao lado (Oñate, 2013).

4. Elementos isoparamétricos

A formulação isoparamétrica permite aos elementos quadrilaterais e hexaédricos assumirem formas não retangulares. É uma formulação versátil que fornece elementos planos, sólidos, de placa e de casca (Cook, Malkus, Plesha, & Witt, 2002).

Elementos retangulares geralmente são fáceis de serem formulados, mas geralmente não são práticos porque são difíceis de representar uma malha utilizando apenas elementos retangulares, especialmente se a malha precisa ser fina o bastante para capturar detalhes em regiões críticas (Cook, Malkus, Plesha, & Witt, 2002). Elementos isoparamétricos podem ser não retangulares e até com lados curvos. Eles utilizam coordenadas auxiliares, as quais chamamos de $\xi\eta$ em duas dimensões e $\xi\eta\zeta$ em três dimensões. Coordenadas de referência mapeiam o elemento físico em um elemento de referência, que é um quadrado ou um cubo. Na formulação do elemento, o preço pago pela generalização da forma física do elemento é ter que lidar com transformação de coordenadas. E também, já que transformação produz formas algébricas que são incômodas para integrar exatamente, integrações numéricas mais dispendiosas são utilizadas.

Funções de forma são usadas para interpolar ambos os campos de deslocamentos (ou alguma outra variável de campo) e a geometria do elemento. Isto é, o

deslocamento de um ponto dentro de um elemento pode ser expresso em termos de graus de liberdade nodais e funções de forma $[N]$, as quais são funções de coordenadas de referência. De maneira similar, a posição global (coordenadas) de um ponto dentro do elemento pode ser expressa em termos de posições nodais globais e funções de forma $[\tilde{N}]$ as quais também são funções de coordenadas de referência.

Simbolicamente:

- Graus de liberdade nodais $\{d\}$ definem deslocamentos $\{u \quad v \quad w\}$ de um ponto dentro do elemento; isto é, $\{u \quad v \quad w\}^T = [N]\{d\}$.
- Coordenadas nodais $\{c\}$ definem coordenadas $\{x \quad y \quad z\}$ de um ponto dentro do elemento; isto é, $\{x \quad y \quad z\}^T = [\tilde{N}]\{c\}$.

Matrizes de funções de forma $[N]$ e $[\tilde{N}]$ são funções de ξ, η e ζ. Um elemento é chamado *isoparamétrico*, que significa "mesmo parâmetro", se $[N]$ e $[\tilde{N}]$ forem idênticos. Se $[\tilde{N}]$ for de grau menor que $[N]$, o elemento é chamado de *sub-paramétrico*. Se $[\tilde{N}]$ for de grau maior que $[N]$, o elemento é chamado de *superparamétrico*.

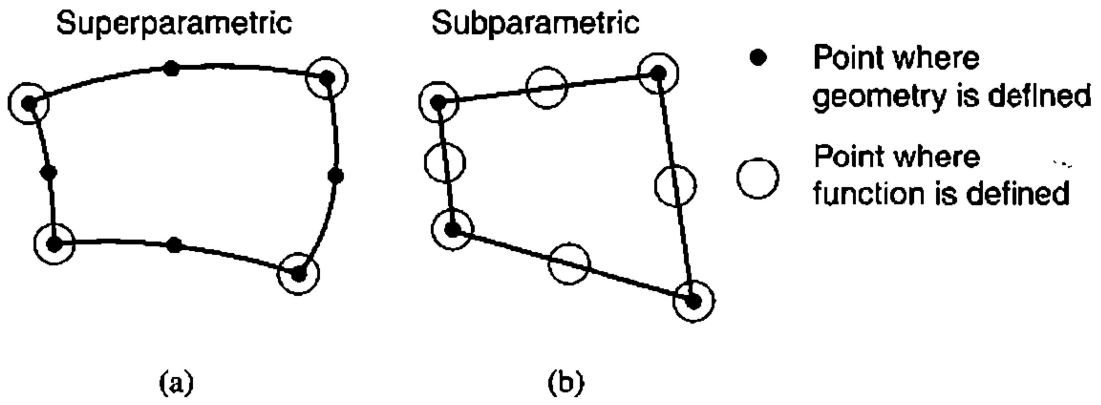


Figura 4-1. Exemplos de elementos planos subparamétricos e superparamétricos. Fonte: (Cook, Malkus, Plesha, & Witt, 2002)

4.1 Quadrilátero bilinear (Q4)

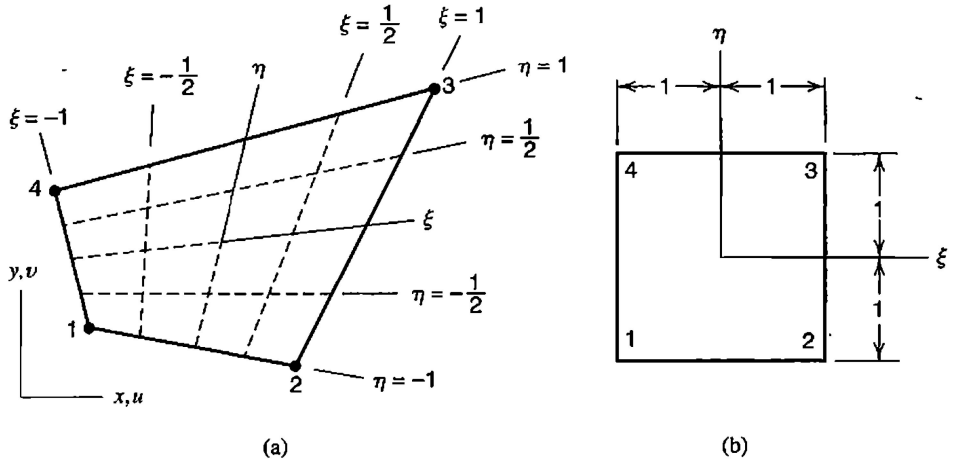


Figura 4.1-1. (a) Elemento plano de 4 nós no espaço físico. (b) O mesmo elemento, mapeado no espaço $\xi\eta$. Fonte: (Cook, Malkus, Plesha, & Witt, 2002)

Por causa de o elemento ser isoparamétrico, as mesmas funções de forma são utilizadas para interpolar ambas coordenadas e deslocamentos de um ponto dentro do elemento das coordenadas e deslocamentos dos nós. Portanto:

$$\begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{Bmatrix} \sum N_i x_i \\ \sum N_i y_i \end{Bmatrix} = \begin{bmatrix} N \end{bmatrix} \{ c \} \text{ e } \begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{Bmatrix} \sum N_i u_i \\ \sum N_i v_i \end{Bmatrix} = \begin{bmatrix} N \end{bmatrix} \{ d \} \tag{4.1}$$

onde o índice **i** nos somatórios vão de 1 a 4, e:

$$\begin{aligned} \{ c \} &= \begin{Bmatrix} x_1 & y_1 & x_2 & y_2 & x_3 & y_3 & x_4 & y_4 \end{Bmatrix}^T \\ \{ d \} &= \begin{Bmatrix} u_1 & v_1 & u_2 & v_2 & u_3 & v_3 & u_4 & v_4 \end{Bmatrix}^T \\ \begin{bmatrix} N \end{bmatrix} &= \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{bmatrix} \end{aligned} \tag{4.2}$$

A partir da figura abaixo, as funções de forma N_i de um elemento Q4 são:

$$\begin{aligned} N_1 &= \frac{(a-x)(b-y)}{4ab} \quad N_2 = \frac{(a+x)(b-y)}{4ab} \\ N_3 &= \frac{(a+x)(b+y)}{4ab} \quad N_4 = \frac{(a-x)(b+y)}{4ab} \end{aligned} \tag{4.3}$$

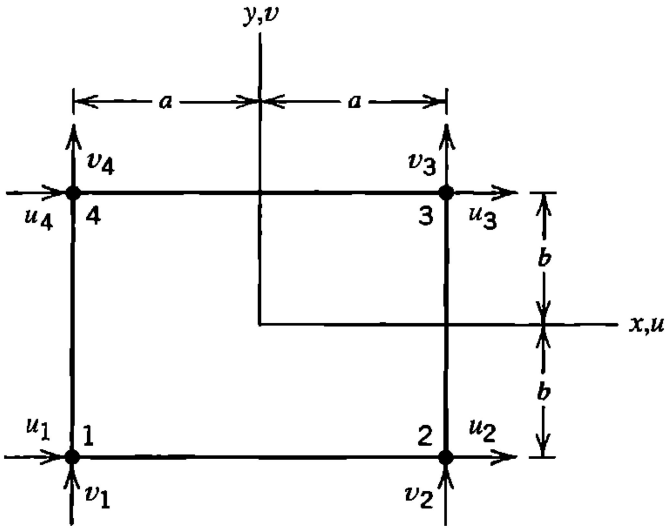


Figura 4.1-2. Quadrilátero bilinear e seus graus de liberdade. Fonte: (Cook, Malkus, Plesha, & Witt, 2002)

Na matriz $[N]$, as funções de forma individuais N_i são obtidas das equações (4.3) assinalando $a = 1, b = 1, x = \xi$ e $y = \eta$, resultando:

$$\begin{aligned} N_1 &= \frac{(1-\xi)(1-\eta)}{4} \quad N_2 = \frac{(1+\xi)(1-\eta)}{4} \\ N_3 &= \frac{(1+\xi)(1+\eta)}{4} \quad N_4 = \frac{(1-\xi)(1+\eta)}{4} \end{aligned} \tag{4.4}$$

Podemos verificar que cada N_i é unitário quando ξ e η assumem os valores das coordenadas dos nós **i**, mas zero quando ξ e η assumem os valores das coordenadas de qualquer outro nó.

Para um dado elemento físico, a orientação dos eixos $\xi\eta$ em relação aos eixos xy é ditada pelas funções de forma e numeração dos nós.

4.1.1 Transformação

A matriz deformação-deslocamento $[B]$ não pode ser escrita tão facilmente quanto as equações anteriores porque envolve gradientes, e um operador como $\partial/\partial x$ não é simplesmente uma constante vezes $\partial/\partial\xi$. No que se segue, consideramos uma função $\phi = \phi(\xi, \eta)$ e examinamos suas derivadas em relação a x e y. Aqui, ϕ pode representar tanto u como v. Exceto pela equação (4.6), as seguintes equações aplicam-se a um elemento plano que possui qualquer número de nós.

Derivadas em relação a x e y não estão disponíveis diretamente. Em vez disso, começamos com as derivadas em relação a ξ e η.

$$\begin{aligned}\frac{\partial \phi}{\partial \xi} &= \frac{\partial \phi}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial \phi}{\partial y}\frac{\partial y}{\partial \xi} \\ \frac{\partial \phi}{\partial \eta} &= \frac{\partial \phi}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial \phi}{\partial y}\frac{\partial y}{\partial \eta}\end{aligned} \quad \text{ou} \quad \begin{Bmatrix}\phi_{,\xi} \\ \phi_{,\eta}\end{Bmatrix} = \left[J\right]\begin{Bmatrix}\phi_{,x} \\ \phi_{,y}\end{Bmatrix} \tag{4.5}$$

onde [J] é chamado de matriz jacobiana.

$$\left[J\right] = \begin{bmatrix} x_{,\xi} & y_{,\xi} \\ x_{,\eta} & y_{,\eta}\end{bmatrix} = \begin{bmatrix} \sum N_{i,\xi} x_i & \sum N_{i,\xi} y_i \\ \sum N_{i,\eta} x_i & \sum N_{i,\eta} y_i\end{bmatrix} \tag{4.6}$$

Das equações (4.1) e (4.2), para o caso especial de elemento plano de 4 nós:

$$\left[J\right] = \frac{1}{4}\begin{bmatrix} -(1-\eta) & (1-\eta) & (1+\eta) & -(1+\eta) \\ -(1-\xi) & -(1+\xi) & (1+\xi) & (1-\xi)\end{bmatrix}\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4\end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22}\end{bmatrix} \tag{4.7}$$

As derivadas procuradas em relação a x e y são obtidas da equação (4.5):

$$\begin{Bmatrix}\phi_{,x} \\ \phi_{,y}\end{Bmatrix} = \underbrace{\begin{bmatrix}\Gamma_{11} & \Gamma_{12} \\ \Gamma_{21} & \Gamma_{22}\end{bmatrix}}_{\left[\Gamma\right]}\begin{Bmatrix}\phi_{,\xi} \\ \phi_{,\eta}\end{Bmatrix} \text{ onde } \left[\Gamma\right] = \left[J\right]^{-1} = \frac{1}{J} = \begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11}\end{bmatrix} \tag{4.8}$$

e J é o determinante da matriz jacobiana:

$$J = \det\left[J\right] = J_{11}J_{22} - J_{21}J_{12} \tag{4.9}$$

Muitas vezes, J é chamado apenas de jacobiano. Pode ser considerado como um fator escalar que multiplica $d\xi d\eta$ para produzir o incremento físico de área $dxdy$ (Cook, Malkus, Plesha, & Witt, 2002). Em geral, J é uma função de ξ e η, mas para retângulos e paralelogramos, ele assume o valor constante $J = A/4$, onde A é a área do elemento físico.

4.1.2 Matriz [B] e matriz de rigidez

No elemento, a relação deformação-deslocamento $\{\varepsilon\} = [B]\{d\}$, a matriz $[B]$ é igual ao produto das três matrizes retangulares das três equações seguintes. Equação (4.10) define as relações deformação-deslocamento . Equação é uma forma expandida da equação (4.8). Equação resulta da diferenciação da segunda das equações (4.1).

$$\{\varepsilon\} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \begin{Bmatrix} u_{,x} \\ u_{,y} \\ v_{,x} \\ v_{,y} \end{Bmatrix} \tag{4.10}$$

$$\begin{Bmatrix} u_{,x} \\ u_{,y} \\ v_{,x} \\ v_{,y} \end{Bmatrix} = \begin{bmatrix} \Gamma_{11} & \Gamma_{12} & 0 & 0 \\ \Gamma_{21} & \Gamma_{22} & 0 & 0 \\ 0 & 0 & \Gamma_{11} & \Gamma_{12} \\ 0 & 0 & \Gamma_{21} & \Gamma_{22} \end{bmatrix} \begin{Bmatrix} u_{,\xi} \\ u_{,\eta} \\ v_{,\xi} \\ v_{,\eta} \end{Bmatrix} \tag{4.11}$$

$$\begin{Bmatrix} u_{,\xi} \\ u_{,\eta} \\ v_{,\xi} \\ v_{,\eta} \end{Bmatrix} = \begin{bmatrix} N_{1,\xi} & 0 & N_{2,\xi} & 0 & N_{3,\xi} & 0 & N_{4,\xi} & 0 \\ N_{1,\eta} & 0 & N_{2,\eta} & 0 & N_{3,\eta} & 0 & N_{4,\eta} & 0 \\ 0 & N_{1,\xi} & 0 & N_{2,\xi} & 0 & N_{3,\xi} & 0 & N_{4,\xi} \\ 0 & N_{1,\eta} & 0 & N_{2,\eta} & 0 & N_{3,\eta} & 0 & N_{4,\eta} \end{bmatrix} \underset{8x1}{\{d\}} \tag{4.12}$$

onde $N_{1,\xi} = -(1-\eta)/4$, e assim sucessivamente. A matriz de rigidez de um elemento de espessura **t** é:

$$\underset{8\times8}{[k]} = \iint \underset{8\times3}{[B]}^T \underset{3\times3}{[D]} \underset{3\times8}{[B]} t\, dxdy = \int_{-1}^{+1}\int_{-1}^{+1} \underset{8\times3}{[B]}^T \underset{3\times3}{[D]} \underset{3\times8}{[B]} t\, J\, dxdy \tag{4.13}$$

Se a espessura do elemento t é variável e definida por espessuras t_i nos nós i, a espessura num local arbitrário pode ser obtida por interpolação de funções de forma, $t = \sum N_i\, t_i$ (Cook, Malkus, Plesha, & Witt, 2002).

5. Integração numérica

Para começar a explicação da quadratura de Gauss, consideramos um problema unidimensional sem referência ao método dos elementos finitos. Dado uma

função polinomial de grau $m \leq 2r-1$, assumimos que pode-se avaliar exatamente a seguinte integral com o método da quadratura de Gauss no intervalo $[-1,+1]$:

$$\int_{-1}^{+1} f(\xi)d\xi = \sum_{i=1}^{n} W_i f(\xi_i) \tag{5.1}$$

Baseado na nossa hipótese, segue-se que a eq. (5.1) é verificada para qualquer função polinomial da forma:

$$f(\xi) = \alpha_1 + \alpha_2\xi + \alpha_3\xi^2 + \cdots + \alpha_{2r}\xi^{2r-1} \tag{5.2}$$

Para obter os pesos W_i e a abscissa ξ_i, que são as incógnitas, substituímos (5.2) por $f(\xi)$ na eq. (5.1), no qual resulta:

$$\begin{aligned}&\alpha_1\int_{-1}^{+1} d\xi + \alpha_2\int_{-1}^{+1} \xi d\xi + \cdots + \alpha_{2r}\int_{-1}^{+1} \xi^{2r-1}d\xi = \alpha_1\left(W_1 + W_2 + \cdots + W_r\right) + \\ &\alpha_2\left(W_1\xi_1 + W_2\xi_2 + \cdots + W_r\xi_r\right) + \cdots + \alpha_{2r}\left(W_1{\xi_1}^{2r-1} + W_2{\xi_2}^{2r-1} + \cdots + W_r{\xi_r}^{2r-1}\right)\end{aligned} \tag{5.3}$$

Para a equação (5.3) ser satisfeita identicamente para todo α_i, devemos ter as seguintes igualdades:

$$\int_{-1}^{+1} \xi^{\alpha} d\xi = \frac{2}{\alpha+1} = \sum_{i=1}^{n} W_i f({\xi_i}^{\alpha}),\ \alpha = 0,2,4,\cdots,2r \tag{5.4}$$

$$\int_{-1}^{+1} \xi^{\alpha} d\xi = 0 = \sum_{i=1}^{n} W_i f({\xi_i}^{\alpha}),\ \alpha = 1,3,5,\cdots,2r-1 \tag{5.5}$$

que resulta:

$$\begin{aligned}2 &= W_1 + W_2 + \cdots + W_r \\ 0 &= W_1\xi_1 + W_2\xi_2 + \cdots + W_r\xi_r \\ \frac{2}{3} &= W_1{\xi_1}^2 + W_2{\xi_2}^2 + \cdots + W_r{\xi_r}^2 \\ 0 &= W_1{\xi_1}^{2r-1} + W_2{\xi_2}^{2r-1} + \cdots + W_r{\xi_r}^{2r-1}\end{aligned} \tag{5.6}$$

O sistema (5.6) é linear em W_i mas não-linear em ξ_i, e determina os parâmetros de (5.1) sob as condições:

$$\left.\begin{array}{c} W_i = 0 \\ -1 \le \xi \le +1 \end{array}\right\} i = 1,2,\cdots,r \tag{5.7}$$

Entretanto, não há necessidade de resolver o sistema (5.6) para obter a abscissa ξ_i e os pesos W_i. A abscissa ξ_i são as raízes dos polinômios de Legendre de ordem **r**, que são definidos, para $k = 1,2,\cdots,r$, como:

$$\begin{aligned} P_0(\xi) &= 1 \\ P_1(\xi) &= \xi \\ \cdots &= \cdots \\ P_k(\xi) &= \frac{2k-1}{k}\xi P_{k-1}(\xi) - \frac{k-1}{k}\xi P_{k-2}(\xi) \end{aligned} \tag{5.8}$$

e os pesos W_i são obtidos por (Khennanne, 2013):

$$W_i = \frac{2(1-\xi_i^2)}{(r(P_{r-1}(\xi))^2} \tag{5.9}$$

Exemplo 1 – pesos e abscissas para $r = 2$:

$$\begin{aligned} P_0(\xi) &= 1 \\ P_1(\xi) &= \xi \\ P_2(\xi) &= \frac{3}{2}\xi^2 - \frac{1}{2} \end{aligned}$$

As raízes de $P_2(\xi) = 0$ são:

$$\xi_i = \pm\frac{1}{\sqrt{3}}$$

Os pesos W_1 e W_2 podem ser obtidos do sistema (5.6) como:

$$\begin{aligned} 2 &= W_1 + W_2 \\ 0 &= -\frac{1}{\sqrt{3}}W_1 + \frac{1}{\sqrt{3}}W_2 \end{aligned}$$

ou diretamente da equação (5.9). Em ambos os casos, obtemos:

$$W_1 = W_2 = 1$$

5.1 Integração sobre um intervalo arbitrário [a,b]

Até agora o método da quadratura de Gauss foi apresentado apenas para o domínio $[-1,+1]$. E se o intervalo de integração for de forma generalizada como $[a,b]$? Isto é, avaliar uma integral da forma:

$$\int_a^b f(x)dx \tag{5.10}$$

Nesse caso, transformamos o intervalo $[-1,+1]$ no intervalo $[a,b]$ através de uma mudança de variável. Em outras palavras, definimos uma transformação linear entre $[-1,+1]$ e $[a,b]$. A expressão analítica da transformação linear entre dois intervalos é dada por:

$$x = \frac{b-a}{2}\xi + \frac{b+a}{2} \tag{5.11}$$

Derivando nos dá:

$$dx = \frac{b-a}{2}d\xi \tag{5.12}$$

Substituindo as equações (5.12) e (5.11) na equação (5.10) resulta:

$$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{+1} f\left(x(\xi)\right)d\xi = \frac{b-a}{2}\sum_{i=1}^{r} W_i f\left(x(\xi_i)\right) \tag{5.13}$$

5.2 Integração em duas e três dimensões

Integrando em duas e três dimensões consiste em usar uma única integral em cada dimensão. Por exemplo, o valor de $\int_{-1}^{+1}\int_{-1}^{+1} f(\xi,\eta)d\xi d\eta$ é executado da seguinte forma:

$$\int_{-1}^{+1}\int_{-1}^{+1} f\left(\xi,\eta\right)d\xi d\eta = \sum_{i=1}^{r_1}\sum_{j=1}^{r_2} W_i W_j f\left(\xi_i,\eta_j\right) \tag{5.14}$$

Note que um número diferente de pontos de Gauss pode ser usado em cada direção. O método integra exatamente o produto de um polinômio de grau $2r_1 - 1$ em ξ e um polinômio de grau $2r_2 - 1$ em η.

Em três dimensões, a equação (5.14) fica:

$$\int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} f\left(\xi,\eta,\zeta\right)d\xi d\eta d\zeta = \sum_{i=1}^{r_1}\sum_{j=1}^{r_2}\sum_{k=1}^{r_3} W_i W_j W_k f\left(\xi_i,\eta_j,\zeta_k\right) \tag{5.15}$$

5.3 Integração sobre um elemento de referência

A quadratura de Gauss avalia integrais entre $[-1,+1]$, integrais duplas sobre um quadrado de lado 2 e integrais triplas sobre um cubo de lado 2. Por exemplo, para avaliar uma integral sobre um quadrilátero, é necessário transformar o quadrilátero em um elemento de referência sobre o qual a integração possa ser executada. Por exemplo, o valor da integral $\int_A f(x,y)dA$ sobre uma área de quadrilátero é executado como segue:

1- Já que o quadrilátero bilinear é isoparamétrico, escrevemos as coordenadas **x** e **y** em termos de coordenadas de referência ξ e η:

$$\begin{aligned} x(\xi,\eta) &= N_1(\xi,\eta)x_1 + N_2(\xi,\eta)x_2 + N_3(\xi,\eta)x_3 + N_4(\xi,\eta)x_4 \\ y(\xi,\eta) &= N_1(\xi,\eta)y_1 + N_2(\xi,\eta)y_2 + N_3(\xi,\eta)y_3 + N_4(\xi,\eta)y_4 \end{aligned} \tag{5.16}$$

2- As funções de forma são dadas pelas equações a seguir:

$$\begin{aligned} N_1(\xi,\eta) &= 0,25(1-\xi-\eta+\xi\eta) \\ N_2(\xi,\eta) &= 0,25(1+\xi-\eta-\xi\eta) \\ N_3(\xi,\eta) &= 0,25(1+\xi+\eta+\xi\eta) \\ N_4(\xi,\eta) &= 0,25(1-\xi+\eta+\xi\eta) \end{aligned} \tag{5.17}$$

3- Utilizamos a equação da transformação do jacobiano para expressar a área $dA = dxdy$ em termos de elementos de área correspondentes $d\xi d\eta$ do elemento de referência:

$$\left[J\right] = \frac{1}{4}\begin{bmatrix} -(1-\eta) & (1-\eta) & (1+\eta) & -(1+\eta) \\ -(1-\xi) & -(1+\xi) & (1+\xi) & (1-\xi) \end{bmatrix}\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix} \tag{5.18}$$

$$dxdy = \det\left[J\right]d\xi d\eta$$

4- Construímos uma aproximação nodal para a função usando seus valores nodais:

$$\bar{f}(\xi,\eta)=\sum_{i=1}^{n}N_i(\xi,\eta)f_i \tag{5.19}$$

5- Finalmente a integral fica:

$$I=\int_{-1}^{+1}\int_{-1}^{+1}\left(\sum_{i=1}^{n}N_i(\xi,\eta)f_i\right)\det\left[J\right]d\xi d\eta \tag{5.20}$$

6. Resultados da implementação

Este capítulo ilustra os resultados obtidos com a implementação do programa.

6.1 Placa de Kirchhoff

Simulou-se várias situações de apoio de uma placa quadrada de lado 1m com carregamento distribuído unitário ($q=-1{,}0\ kN/m$) e espessura 0,01m. As propriedades mecânicas do material são: módulo de elasticidade ($E=10{,}92\ kN/m^2$) e coeficiente de Poisson ($\nu=0{,}3$). Em cada situação, a placa foi discretizada utilizando-se uma malha estruturada de 400 elementos (vinte ao longo da largura e vinte ao longo do comprimento), formando-se 1323 nós no total. OBS: a implementação considera o momento fletor que traciona as fibras da face de baixo da placa com sinal negativo.

6.1.1 Os quatro cantos engastados

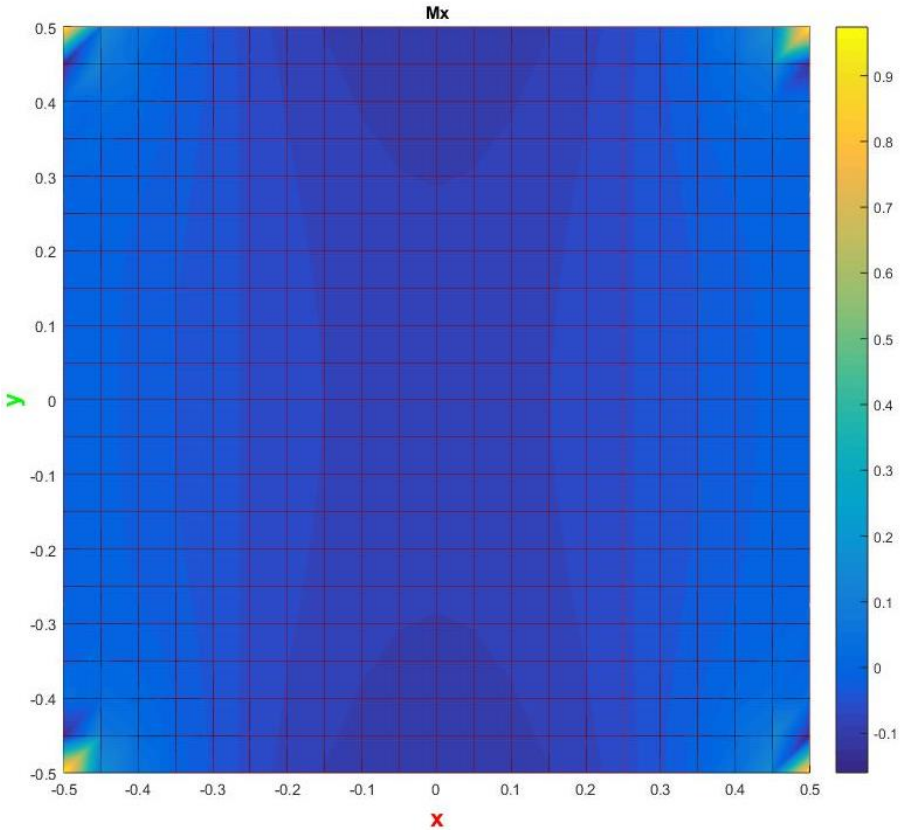


Figura 6.1-1. Momento fletor Mx. Fonte: Elaborado pelo autor.

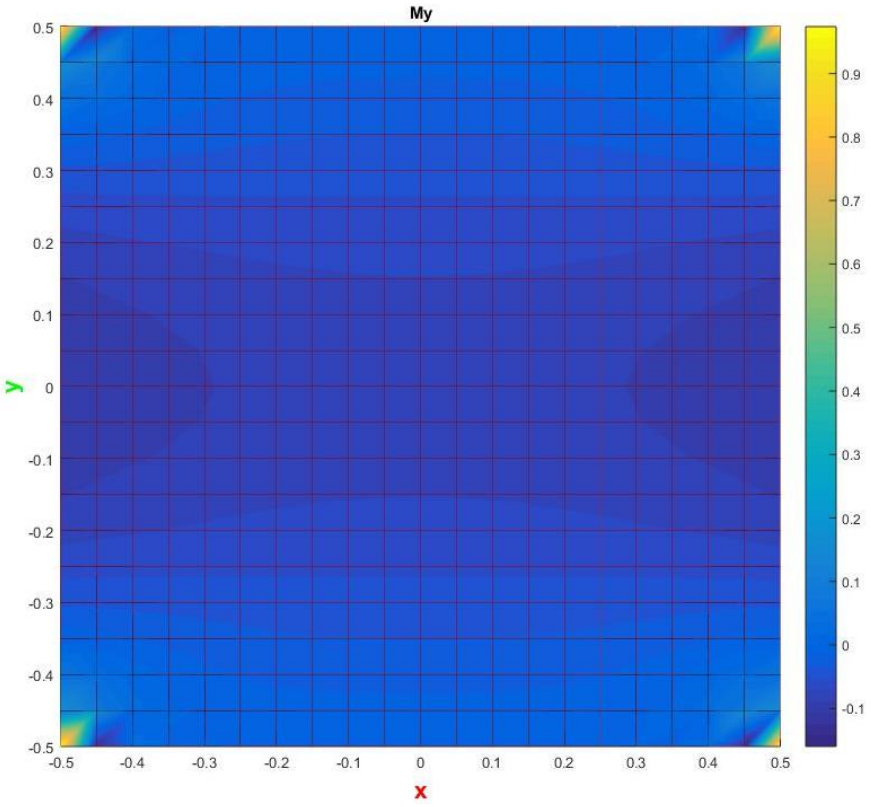


Figura 6.1-2. Momento fletor My. Fonte: Elaborado pelo autor.

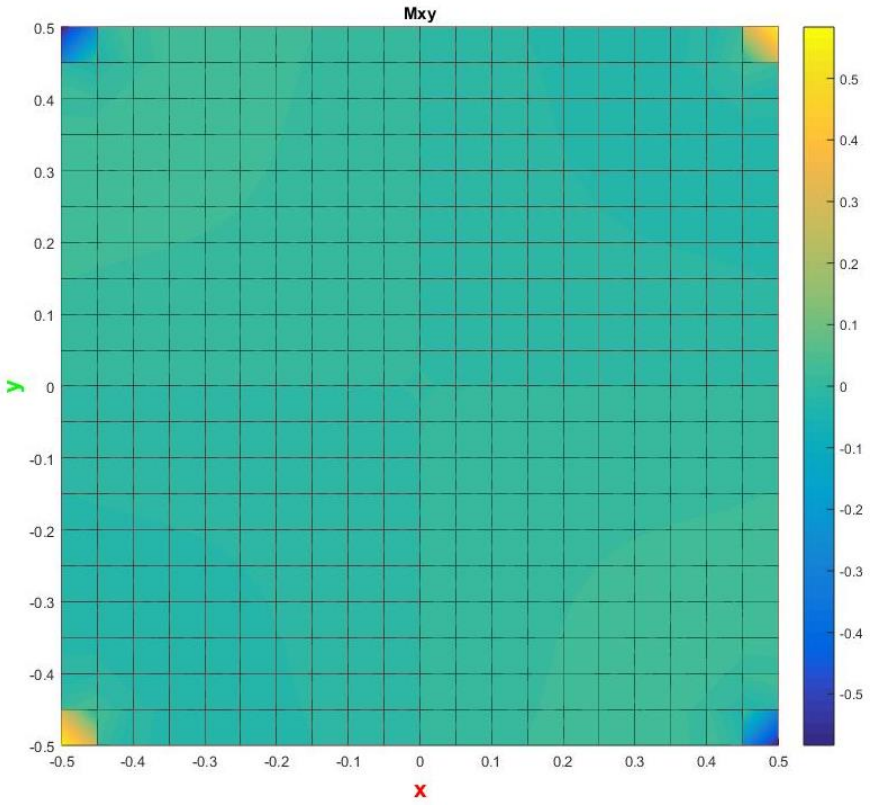


Figura 6.1-3. Momento torçor Mxy. Fonte: Elaborado pelo autor.

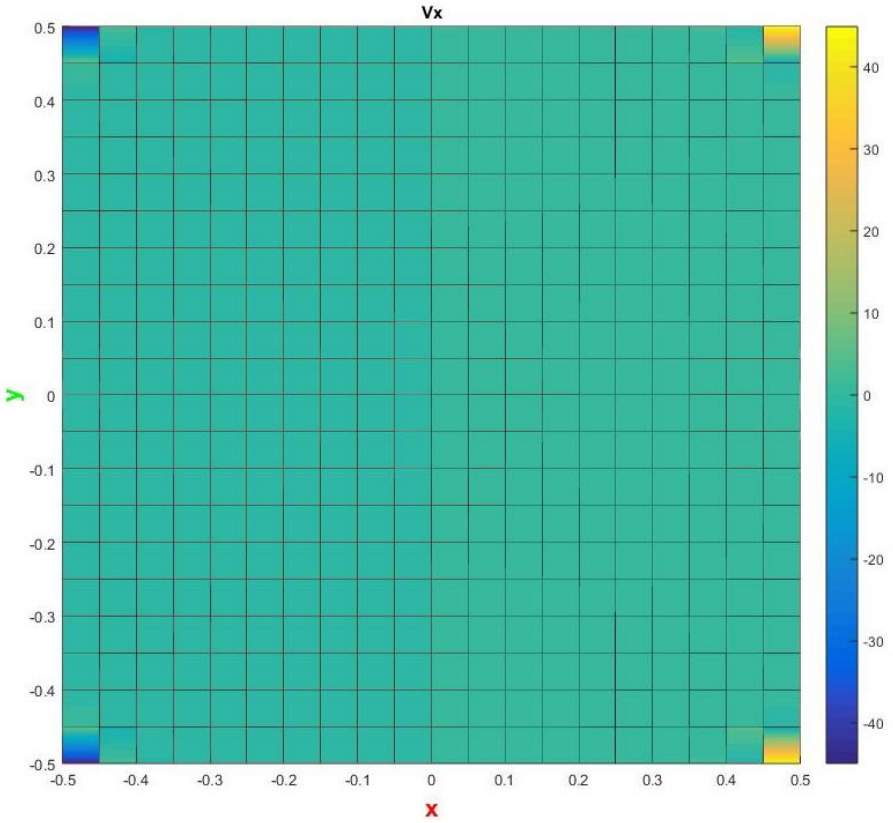


Figura 6.1-4. Esforço cortante Vx. Fonte: Elaborado pelo autor.

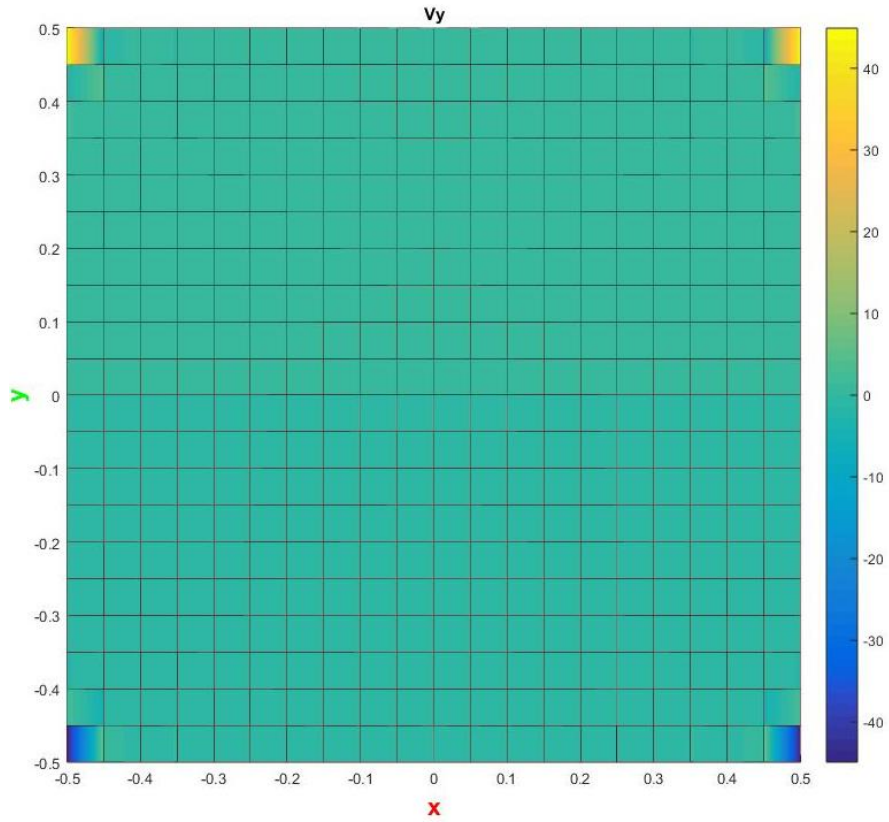


Figura 6.1-5. Esforço cortante Vy. Fonte: Elaborado pelo autor.

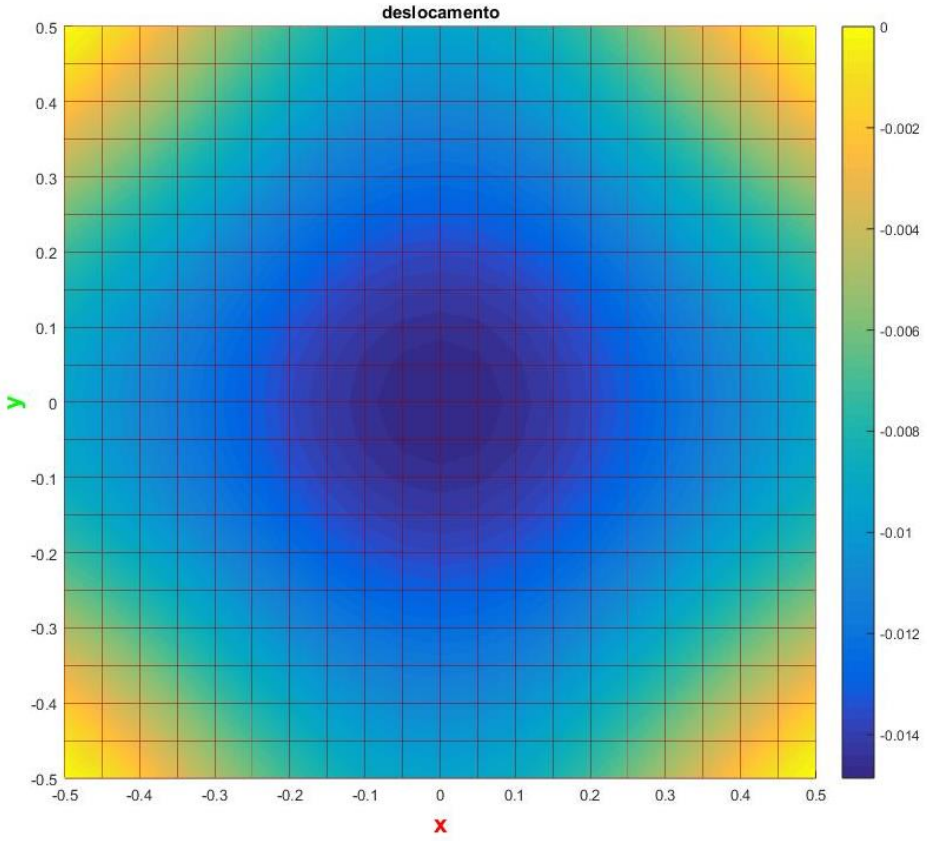


Figura 6.1-6. Deslocamento vertical. Fonte: Elaborado pelo autor.

Tabela 1 – esforços em uma placa com os quatro cantos engastados

Tipo de esforço	Valor
Deslocamento máximo	-1.4859e-02 m
Momento fletor positivo máximo (Mx)	-1.5984e-01 kN.m
Momento fletor negativo máximo (Mx)	9.7392e-01 kN.m
Momento fletor positivo máximo (My)	-1.5984e-01 kN.m
Momento fletor negativo máximo (My)	9.7392e-01 kN.m
Momento torçor positivo máximo (Mxy)	5.8333e-01 kN.m
Momento torçor negativo máximo (Mxy)	-5.8333e-01 kN.m
Esforço cortante positivo máximo (Vx)	3.1756e+01 kN/m
Esforço cortante negativo máximo (Vx)	-3.1756e+01 kN/m
Esforço cortante positivo máximo (Vy)	3.1756e+01 kN/m
Esforço cortante negativo máximo (Vy)	-3.1756e+01 kN/m

Fonte: Elaborado pelo autor

6.1.2 Os quatro cantos apoiados

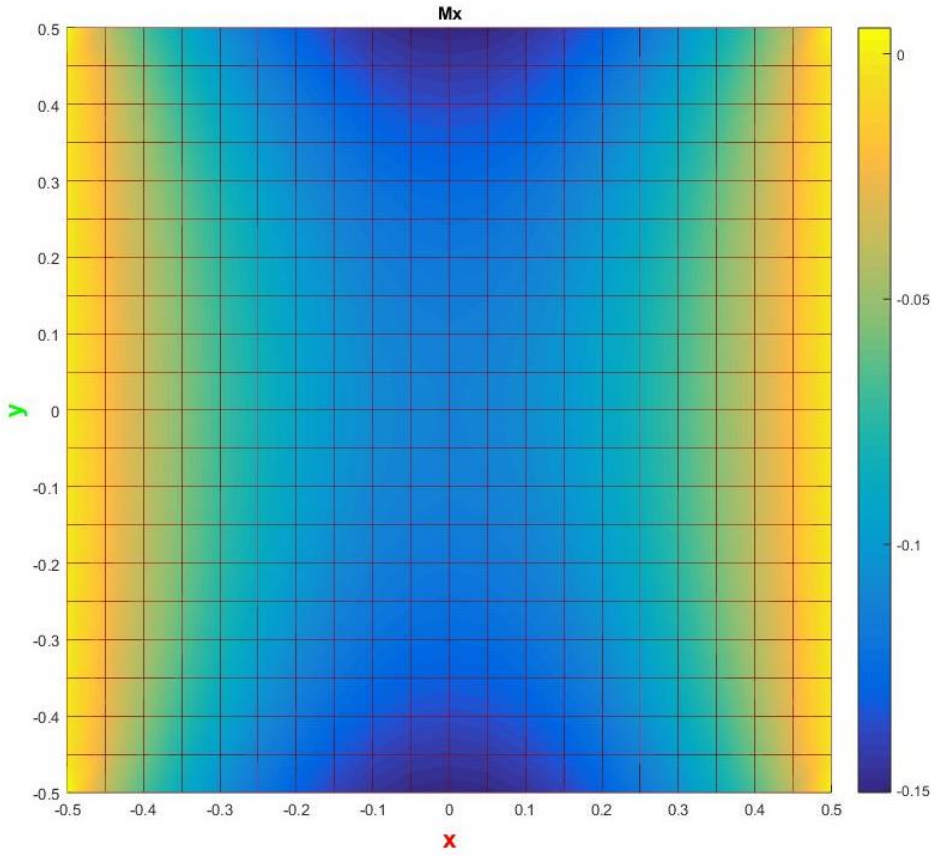


Figura 6.1-7. Momento fletor Mx. Fonte: Elaborado pelo autor.

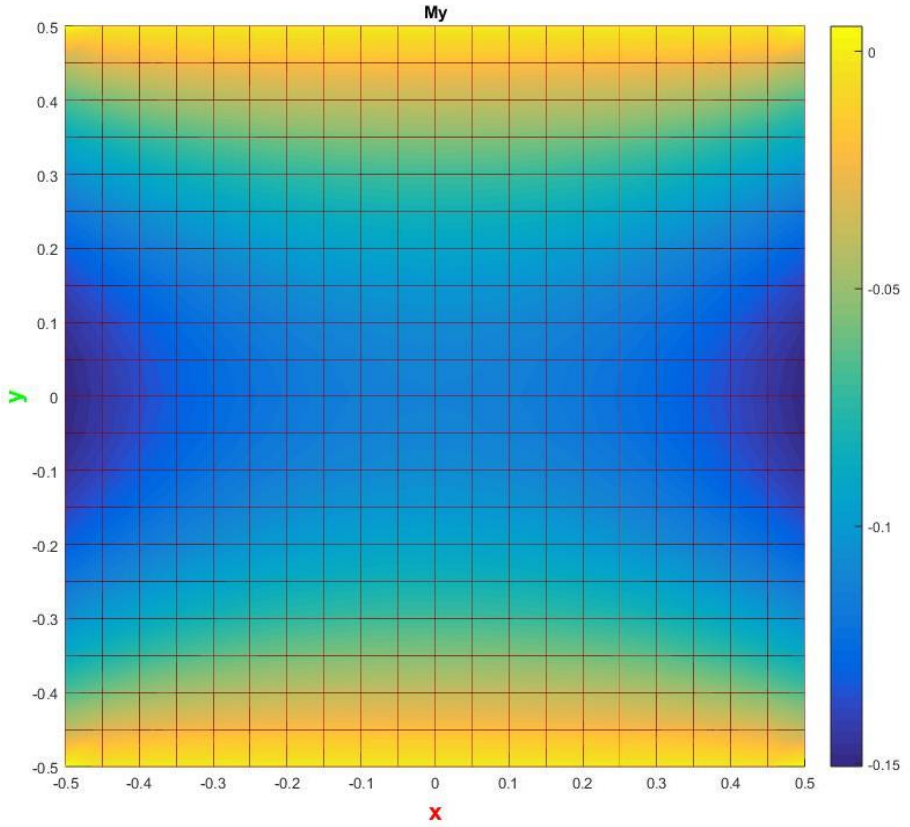


Figura 6.1-8. Momento fletor My. Fonte: Elaborado pelo autor.

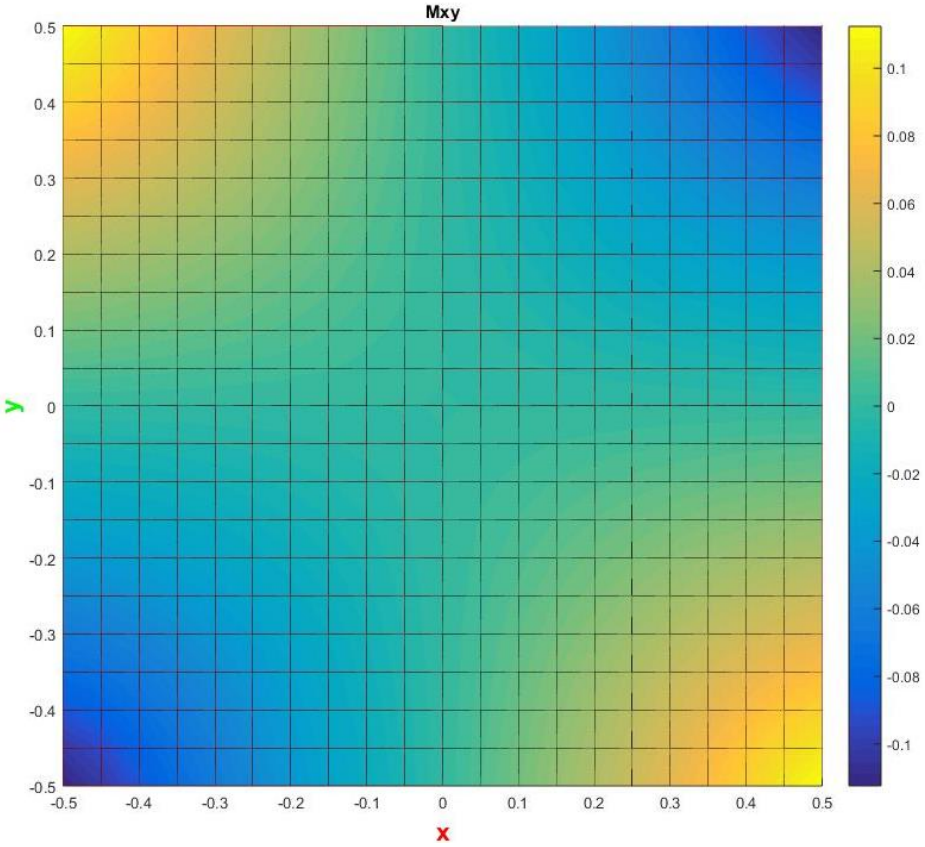


Figura 6.1-9. Momento torçor Mxy. Fonte: Elaborado pelo autor.

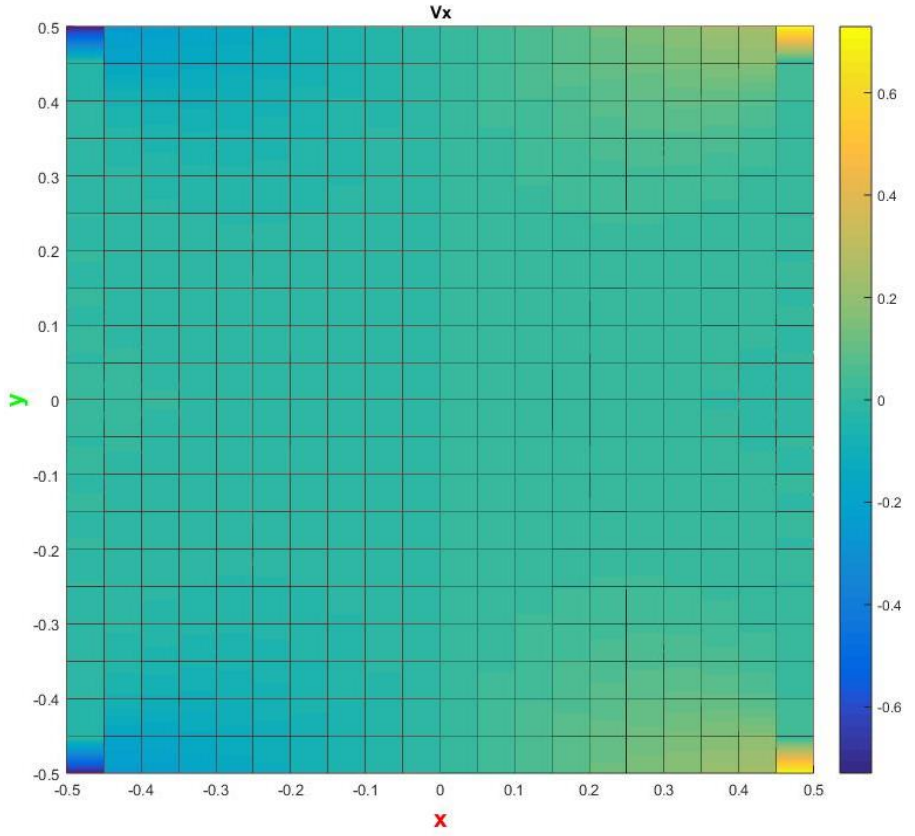


Figura 6.1-10. Esforço cortante Vx. Fonte: Elaborado pelo autor.

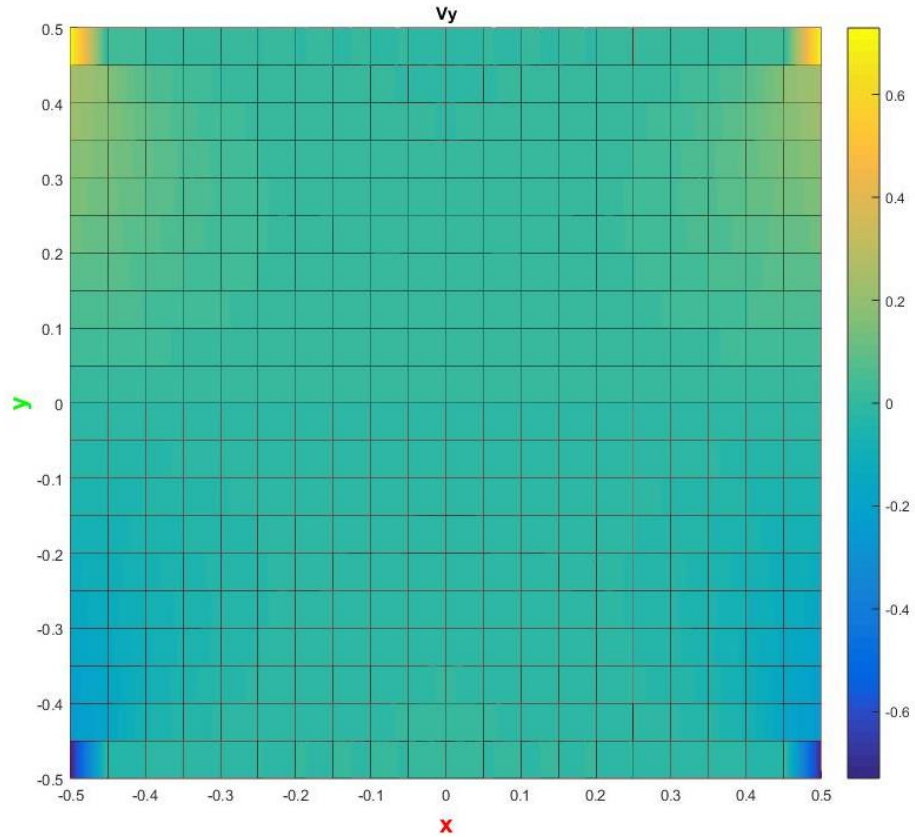


Figura 6.1-11. Esforço cortante Vy. Fonte: Elaborado pelo autor.

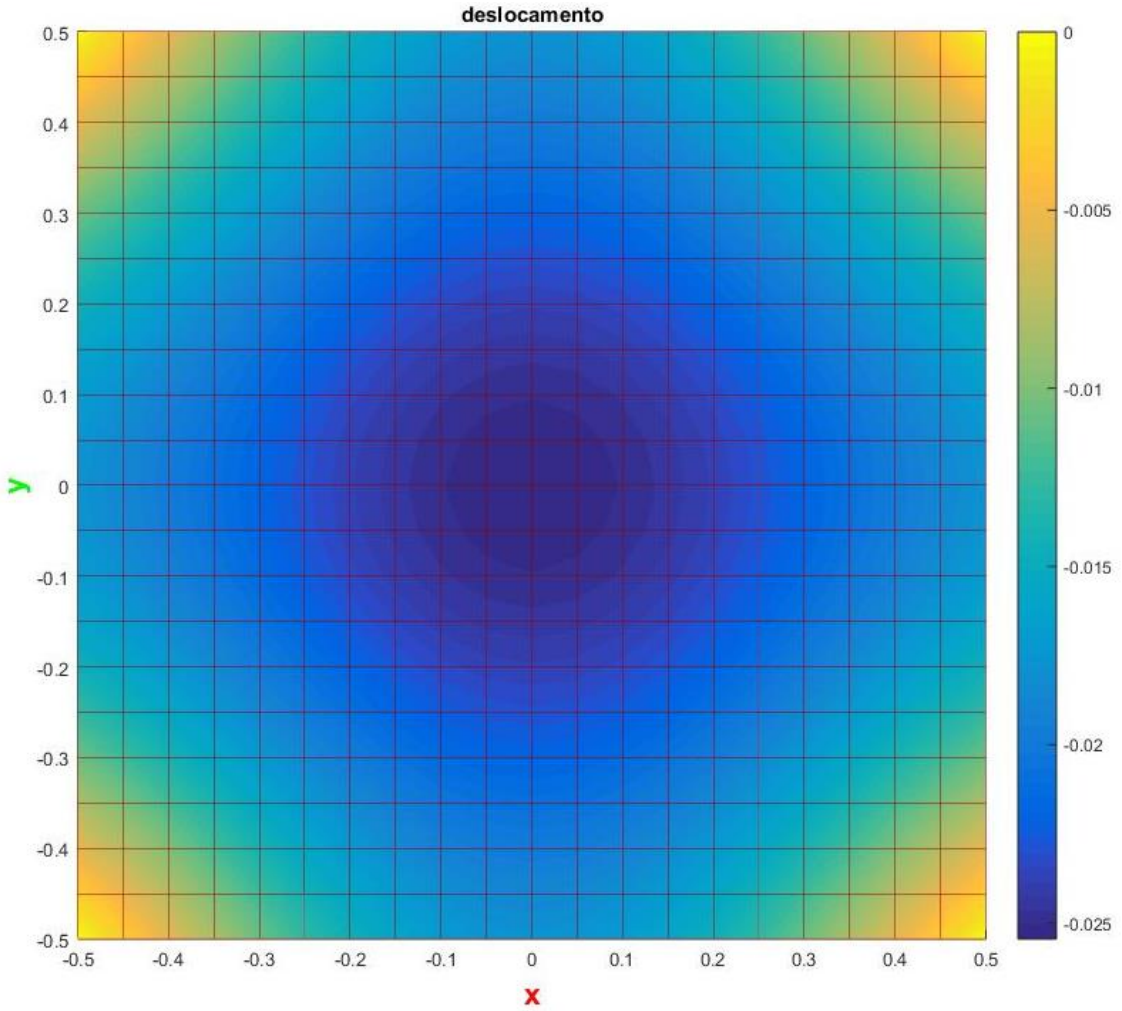


Figura 6.1-12. Deslocamento vertical. Fonte: Elaborado pelo autor.

Tabela 2 – esforços em uma placa com os quatro cantos apoiados

Tipo de esforço	Valor
Deslocamento máximo	-2.5452e-02 m
Momento fletor positivo máximo (Mx)	-1.5054e-01 kN.m
Momento fletor negativo máximo (Mx)	5.2616e-03 kN.m
Momento fletor positivo máximo (My)	-1.5054e-01 kN.m
Momento fletor negativo máximo (My)	5.2616e-03 kN.m
Momento torçor positivo máximo (Mxy)	1.1244e-01 kN.m
Momento torçor negativo máximo (Mxy)	-1.1244e-01 kN.m
Esforço cortante positivo máximo (Vx)	7.2915e-01 kN/m
Esforço cortante negativo máximo (Vx)	-7.2915e-01 kN/m
Esforço cortante positivo máximo (Vy)	7.2915e-01 kN/m
Esforço cortante negativo máximo (Vy)	-7.2915e-01 kN/m

Fonte: Elaborado pelo autor

6.1.3 Os quatro lados engastados

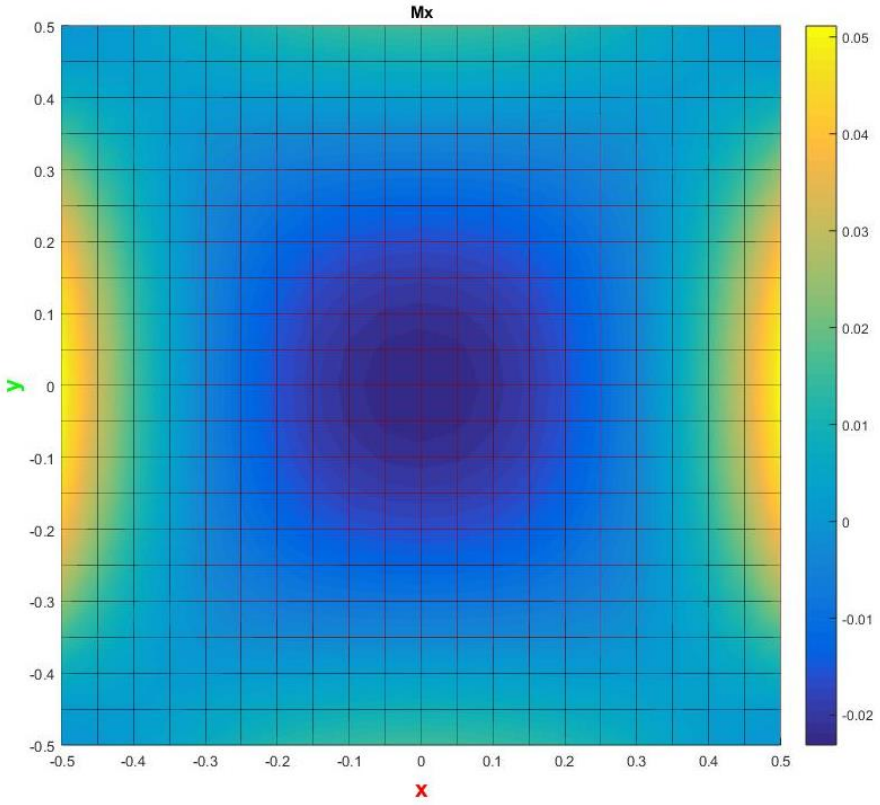


Figura 6.1-13. Momento fletor Mx. Fonte: Elaborado pelo autor.

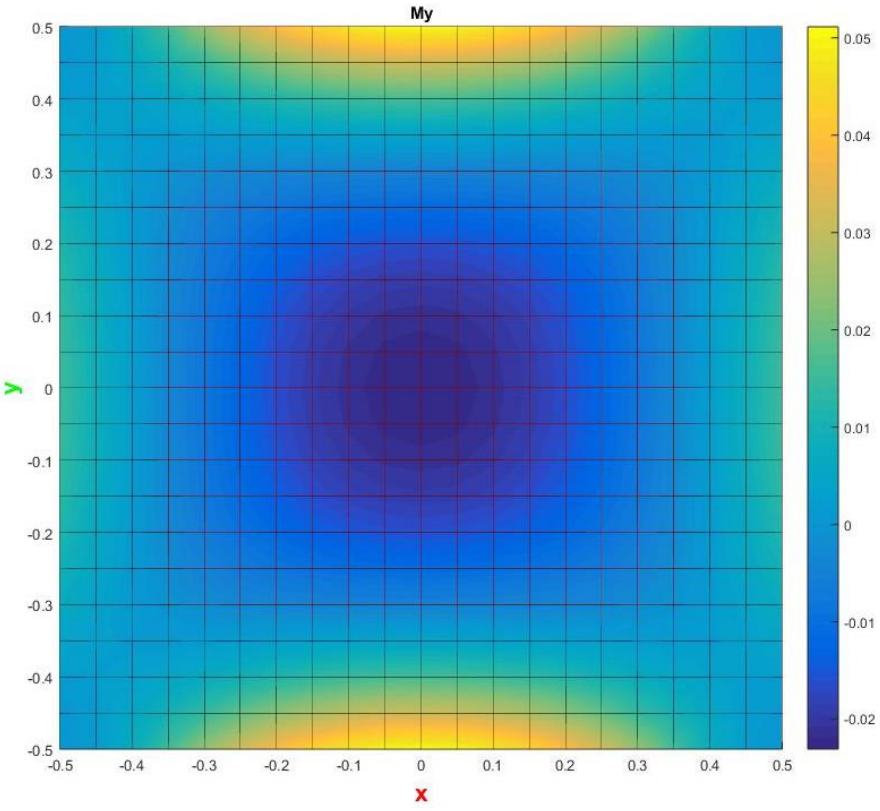


Figura 6.1-14. Momento fletor My. Fonte: Elaborado pelo autor.

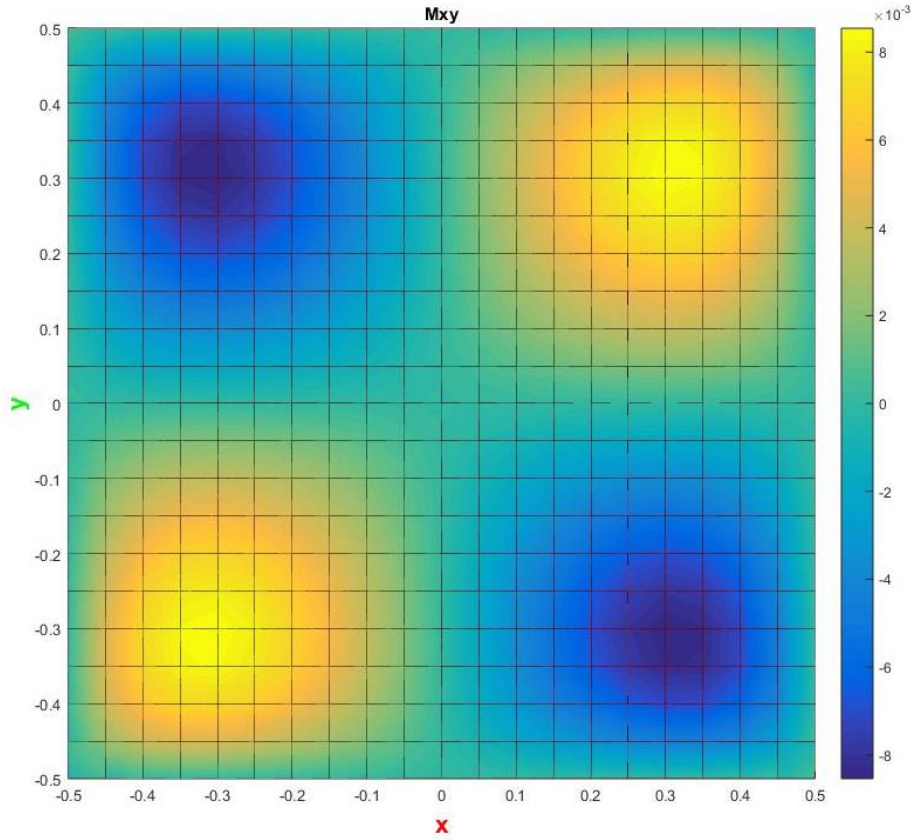


Figura 6.1-15. Momento torçor Mxy. Fonte: Elaborado pelo autor.

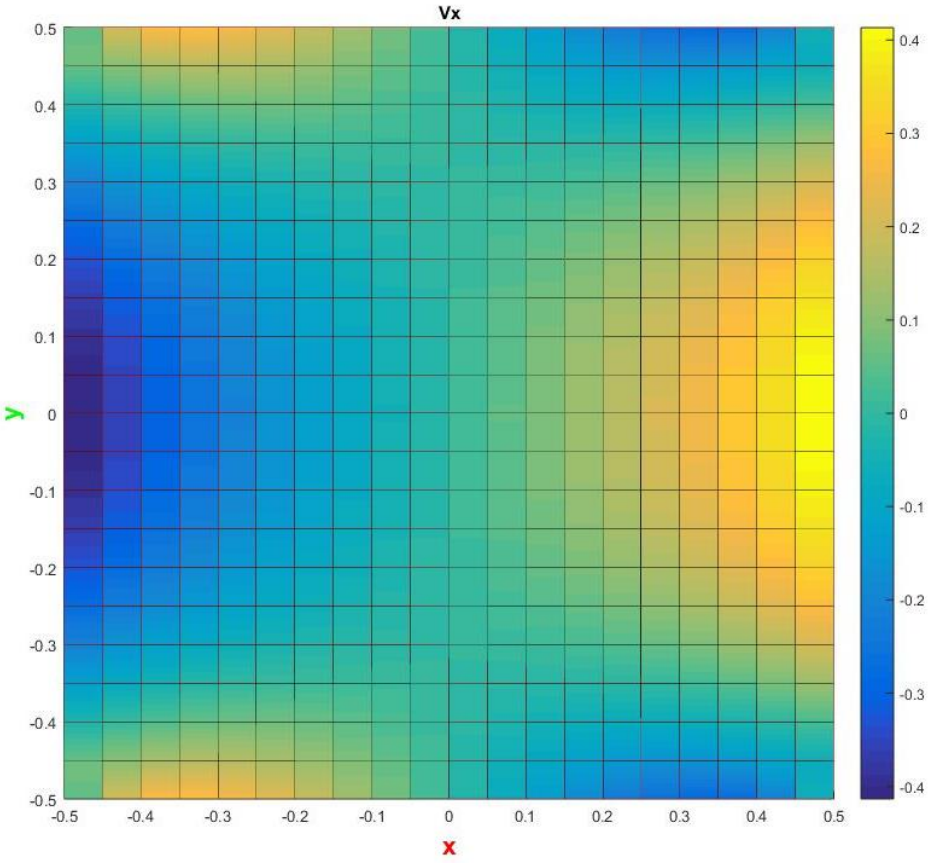


Figura 6.1-16. Esforço cortante Vx. Fonte: Elaborado pelo autor.

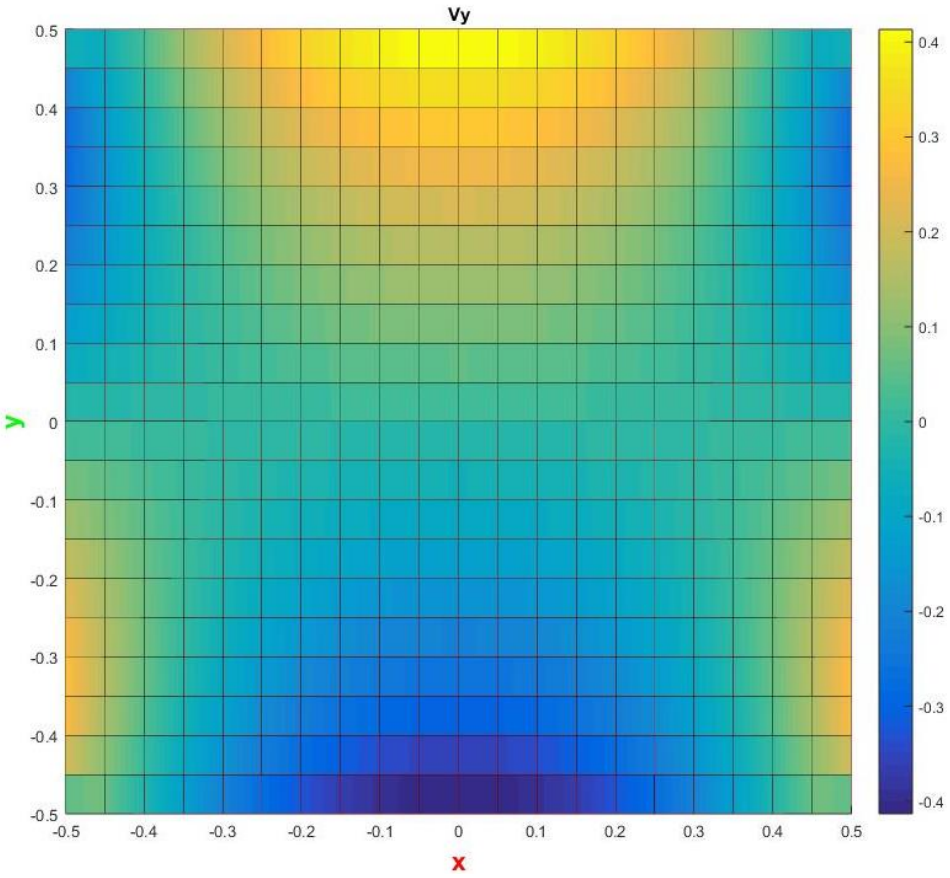


Figura 6.1-17. Esforço cortante Vy. Fonte: Elaborado pelo autor.

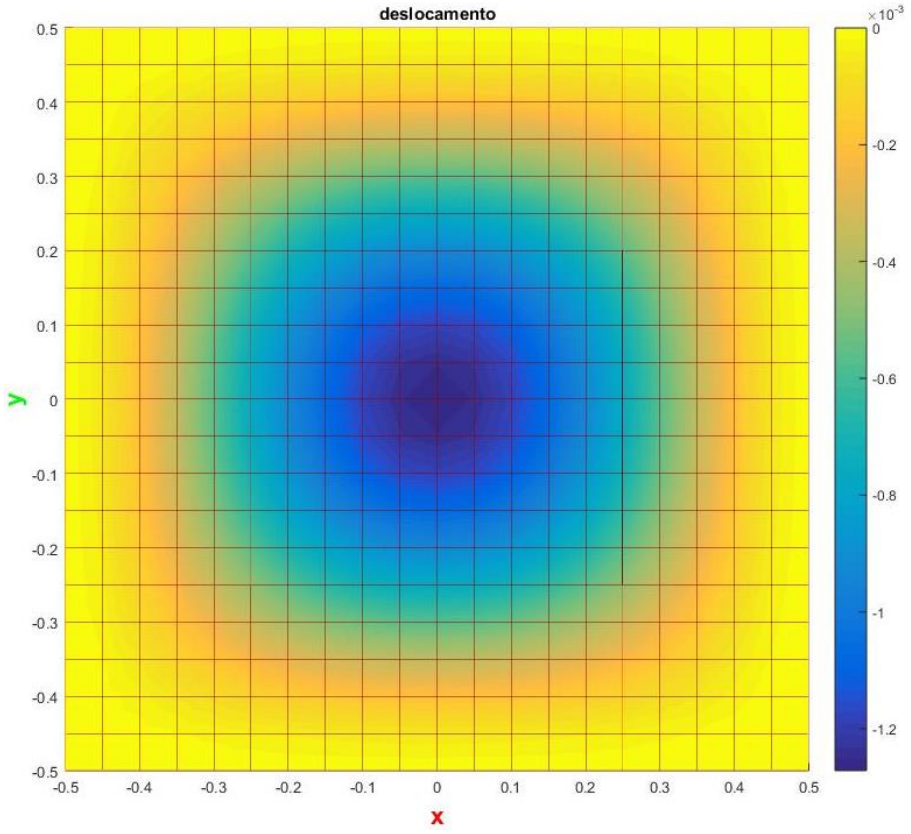


Figura 6.1-18. Deslocamento vertical. Fonte: Elaborado pelo autor.

Tabela 3 – esforços em uma placa com os quatro lados engastados

Tipo de esforço	Valor
Deslocamento máximo	-1.2716e-03 m
Momento fletor positivo máximo (Mx)	-2.3085e-02 kN.m
Momento fletor negativo máximo (Mx)	5.1140e-02 kN.m
Momento fletor positivo máximo (My)	-2.3085e-02 kN.m
Momento fletor negativo máximo (My)	5.1140e-02 kN.m
Momento torçor positivo máximo (Mxy)	8.5432e-03 kN.m
Momento torçor negativo máximo (Mxy)	-8.5432e-03 kN.m
Esforço cortante positivo máximo (Vx)	4.1335e-01 kN/m
Esforço cortante negativo máximo (Vx)	- 4.1335e-01 kN/m
Esforço cortante positivo máximo (Vy)	4.1335e-01 kN/m
Esforço cortante negativo máximo (Vy)	- 4.1335e-01 kN/m

Fonte: Elaborado pelo autor

Tabela 4 – resultados do deslocamento no centro da placa, momentos para as discretizações distintas e os elementos empregados.

	Número de grados de libertad		Desplazamiento vertical en el centro de la placa $(\times 10^{-3})$		Momento M_x en el centro de la placa $(\times 10^{-2})$	
Número de Elementos	ACM	DKT4	ACM	DKT4	ACM	DKT4
4	27	39	−1,4796	−1,4196	−4,6165	−3,6208
16	75	123	−1,4033	−1,3246	−2,7783	−2,5595
440	1323	2523	−1,2716	−1,2682	−2,3085	−2,301
1600	4800	6400	−1,2669	−1,2661	−2,295	−2,2931

Fonte: (Chaves & Minguez, 2010)

6.1.4 Os quatro lados apoiados

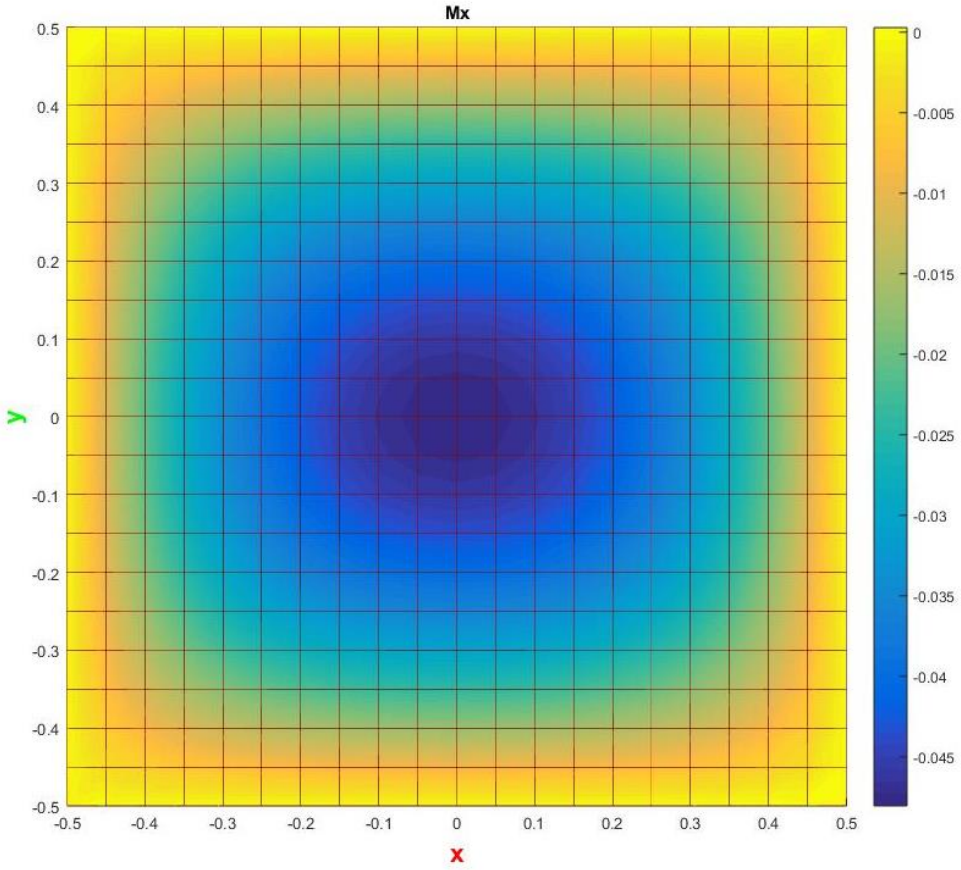


Figura 6.1-19. Momento fletor Mx. Fonte: Elaborado pelo autor.

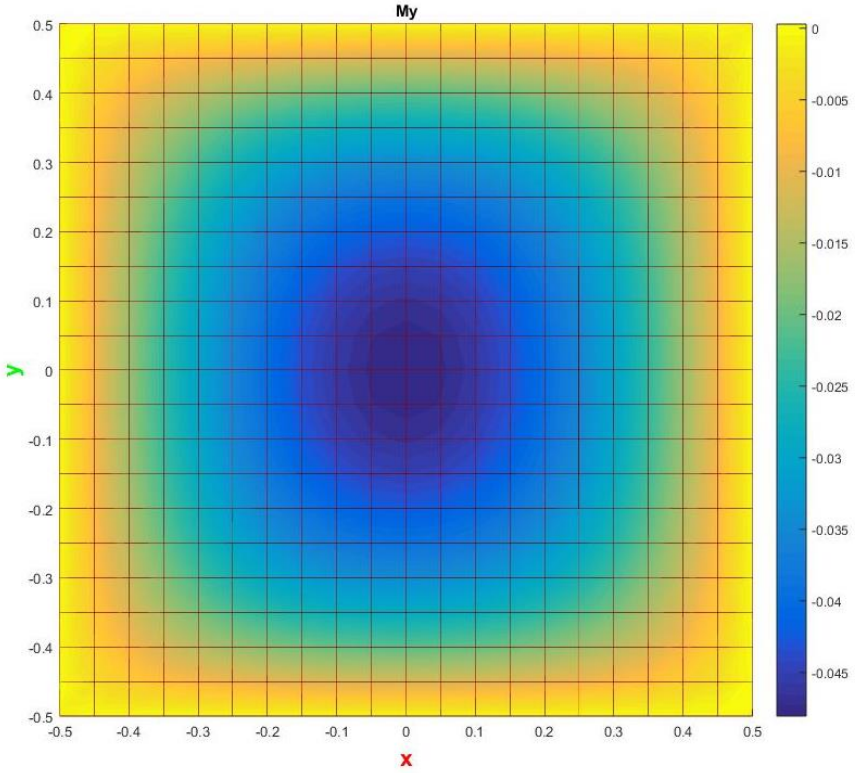


Figura 6.1-20. Momento fletor My. Fonte: Elaborado pelo autor.

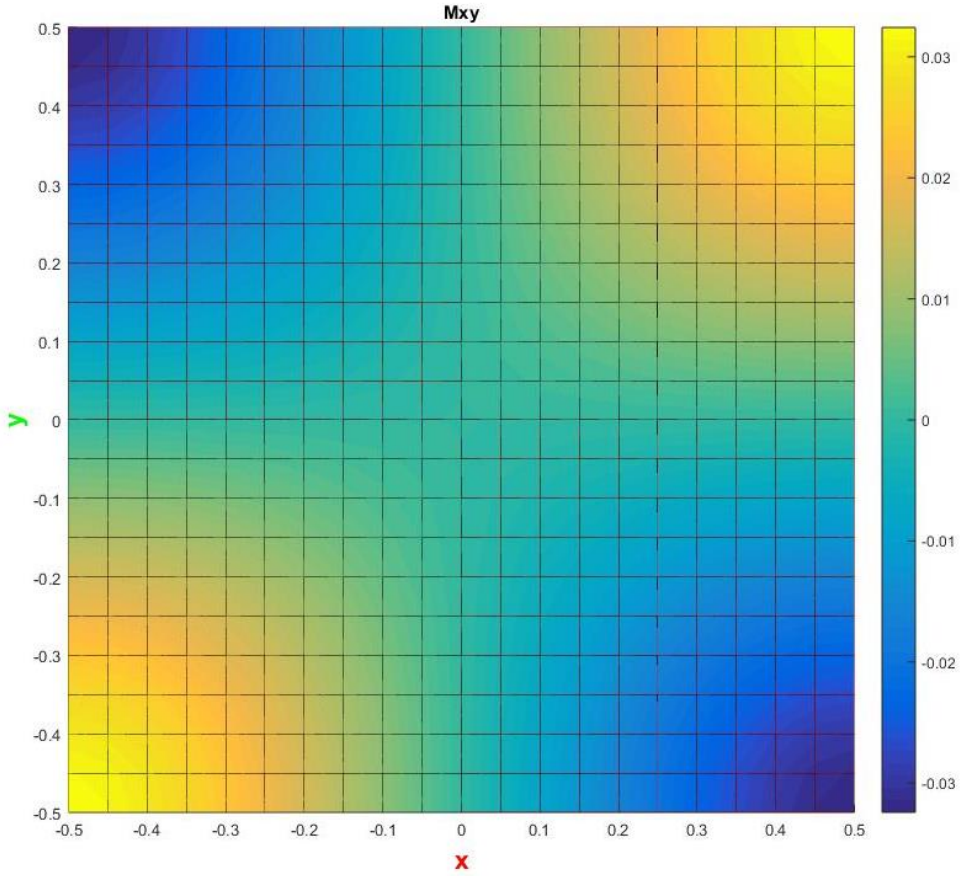


Figura 6.1-21. Momento torçor Mxy. Fonte: Elaborado pelo autor.

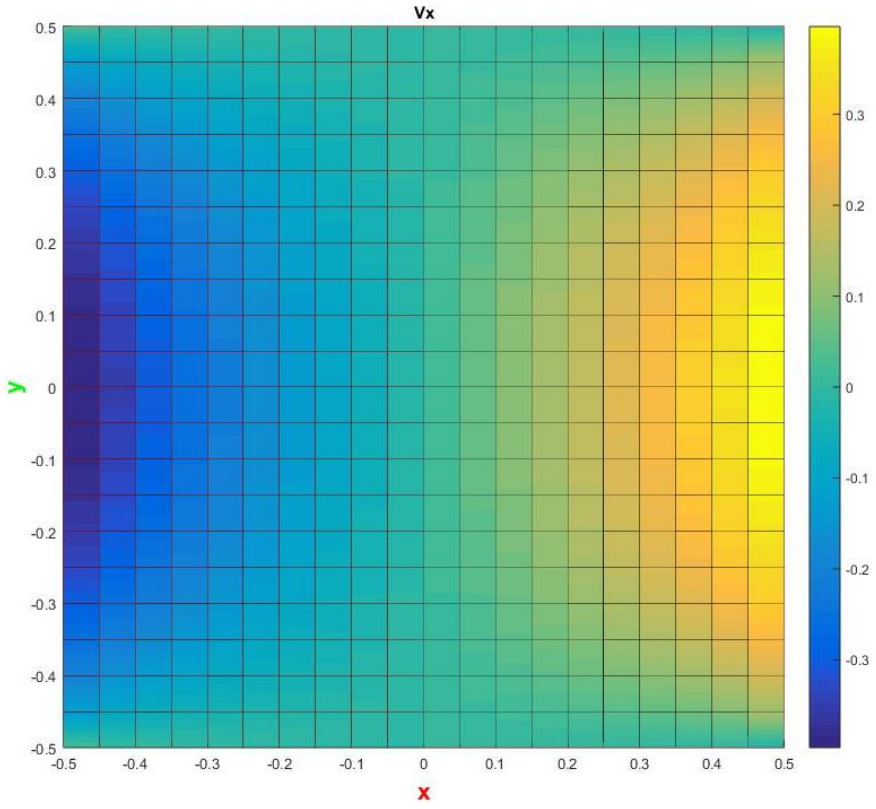


Figura 6.1-22. Esforço cortante Vx. Fonte: Elaborado pelo autor.

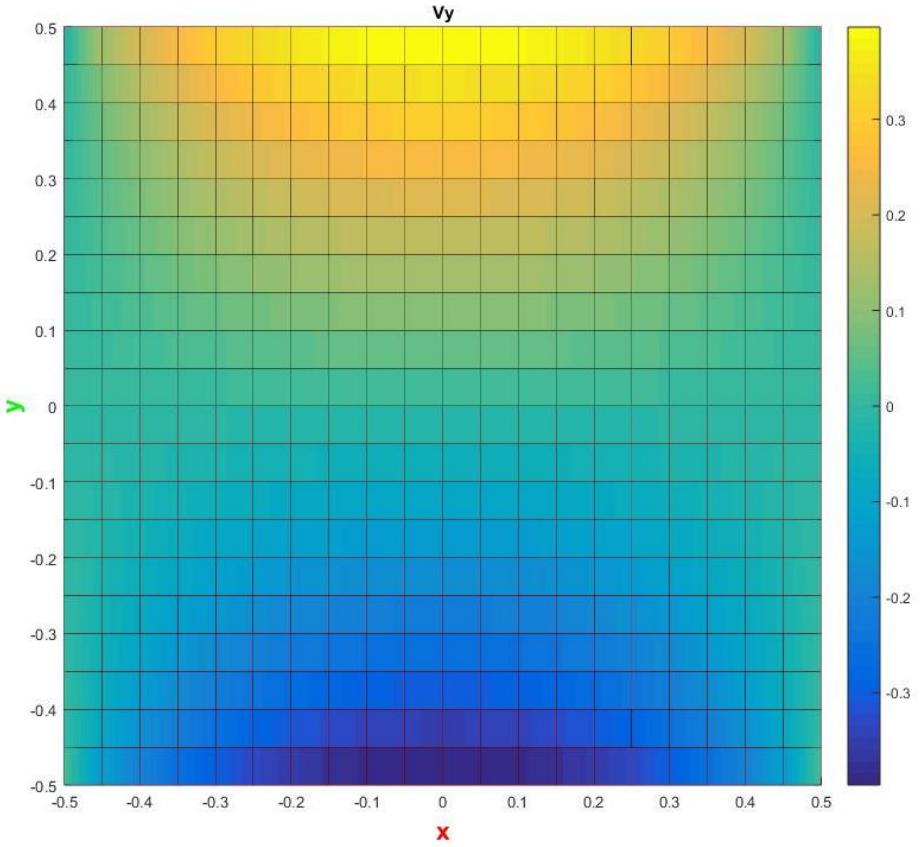


Figura 6.1-23. Esforço cortante Vy. Fonte: Elaborado pelo autor.

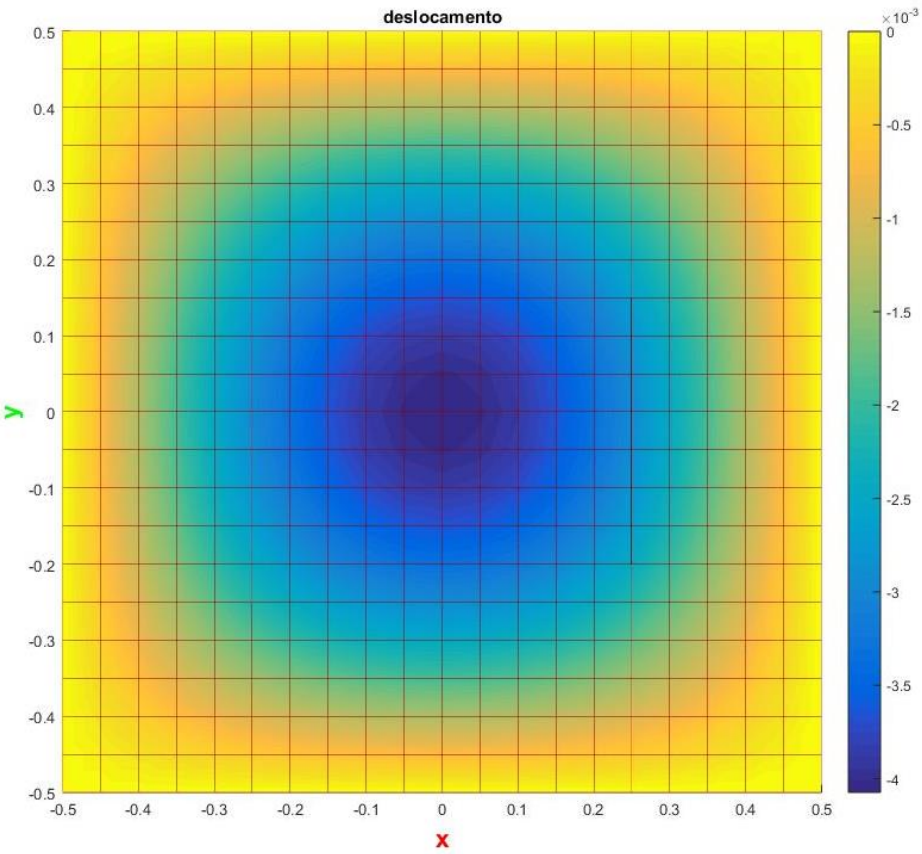


Figura 6.1-24. Deslocamento vertical. Fonte: Elaborado pelo autor.

Tabela 5 – esforços em uma placa com os quatro lados apoiados

Tipo de esforço	Valor
Deslocamento máximo	-4.0731e-03 m
Momento fletor positivo máximo (Mx)	-4.8050e-02 kN.m
Momento fletor negativo máximo (Mx)	2.8515e-04 kN.m
Momento fletor positivo máximo (My)	-4.8050e-02 kN.m
Momento fletor negativo máximo (My)	2.8515e-04 kN.m
Momento torçor positivo máximo (Mxy)	3.2429e-02 kN.m
Momento torçor negativo máximo (Mxy)	-3.2429e-02 kN.m
Esforço cortante positivo máximo (Vx)	3.9655e-01 kN/m
Esforço cortante negativo máximo (Vx)	- 3.9655e-01 kN/m
Esforço cortante positivo máximo (Vy)	3.9655e-01 kN/m
Esforço cortante negativo máximo (Vy)	- 3.9655e-01 kN/m

Fonte: Elaborado pelo autor

6.1.5 Duas bordas opostas livres e duas bordas engastadas

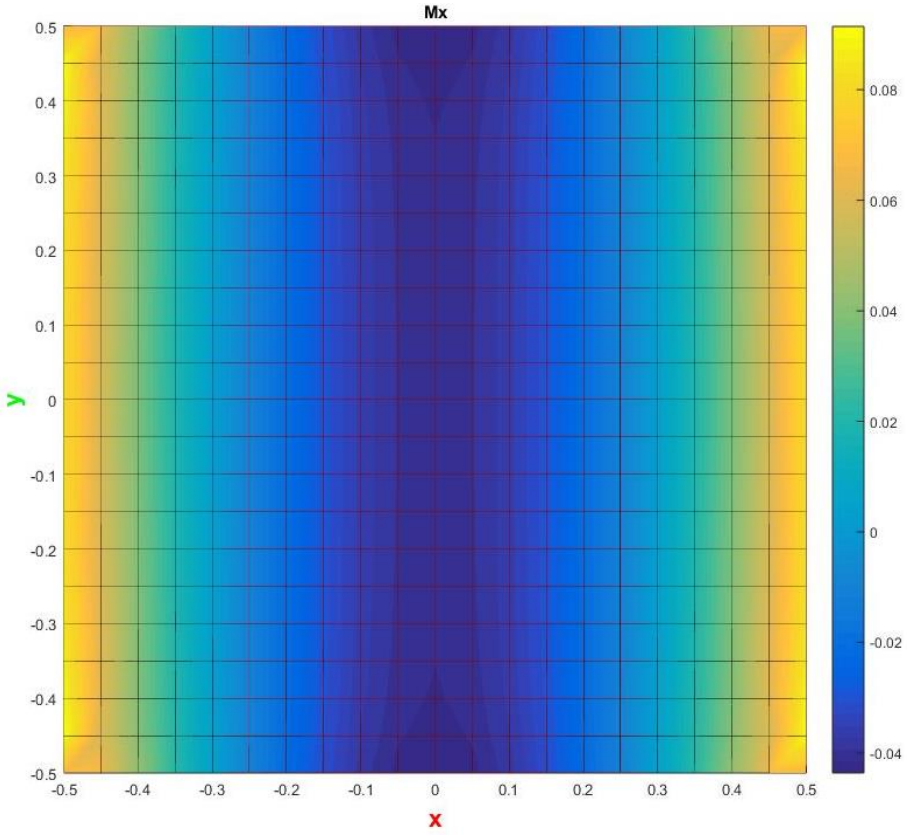


Figura 6.1-25. Momento fletor Mx. Fonte: Elaborado pelo autor.

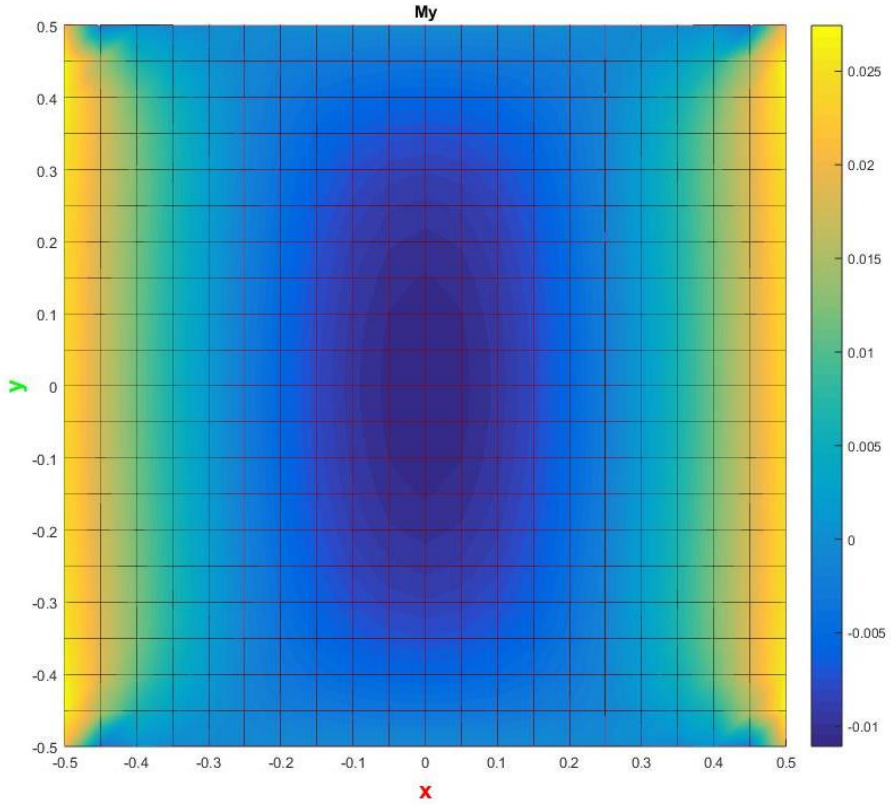


Figura 6.1-26. Momento fletor My. Fonte: Elaborado pelo autor.

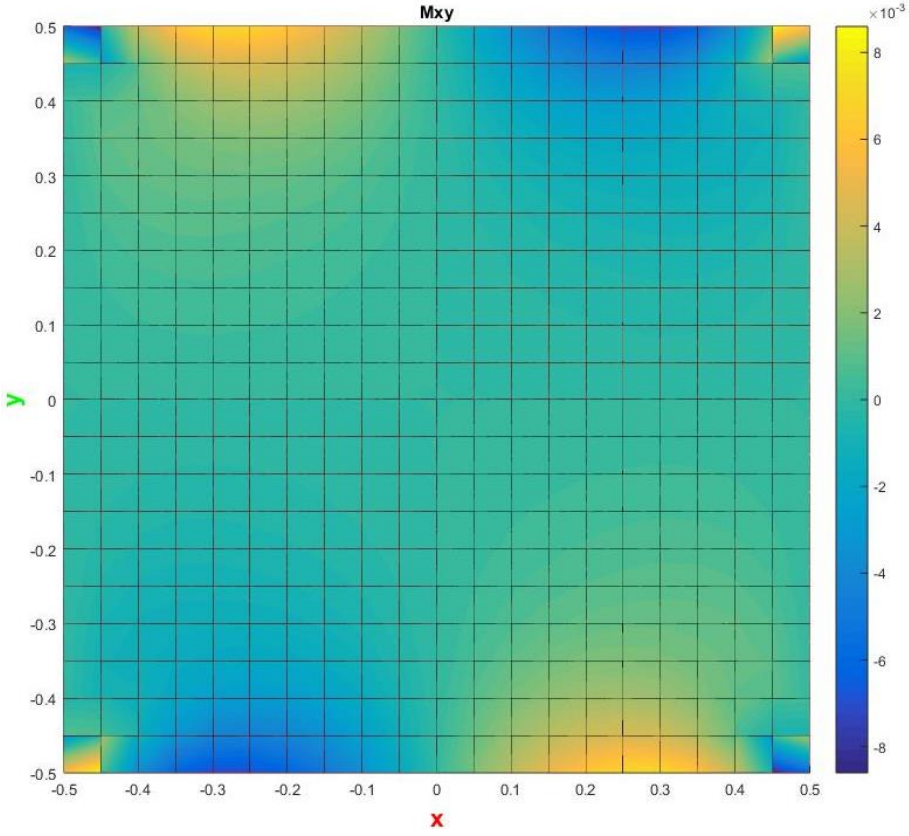


Figura 6.1-27. Momento torçor Mxy. Fonte: Elaborado pelo autor.

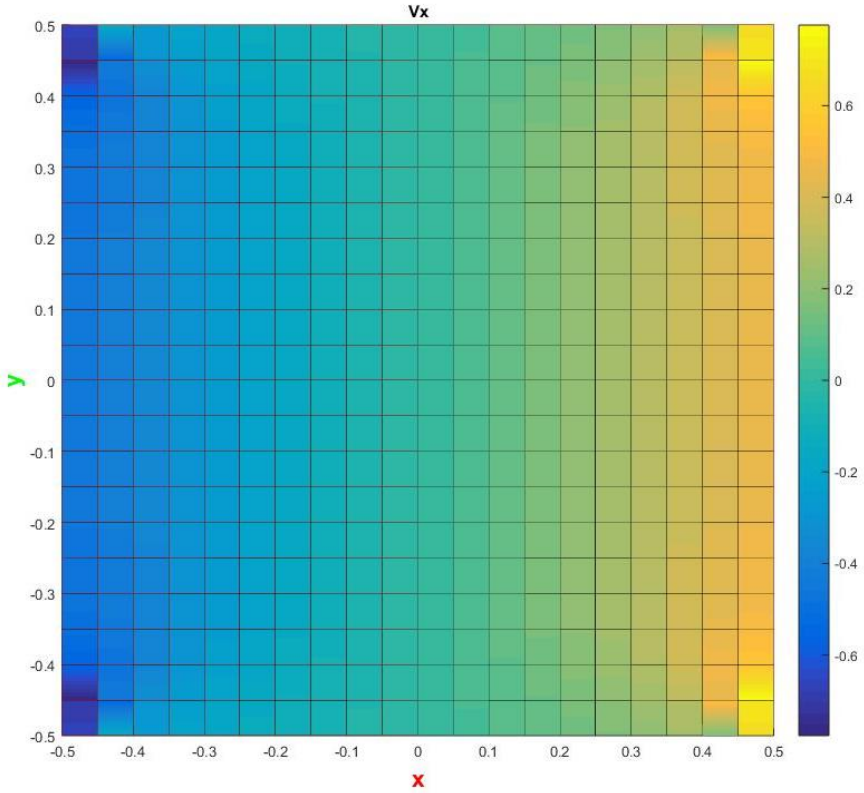


Figura 6.1-28. Esforço cortante Vx. Fonte: Elaborado pelo autor.

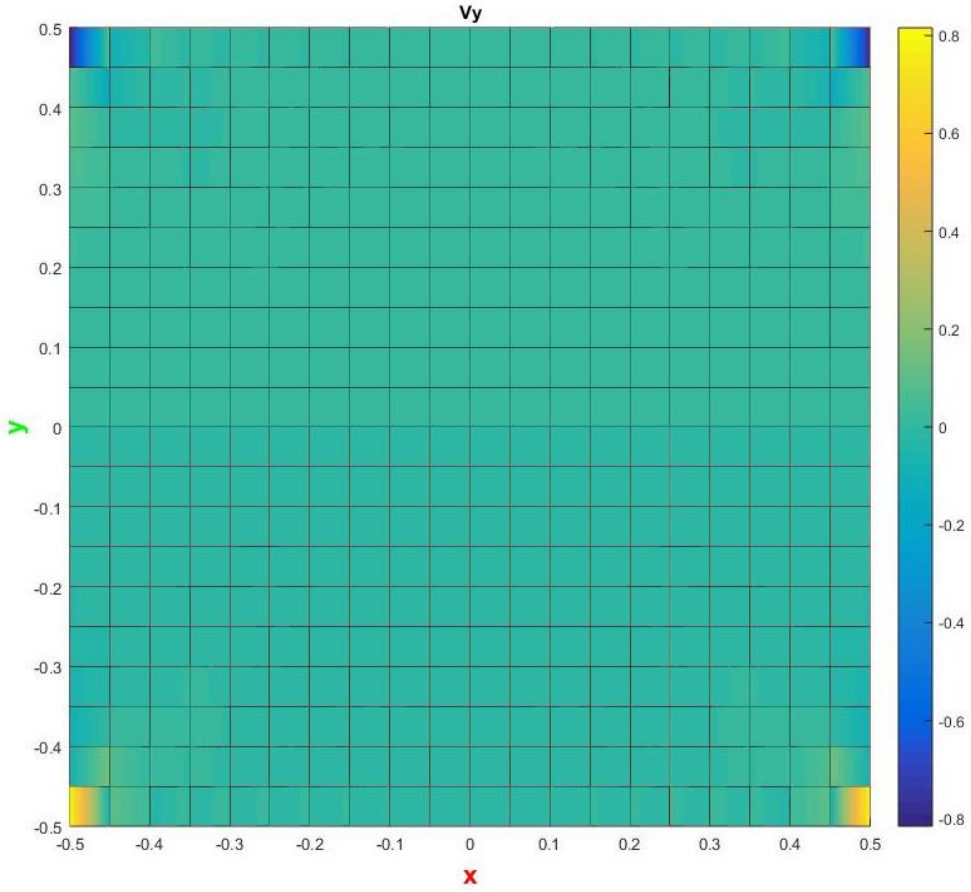


Figura 6.1-29. Esforço cortante Vy. Fonte: Elaborado pelo autor.

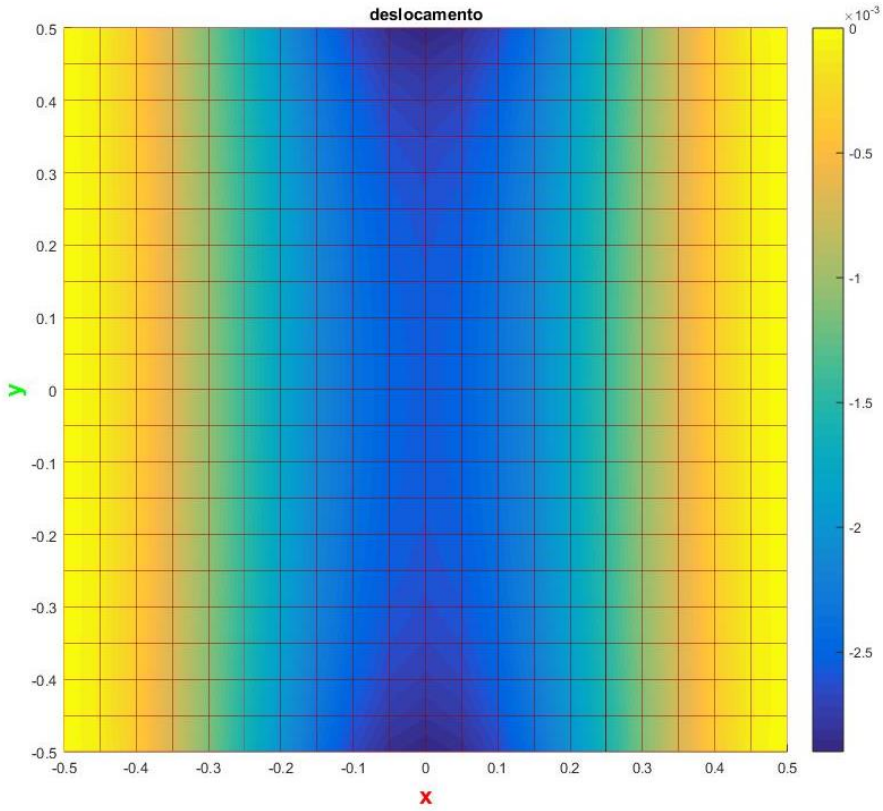


Figura 6.1-30. Deslocamento vertical. Fonte: Elaborado pelo autor.

Tabela 6 – esforços em uma placa com duas bordas opostas livres e bordas engastadas

Tipo de esforço	Valor
Deslocamento máximo	-2.8980e-03 m
Momento fletor positivo máximo (Mx)	-4.3627e-02 kN.m
Momento fletor negativo máximo (Mx)	9.1474e-02 kN.m
Momento fletor positivo máximo (My)	-1.1069e-02 kN.m
Momento fletor negativo máximo (My)	2.7442e-02 kN.m
Momento torçor positivo máximo (Mxy)	8.6000e-03 kN.m
Momento torçor negativo máximo (Mxy)	-8.6000e-03 kN.m
Esforço cortante positivo máximo (Vx)	7.7617e-01 kN/m
Esforço cortante negativo máximo (Vx)	-7.7617e-01 kN/m
Esforço cortante positivo máximo (Vy)	8.1585e-01 kN/m
Esforço cortante negativo máximo (Vy)	-8.1585e-01 kN/m

Fonte: Elaborado pelo autor

6.1.6 Duas bordas adjacentes livres e duas bordas engastadas

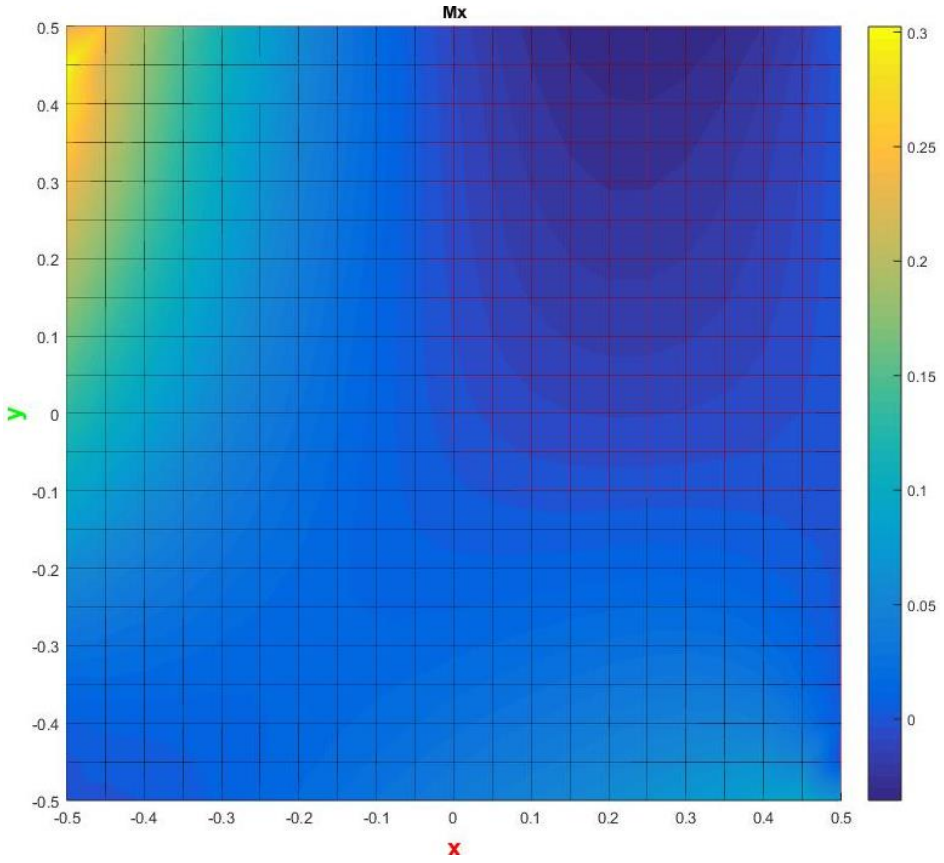


Figura 6.1-31. Momento fletor Mx. Fonte: Elaborado pelo autor.

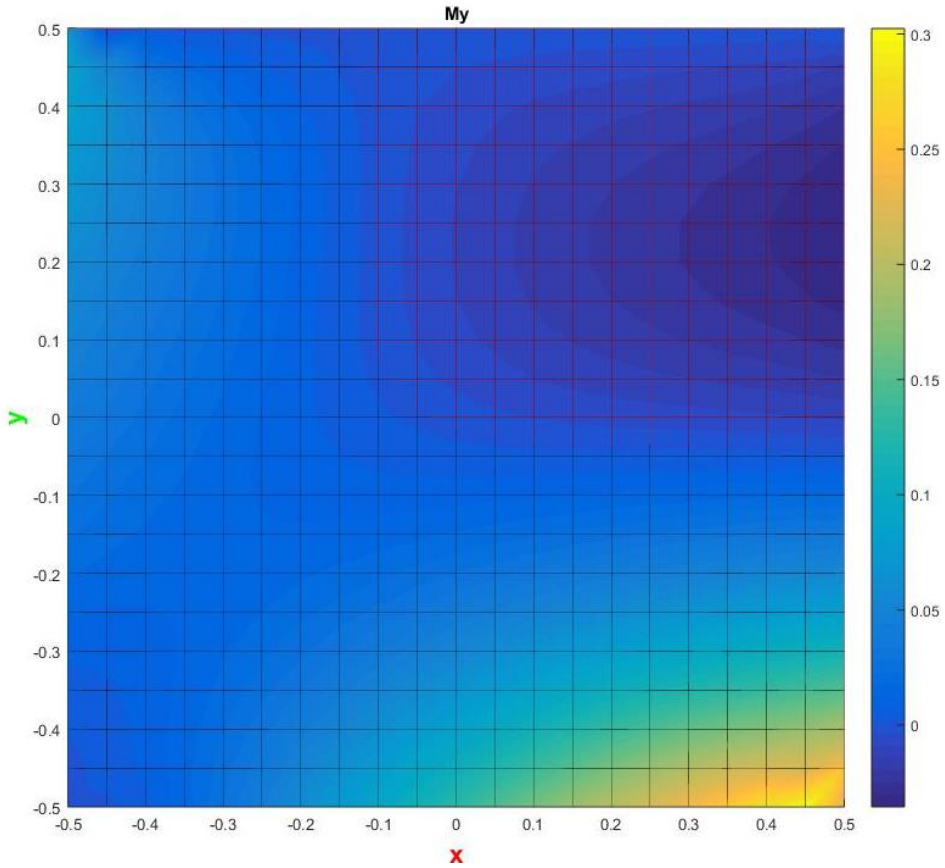


Figura 6.1-32. Momento fletor My. Fonte: Elaborado pelo autor.

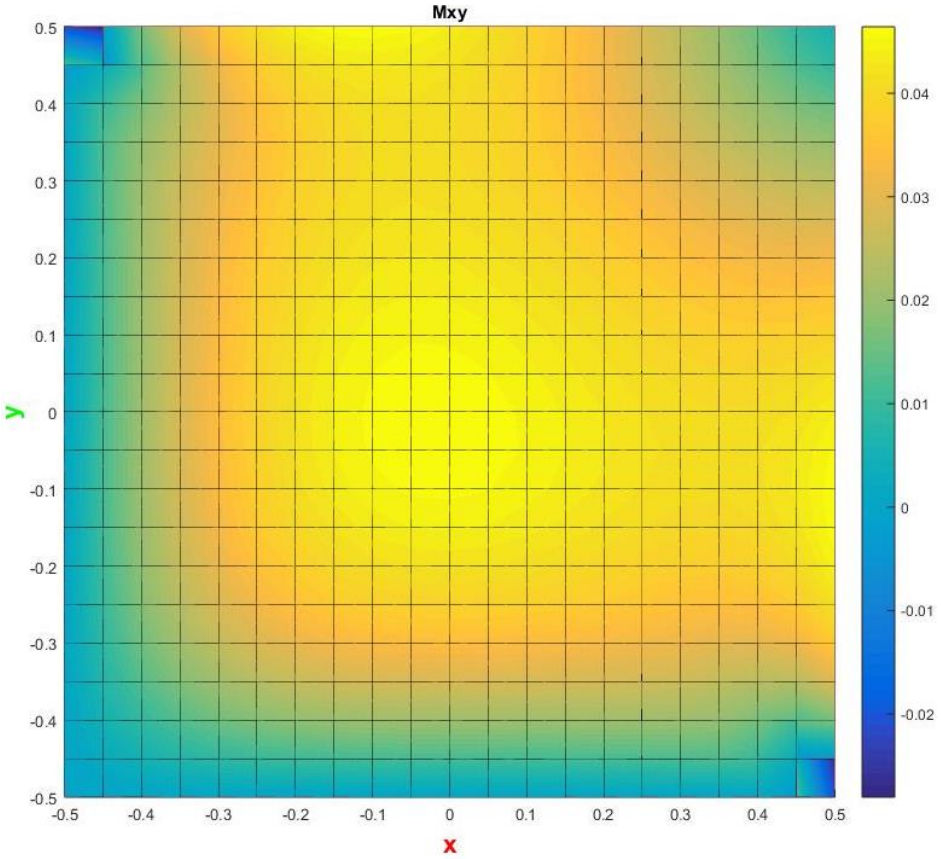


Figura 6.1-33. Momento torçor Mxy. Fonte: Elaborado pelo autor.

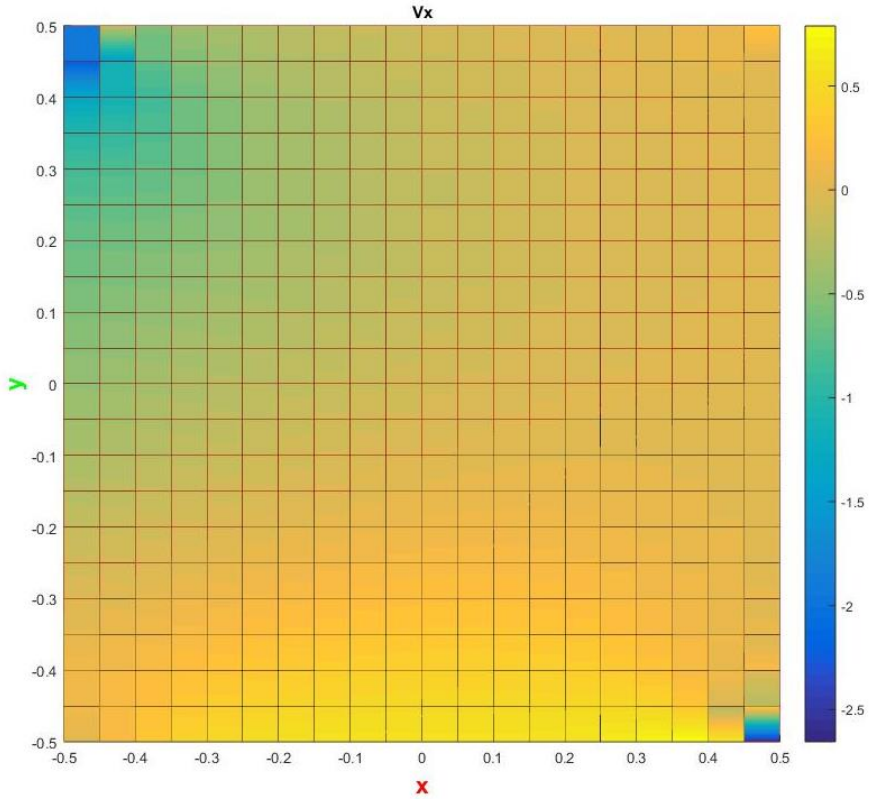


Figura 6.1-34. Esforço cortante Vx. Fonte: Elaborado pelo autor.

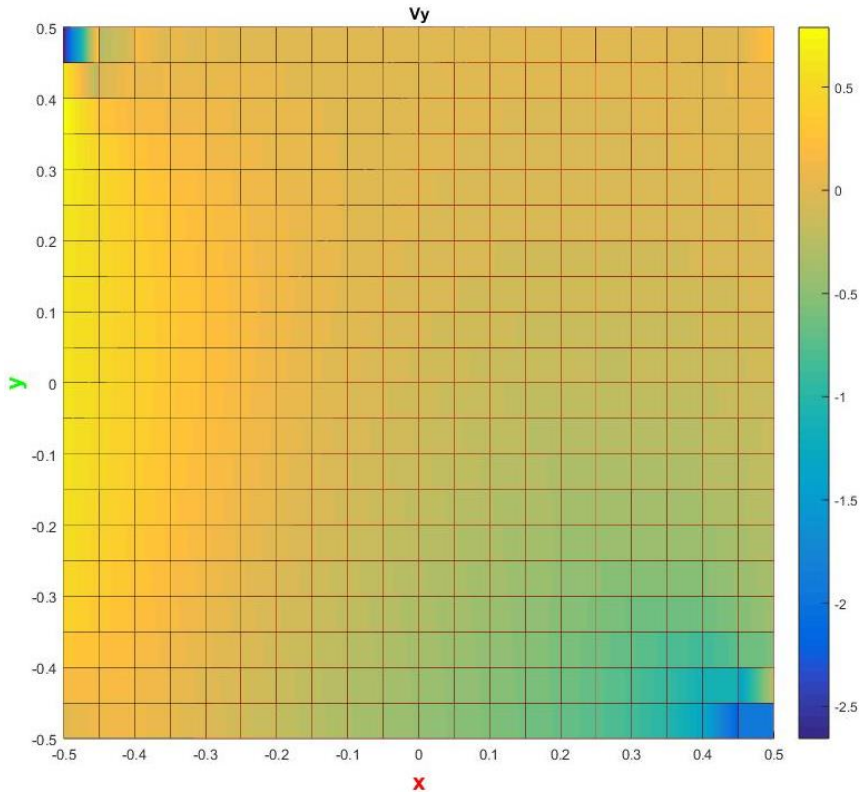


Figura 6.1-35. Esforço cortante Vy. Fonte: Elaborado pelo autor.

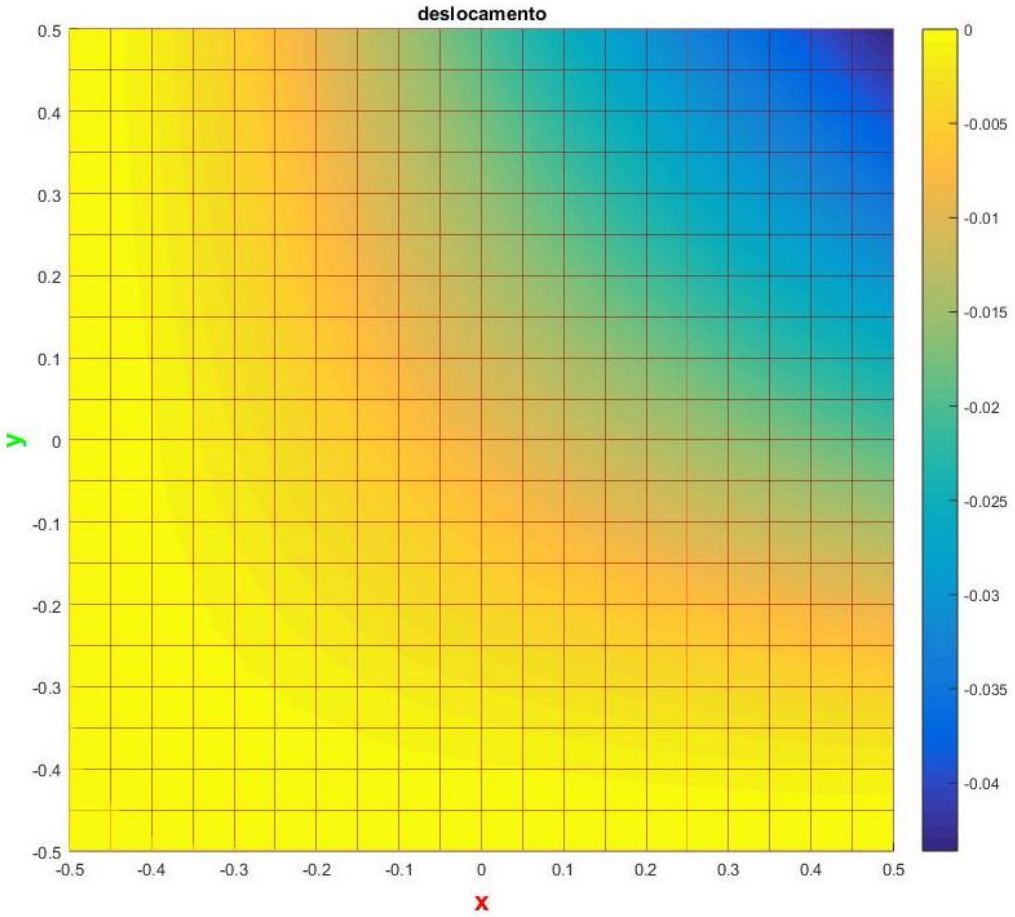


Figura 6.1-36. Deslocamento vertical. Fonte: Elaborado pelo autor.

Tabela 7 – esforços em uma placa duas bordas adjacentes livres e duas bordas engastadas

Tipo de esforço	Valor
Deslocamento máximo	-4.3602e-02 m
Momento fletor positivo máximo (Mx)	-3.5450e-02 kN.m
Momento fletor negativo máximo (Mx)	3.0239e-01 kN.m
Momento fletor positivo máximo (My)	-3.5450e-02 kN.m
Momento fletor negativo máximo (My)	3.0239e-01 kN.m
Momento torçor positivo máximo (Mxy)	4.6433e-02 kN.m
Momento torçor negativo máximo (Mxy)	-2.8037e-02 kN.m
Esforço cortante positivo máximo (Vx)	7.8879e-01 kN/m
Esforço cortante negativo máximo (Vx)	-2.6548e+00 kN/m
Esforço cortante positivo máximo (Vy)	7.8879e-01 kN/m
Esforço cortante negativo máximo (Vy)	-2.6548e+00 kN/m

Fonte: Elaborado pelo autor

6.2 Placa de Reissner-Mindlin

Da mesma forma utilizada para Kirchhoff, simulou-se várias situações de apoio de uma placa quadrada de lado 36 metros com carregamento vertical nodal de 1000 lb e espessura 0,25m. As propriedades mecânicas do material são: módulo de elasticidade ($E = 30x10^6\ kN/m^2$) e coeficiente de Poisson ($\nu = 0{,}3$). Em cada situação, a placa foi discretizada utilizando-se uma malha estruturada de 256 elementos (dezesseis ao longo da largura e dezesseis ao longo do comprimento), formando-se 867 nós no total. OBS: a implementação considera o momento fletor que traciona as fibras da face de baixo da placa com sinal positivo.

6.2.1 Os quatro cantos engastados

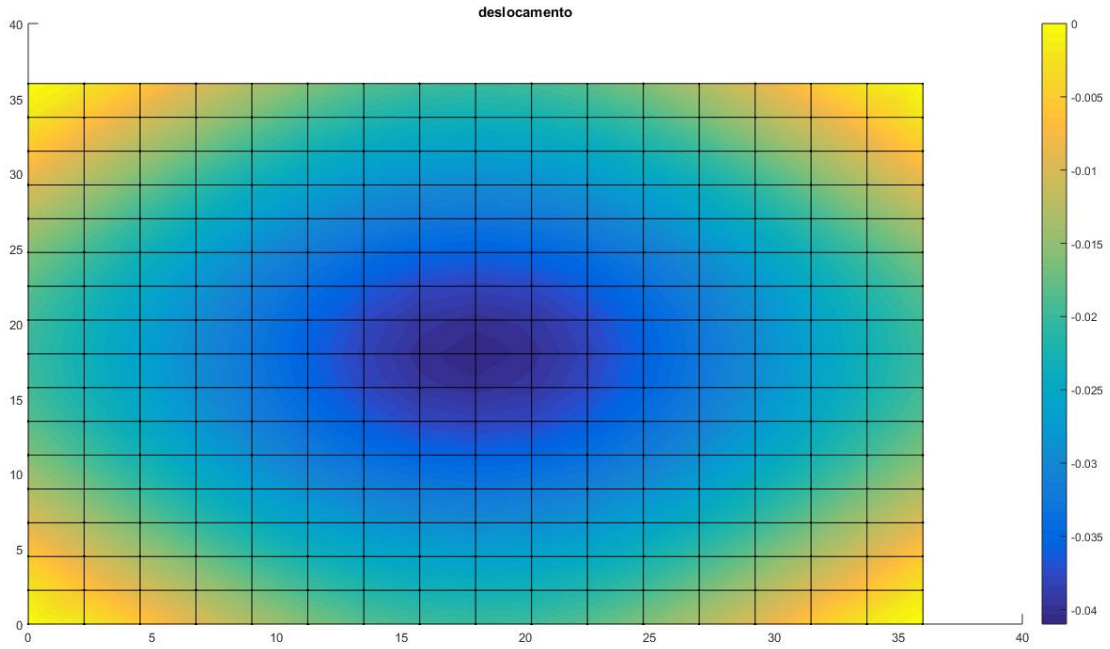


Figura 6.1-37. Deslocamento vertical. Fonte: Elaborado pelo autor

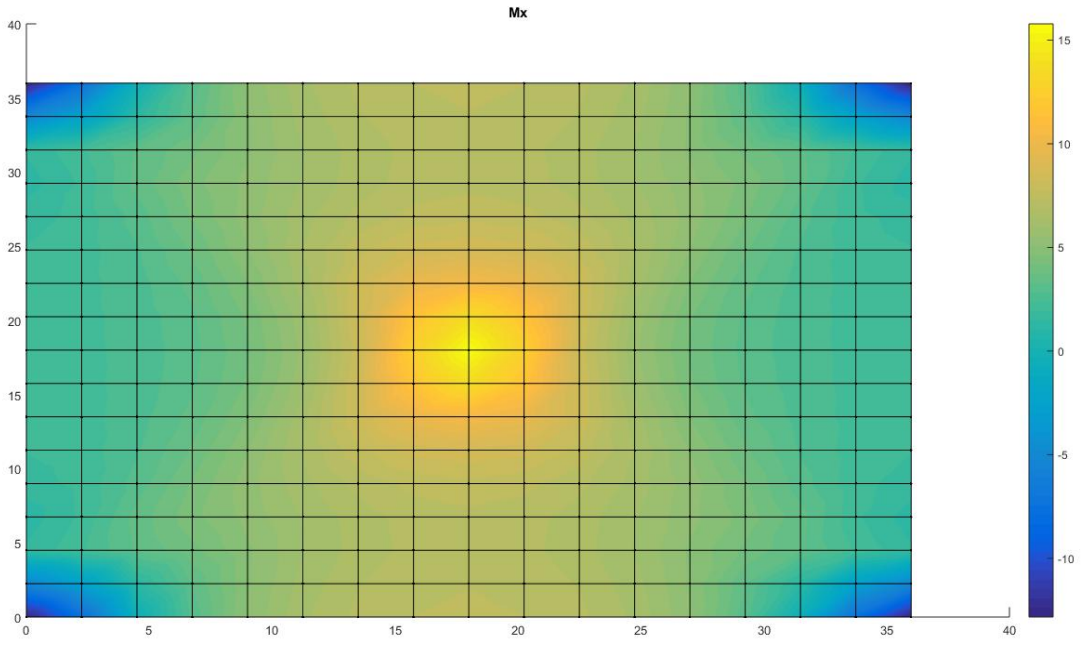


Figura 6.1-38. Momento fletor Mx. Fonte: Elaborado pelo autor

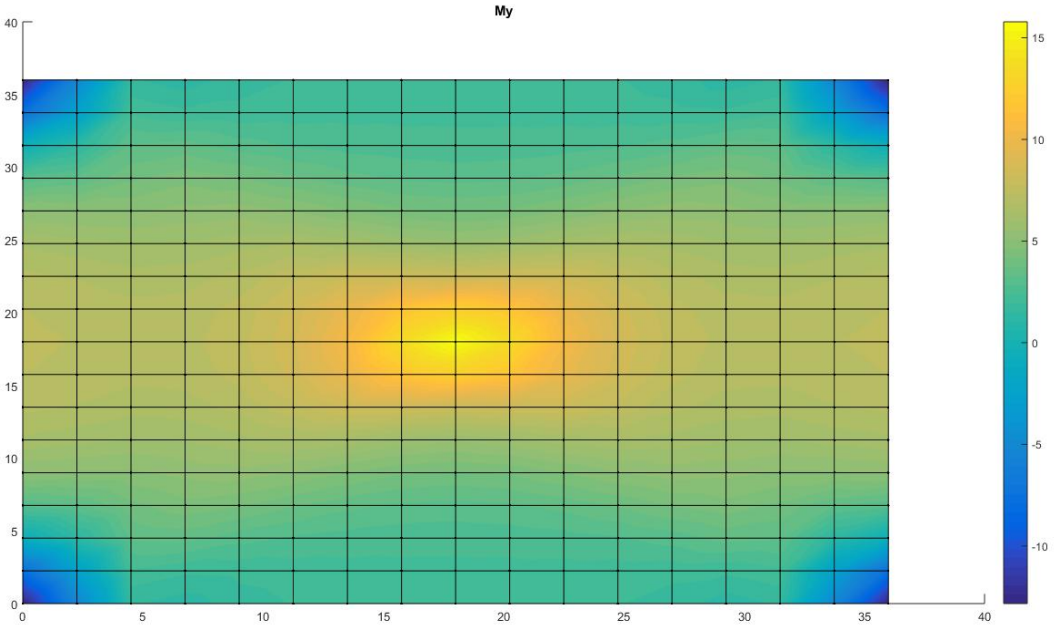


Figura 6.1-39. Momento fletor My. Fonte: Elaborado pelo autor

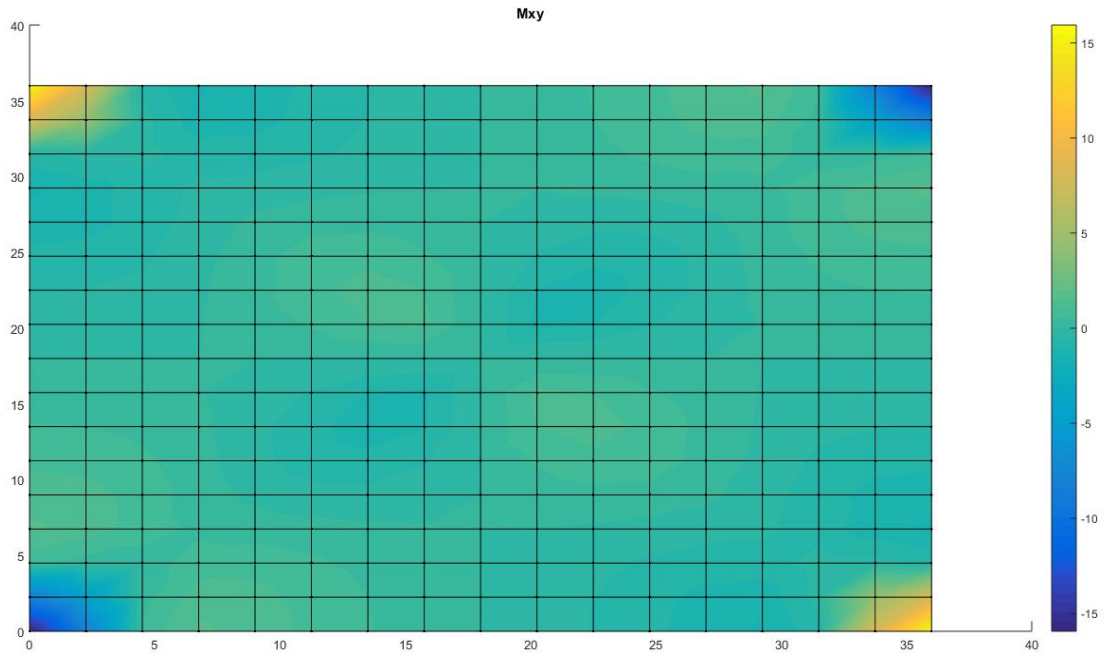


Figura 6.1-40. Momento torçor Mxy. Fonte: Elaborado pelo autor

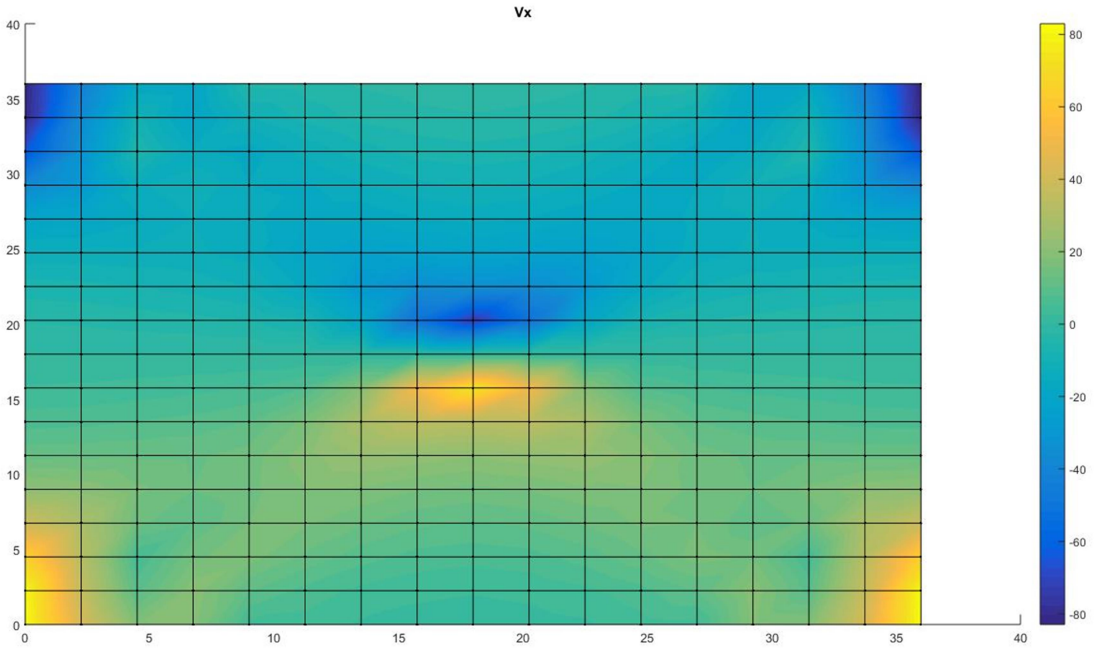


Figura 6.1-41. Esforço cortante Vx. Fonte: Elaborado pelo autor

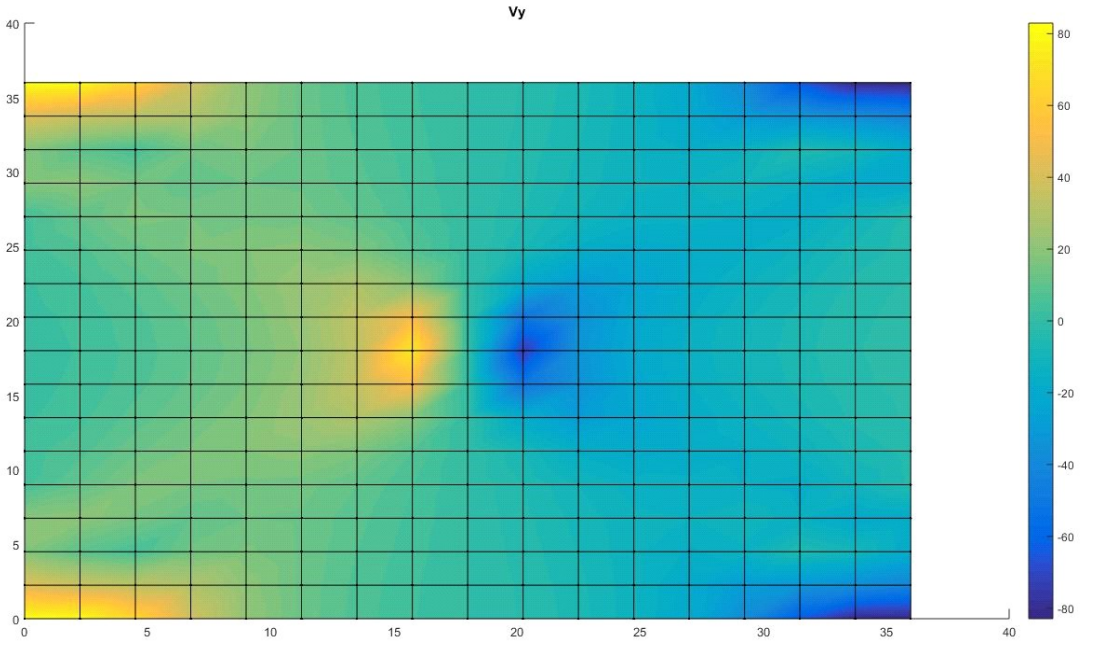


Figura 6.1-42. Esforço cortante Vy. Fonte: Elaborado pelo autor

Tabela 8 – esforços em uma placa duas bordas adjacentes livres e duas bordas engastadas

Tipo de esforço	Valor
Deslocamento máximo	-4.0984e-02 m
Momento fletor positivo máximo (Mx)	1.5785e+01 kN.m
Momento fletor negativo máximo (Mx)	-1.2830e+01 kN.m
Momento fletor positivo máximo (My)	1.5785e+01 kN.m
Momento fletor negativo máximo (My)	-1.2830e+01 kN.m
Momento torçor positivo máximo (Mxy)	1.5939e+01 kN.m
Momento torçor negativo máximo (Mxy)	-1.5939e+01 kN.m
Esforço cortante positivo máximo (Vx)	8.2957e+01 kN/m
Esforço cortante negativo máximo (Vx)	-8.2957e+01 kN/m
Esforço cortante positivo máximo (Vy)	8.2957e+01 kN/m
Esforço cortante negativo máximo (Vy)	-8.2957e+01 kN/m

Fonte: Elaborado pelo autor

6.2.2 Os quatro cantos apoiados

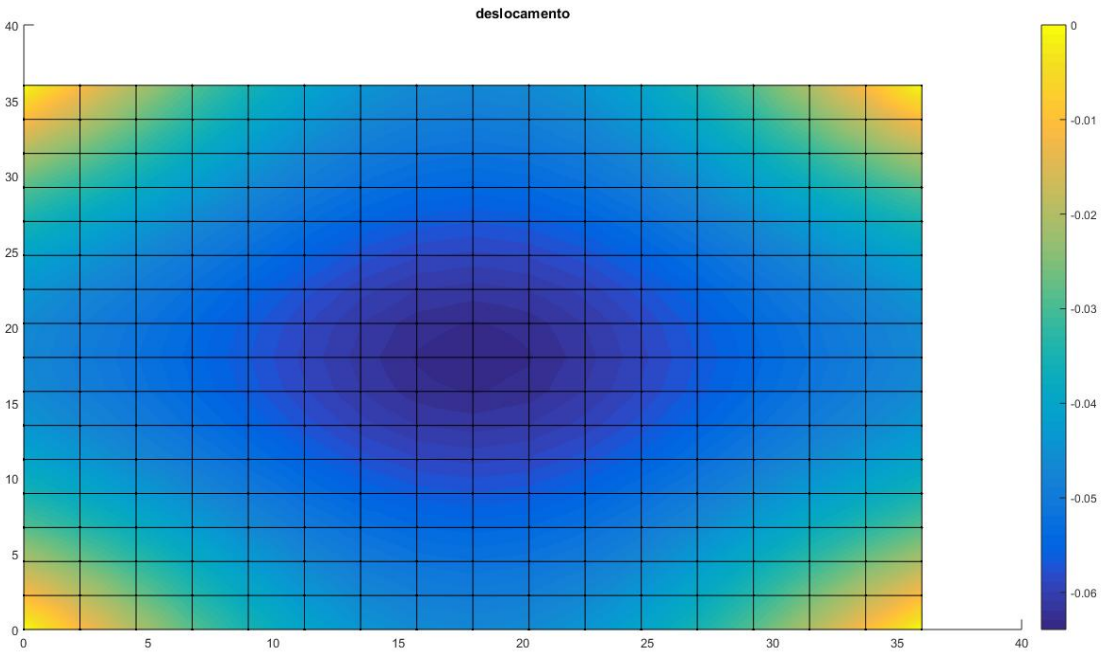


Figura 6.1-43. Deslocamento vertical. Fonte: Elaborado pelo autor

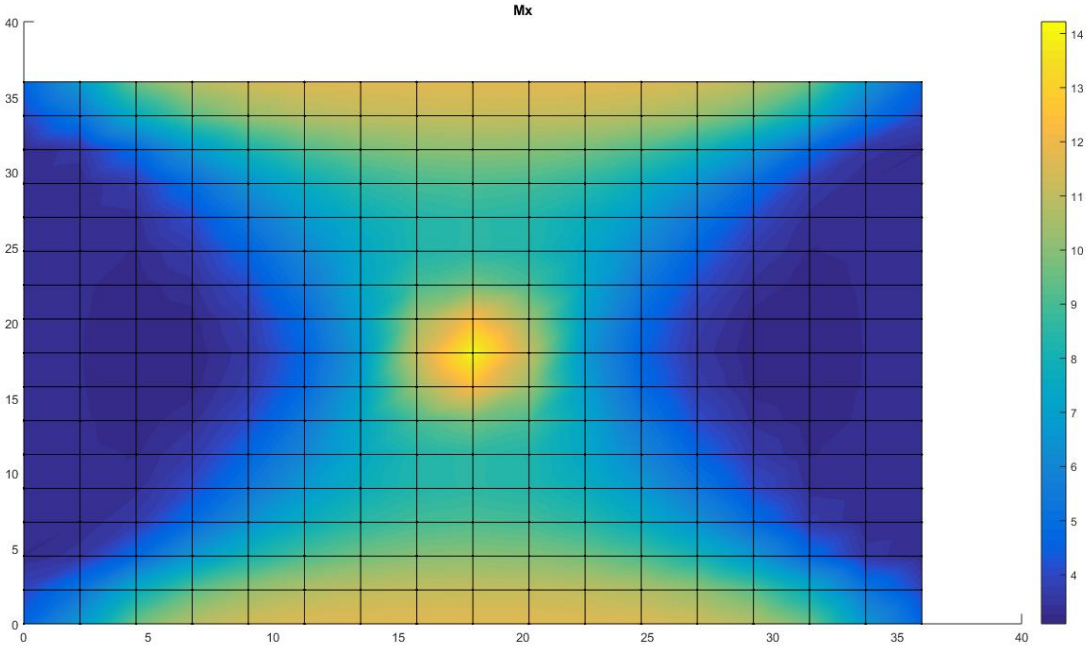


Figura 6.1-44. Momento fletor Mx. Fonte: Elaborado pelo autor

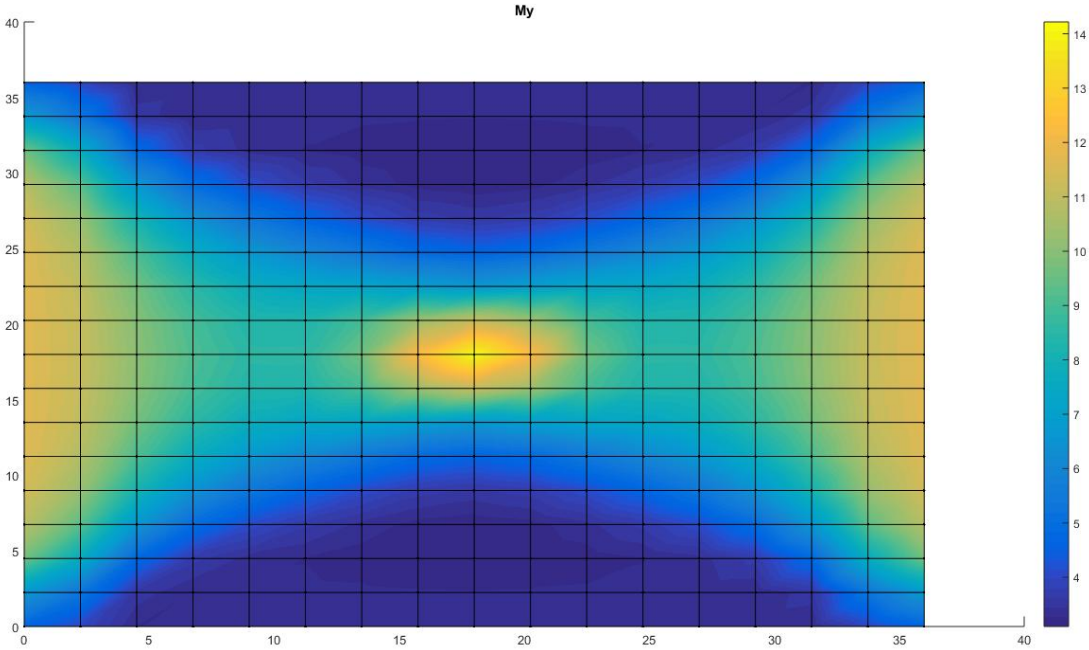


Figura 6.1-45. Momento fletor My. Fonte: Elaborado pelo autor

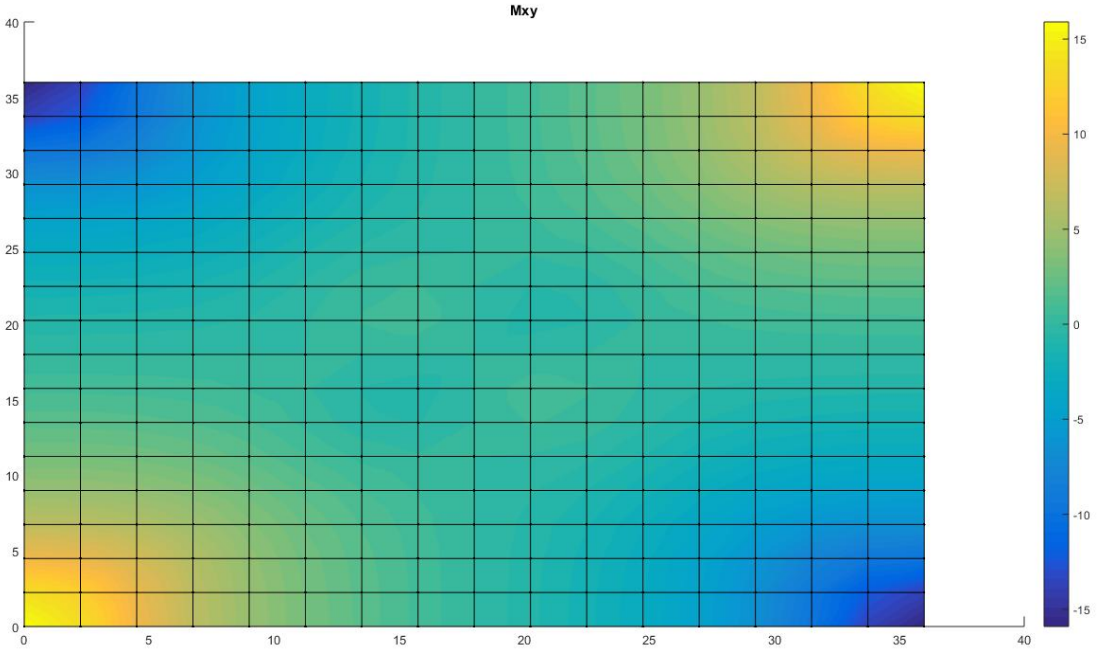


Figura 6.1-46. Momento torçor Mxy. Fonte: Elaborado pelo autor

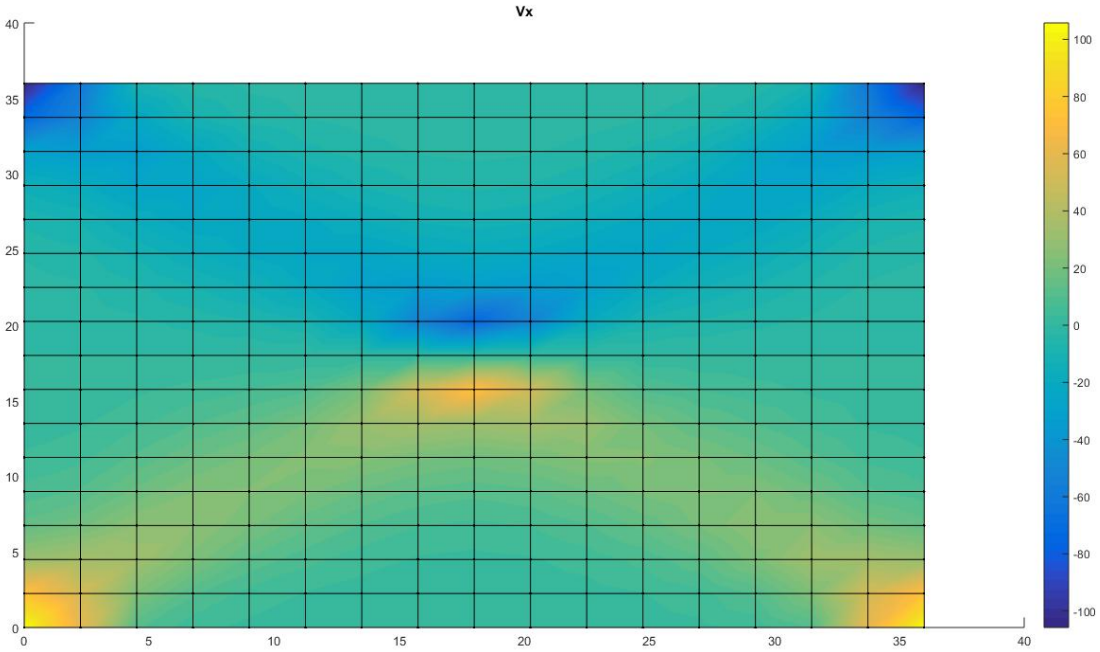


Figura 6.1-47. Esforço cortante Vx. Fonte: Elaborado pelo autor

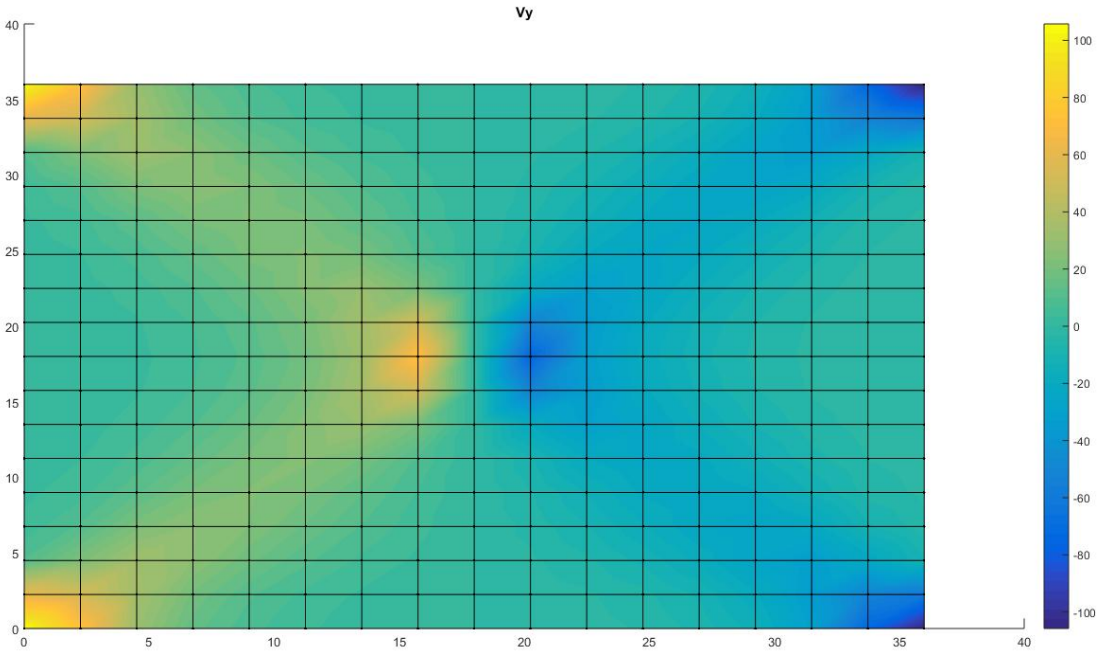


Figura 6.1-48. Esforço cortante Vy. Fonte: Elaborado pelo autor

Tabela 9 – esforços em uma placa duas bordas adjacentes livres e duas bordas engastadas

Tipo de esforço	Valor
Deslocamento máximo	-6.3916e-02 m
Momento fletor positivo máximo (Mx)	1.4218e+01 kN.m
Momento fletor negativo máximo (Mx)	3.0953e+00 kN.m
Momento fletor positivo máximo (My)	1.4218e+01 kN.m
Momento fletor negativo máximo (My)	3.0953e+00 kN.m
Momento torçor positivo máximo (Mxy)	1.5905e+01 kN.m
Momento torçor negativo máximo (Mxy)	-1.5905e+01 kN.m
Esforço cortante positivo máximo (Vx)	1.0574e+02 kN/m
Esforço cortante negativo máximo (Vx)	-1.0574e+02 kN/m
Esforço cortante positivo máximo (Vy)	1.0574e+02 kN/m
Esforço cortante negativo máximo (Vy)	-1.0574e+02 kN/m

Fonte: Elaborado pelo autor

6.2.3 Os quatro lados engastados

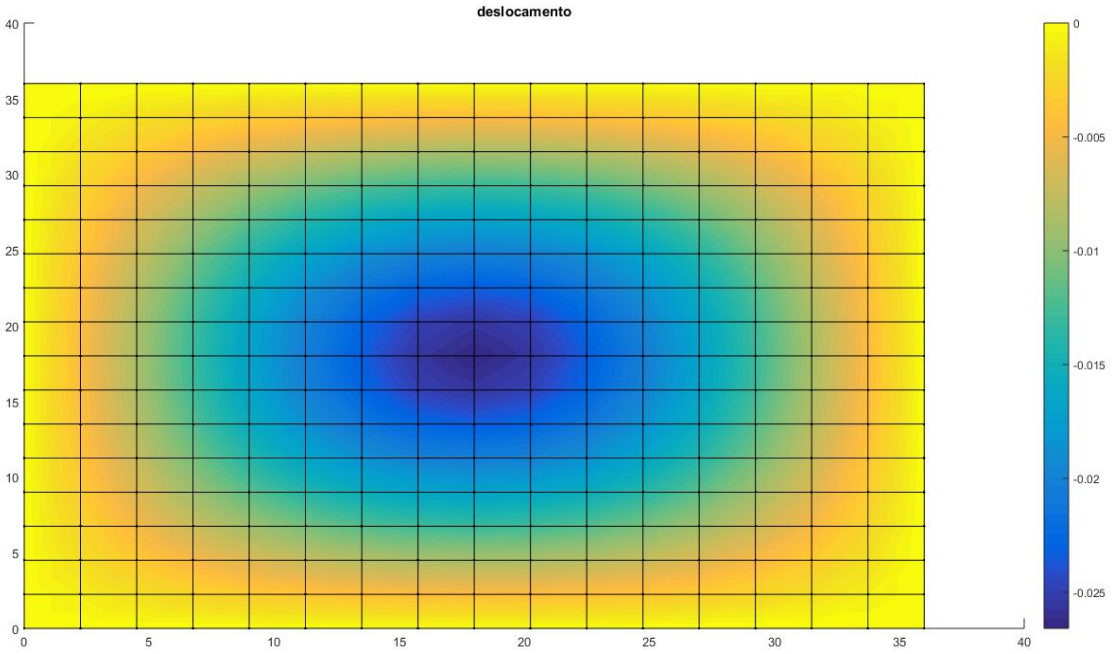


Figura 6.1-43. Deslocamento vertical. Fonte: Elaborado pelo autor

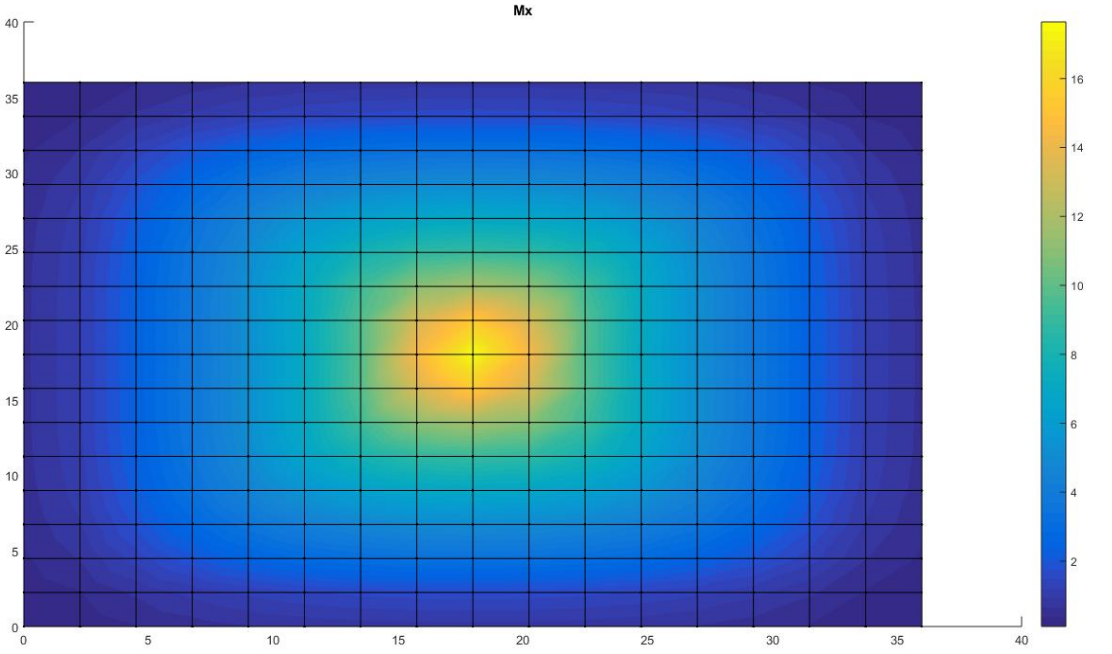


Figura 6.1-44. Momento fletor Mx. Fonte: Elaborado pelo autor

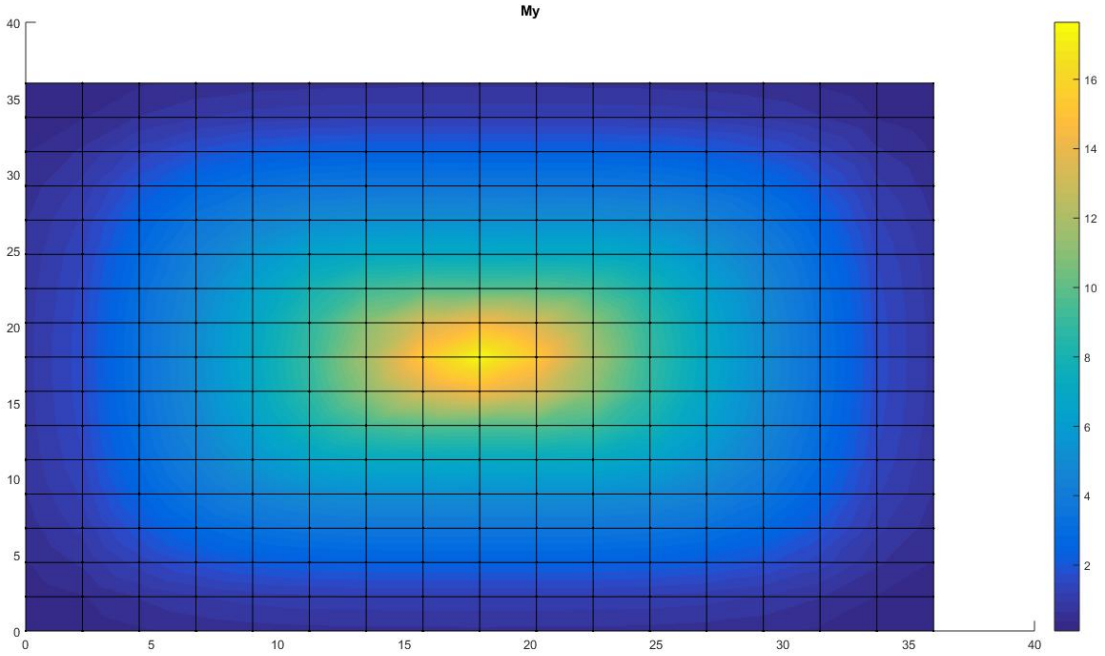


Figura 6.1-45. Momento fletor My. Fonte: Elaborado pelo autor

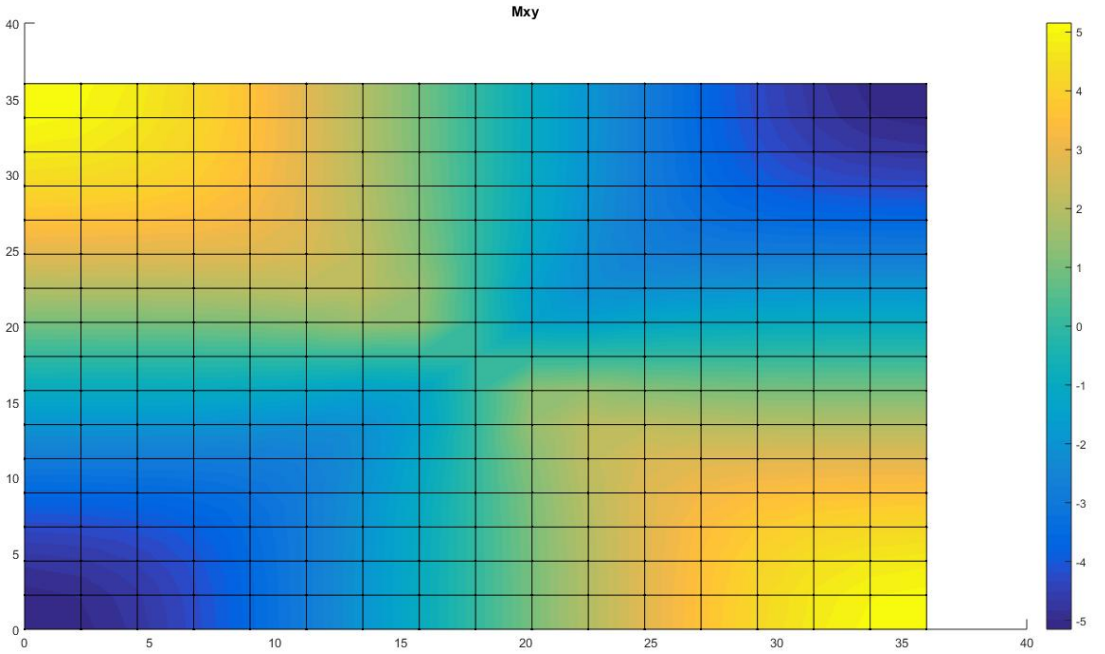


Figura 6.1-46. Momento torçor Mxy. Fonte: Elaborado pelo autor

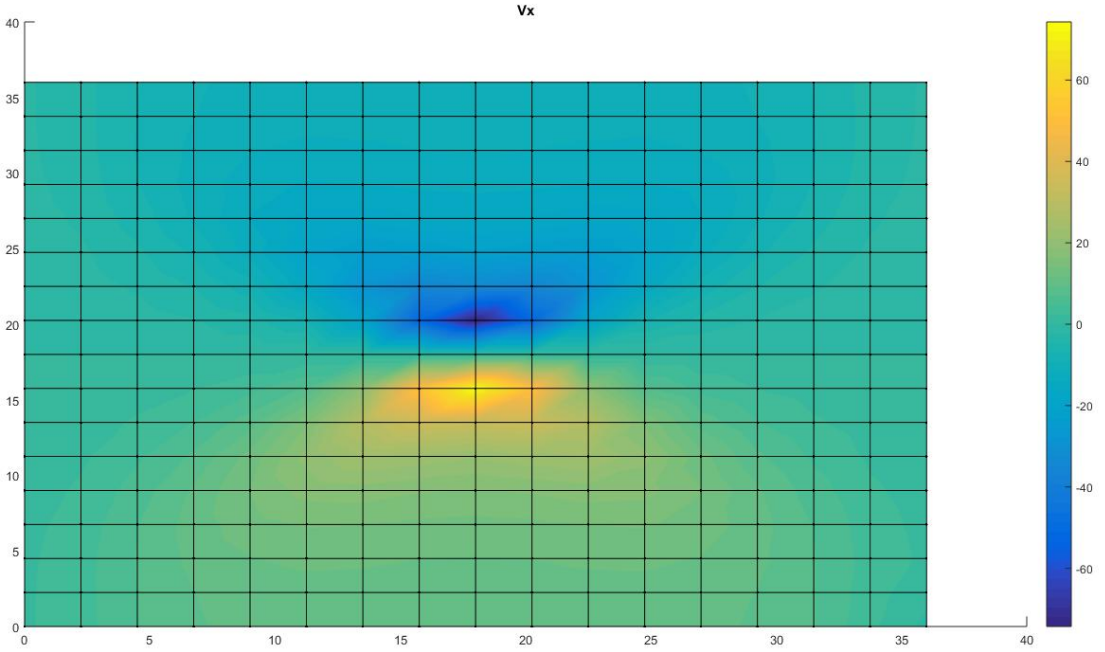


Figura 6.1-47. Esforço cortante Vx. Fonte: Elaborado pelo autor

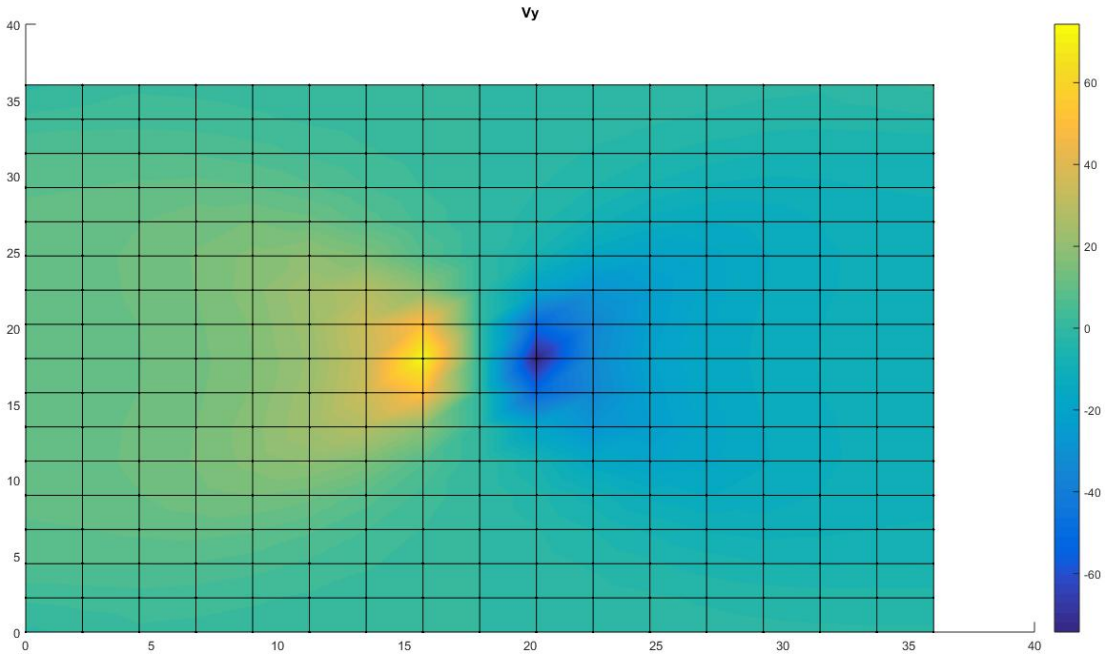


Figura 6.1-48. Esforço cortante Vy. Fonte: Elaborado pelo autor

Tabela 10 – esforços em uma placa duas bordas adjacentes livres e duas bordas engastadas

Tipo de esforço	Valor
Deslocamento máximo	-2.6588e-02 m
Momento fletor positivo máximo (Mx)	1.7642e+01 kN.m
Momento fletor negativo máximo (Mx)	1.0436e-01 kN.m
Momento fletor positivo máximo (My)	1.7642e+01 kN.m
Momento fletor negativo máximo (My)	1.0436e-01 kN.m
Momento torçor positivo máximo (Mxy)	5.1533e+00 kN.m
Momento torçor negativo máximo (Mxy)	-5.1533e+00 kN.m
Esforço cortante positivo máximo (Vx)	7.4277e+01 kN/m
Esforço cortante negativo máximo (Vx)	-7.4277e+01 kN/m
Esforço cortante positivo máximo (Vy)	7.4277e+01 kN/m
Esforço cortante negativo máximo (Vy)	-7.4277e+01 kN/m

Fonte: Elaborado pelo autor

6.2.4 Os quatro lados apoiados

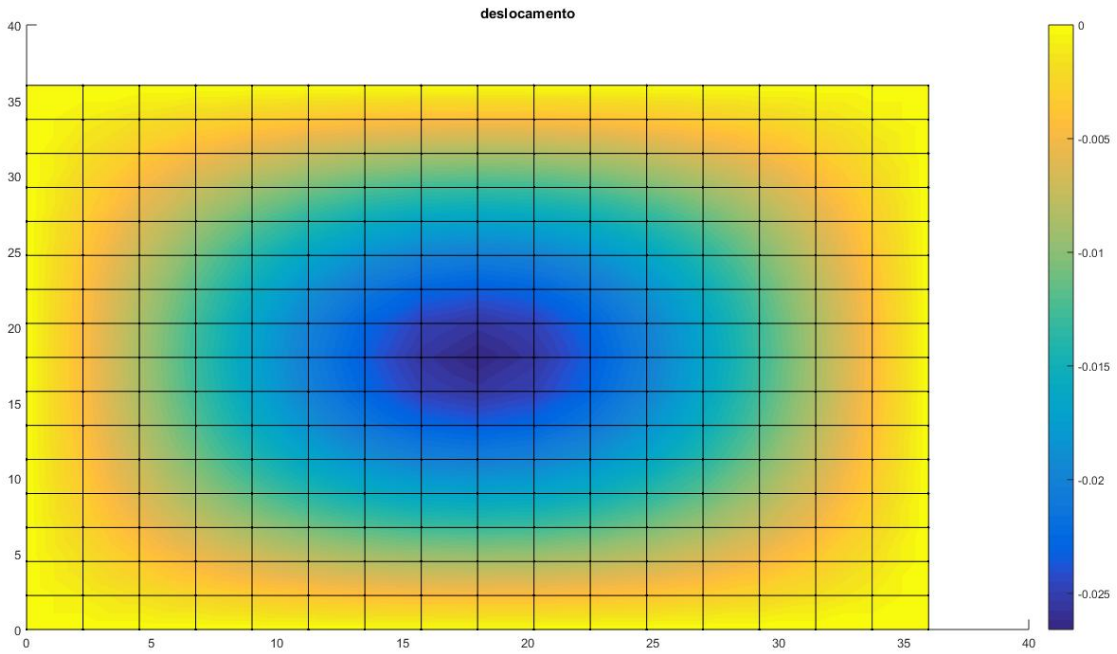


Figura 6.1-43. Deslocamento vertical. Fonte: Elaborado pelo autor

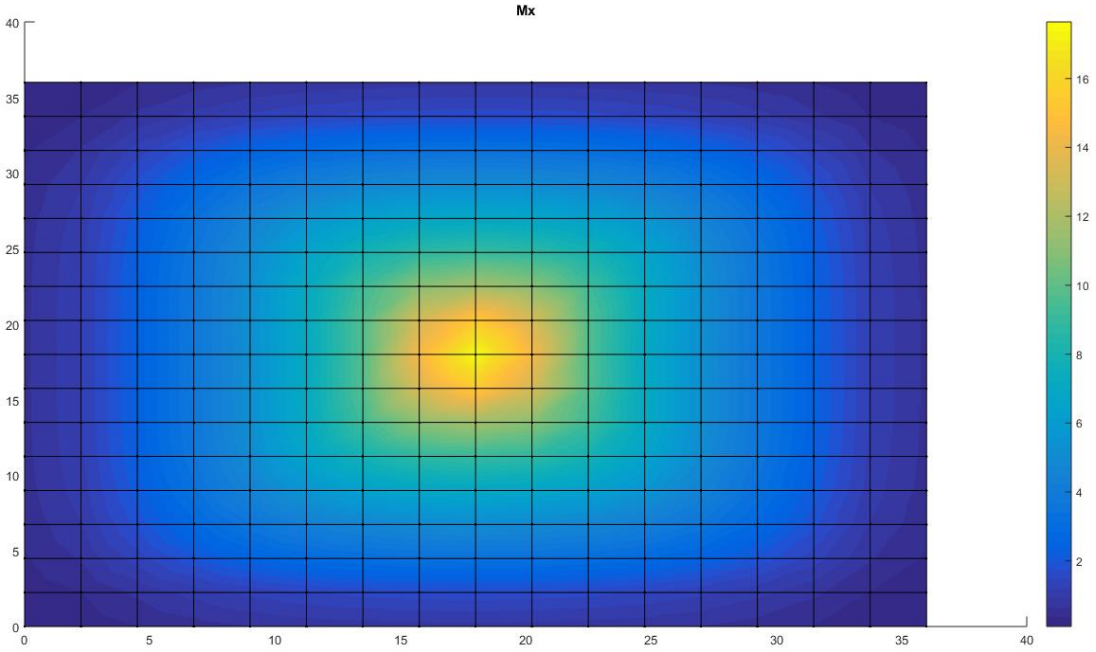


Figura 6.1-44. Momento fletor Mx. Fonte: Elaborado pelo autor

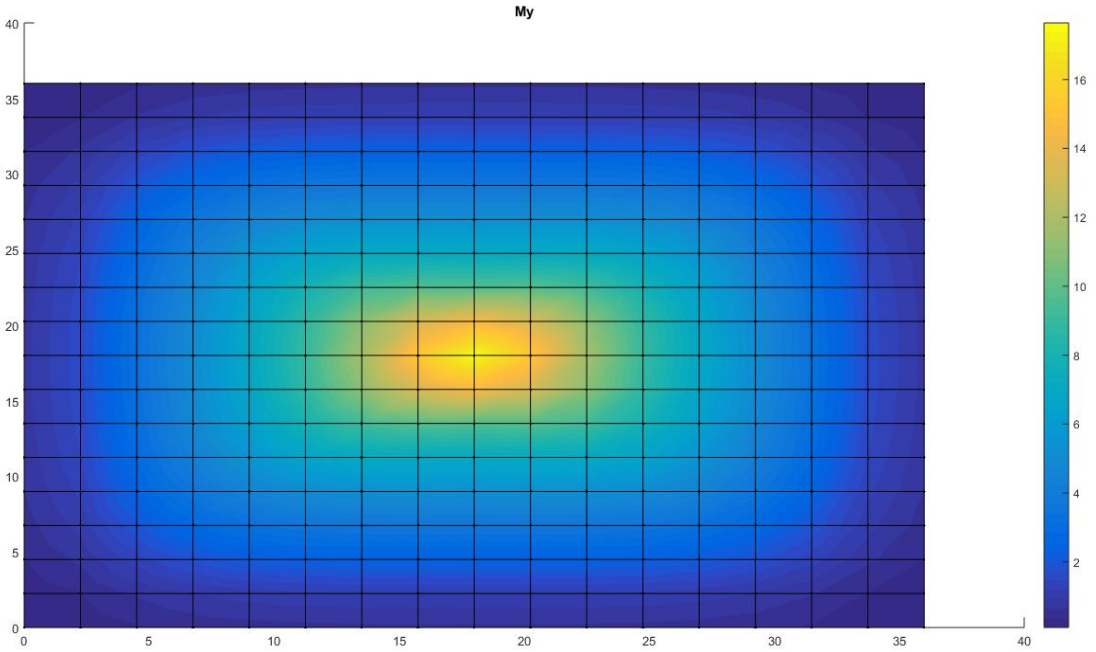


Figura 6.1-45. Momento fletor My. Fonte: Elaborado pelo autor

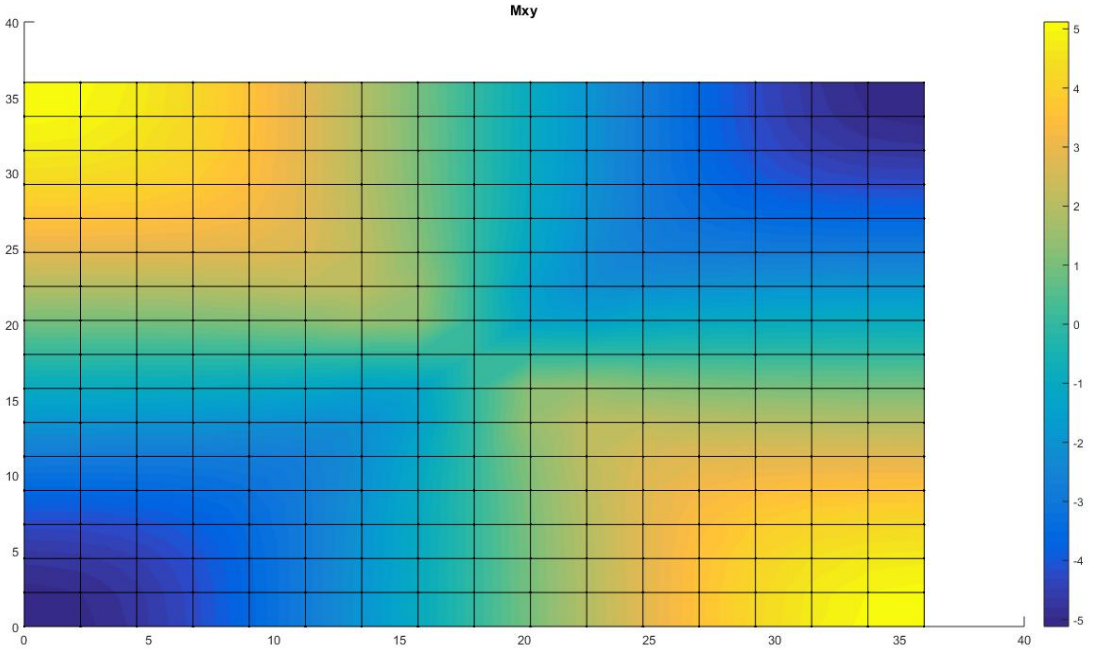


Figura 6.1-46. Momento torçor Mxy. Fonte: Elaborado pelo autor

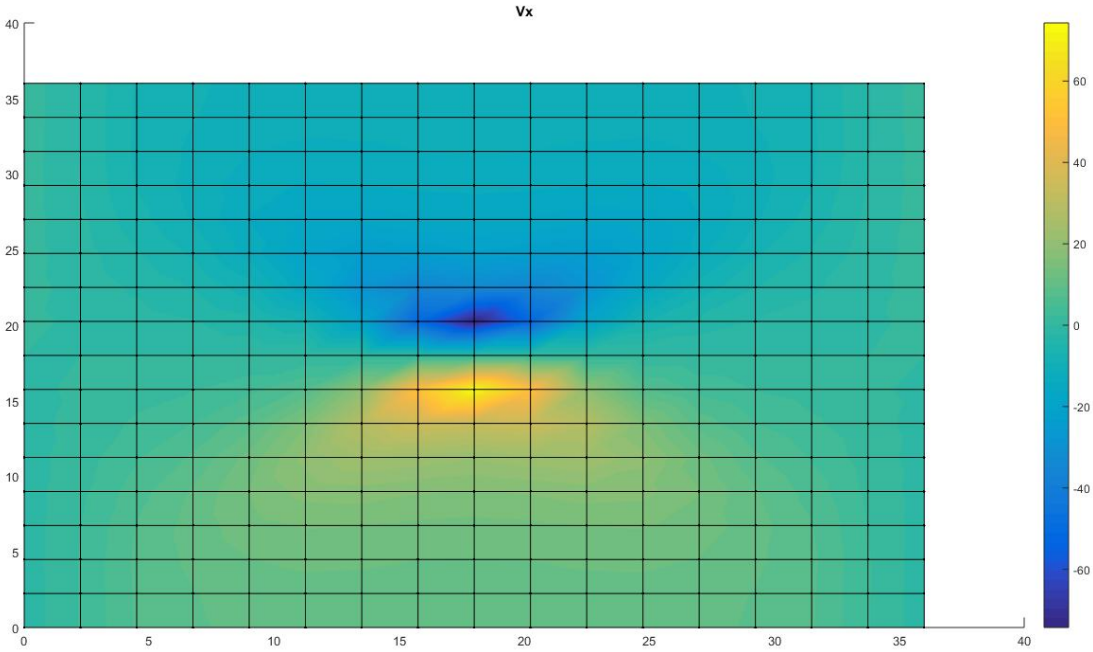


Figura 6.1-47. Esforço cortante Vx. Fonte: Elaborado pelo autor

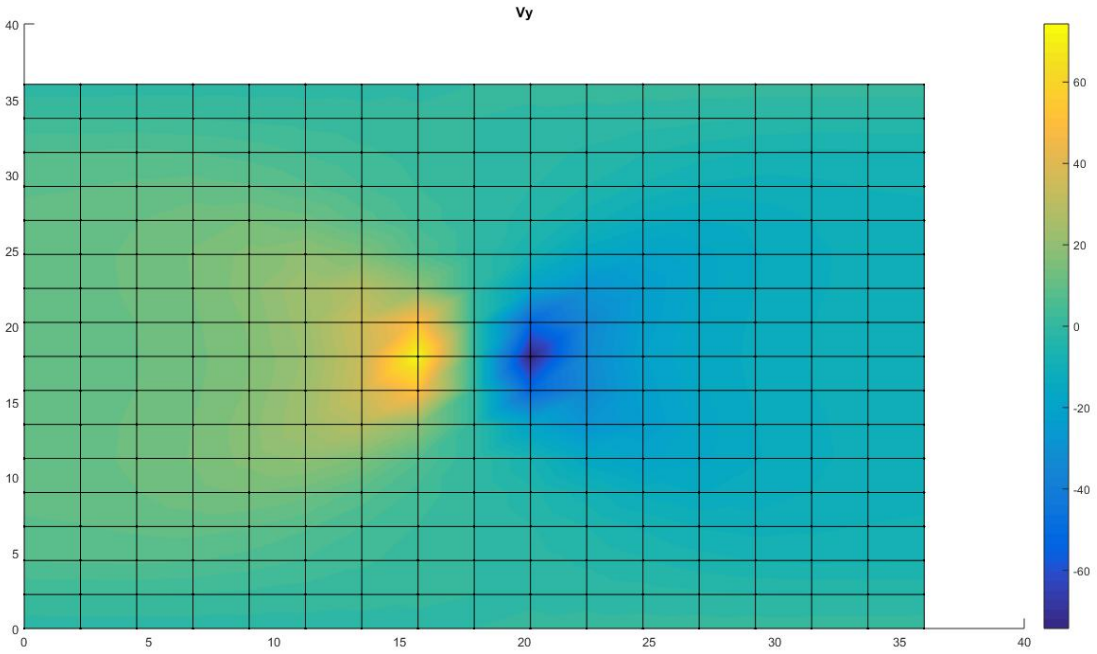


Figura 6.1-48. Esforço cortante Vy. Fonte: Elaborado pelo autor

Tabela 11 – esforços em uma placa duas bordas adjacentes livres e duas bordas engastadas

Tipo de esforço	Valor
Deslocamento máximo	-2.6589e-02 m
Momento fletor positivo máximo (Mx)	1.7642e+01 kN.m
Momento fletor negativo máximo (Mx)	9.4356e-02 kN.m
Momento fletor positivo máximo (My)	1.7642e+01 kN.m
Momento fletor negativo máximo (My)	9.4356e-02 kN.m
Momento torçor positivo máximo (Mxy)	5.1187e+00 kN.m
Momento torçor negativo máximo (Mxy)	-5.1187e+00 kN.m
Esforço cortante positivo máximo (Vx)	7.4277e+01 kN/m
Esforço cortante negativo máximo (Vx)	-7.4277e+01 kN/m
Esforço cortante positivo máximo (Vy)	7.4277e+01 kN/m
Esforço cortante negativo máximo (Vy)	-7.4277e+01 kN/m

Fonte: Elaborado pelo autor

6.2.5 Duas bordas opostas livres e duas bordas engastadas

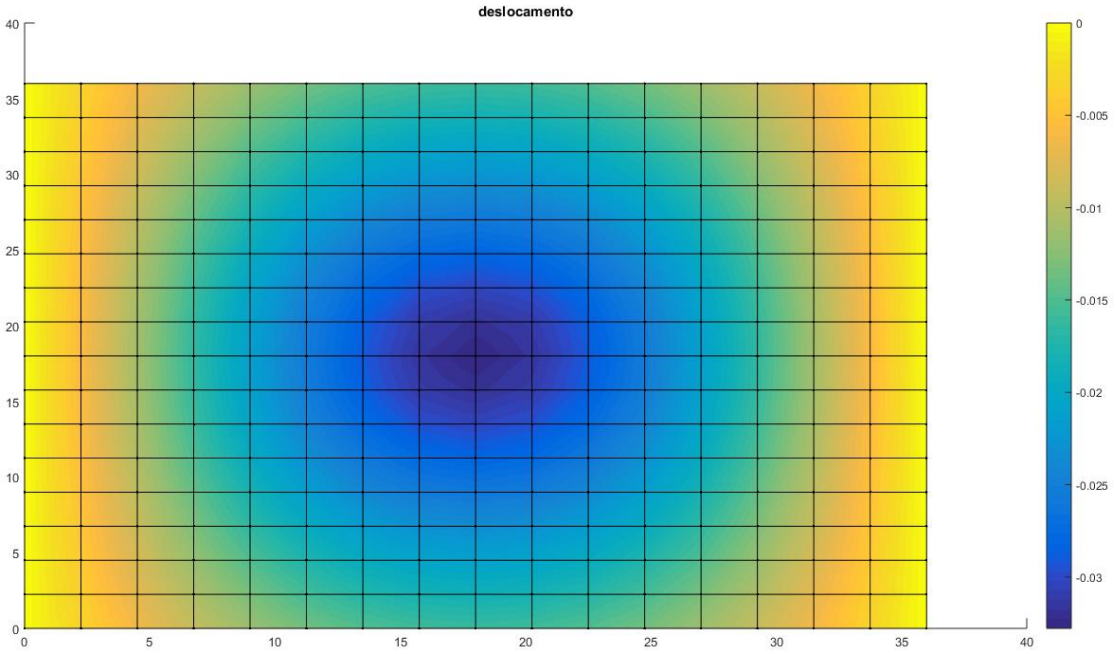


Figura 6.1-43. Deslocamento vertical. Fonte: Elaborado pelo autor

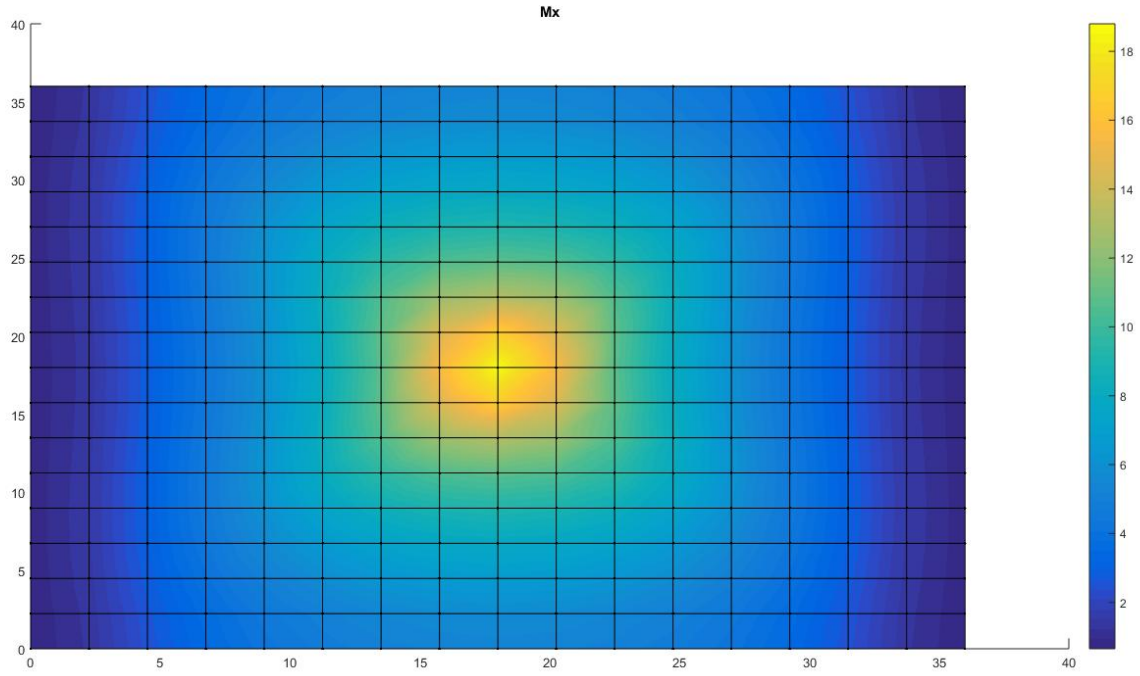


Figura 6.1-44. Momento fletor Mx. Fonte: Elaborado pelo autor

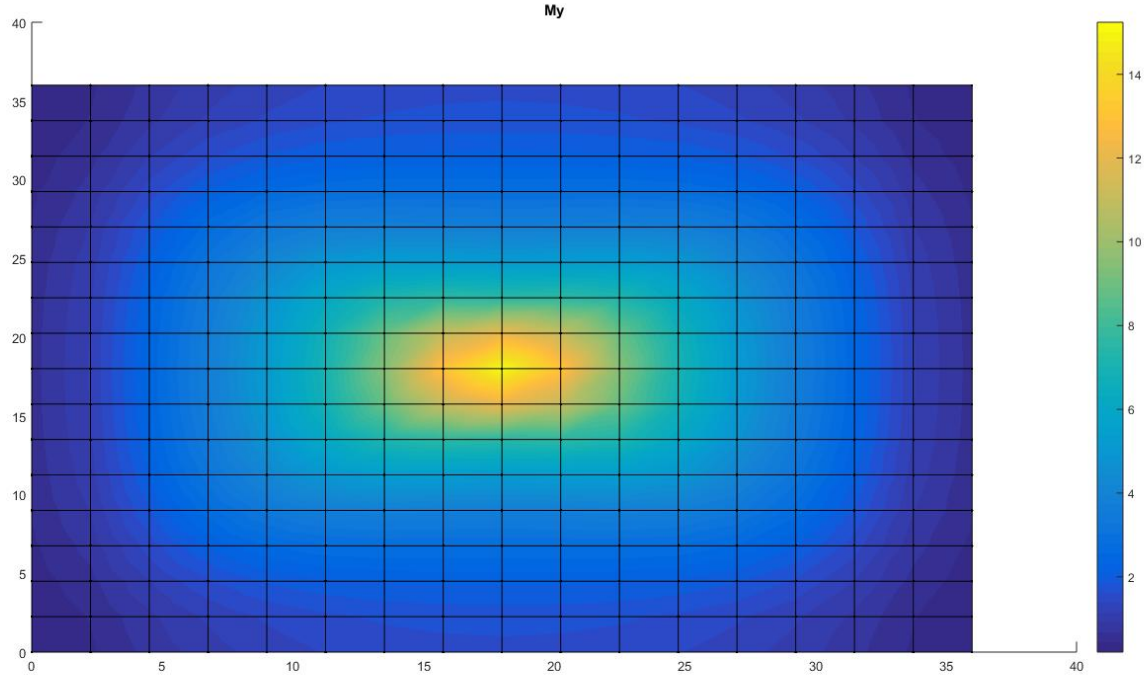


Figura 6.1-45. Momento fletor My. Fonte: Elaborado pelo autor

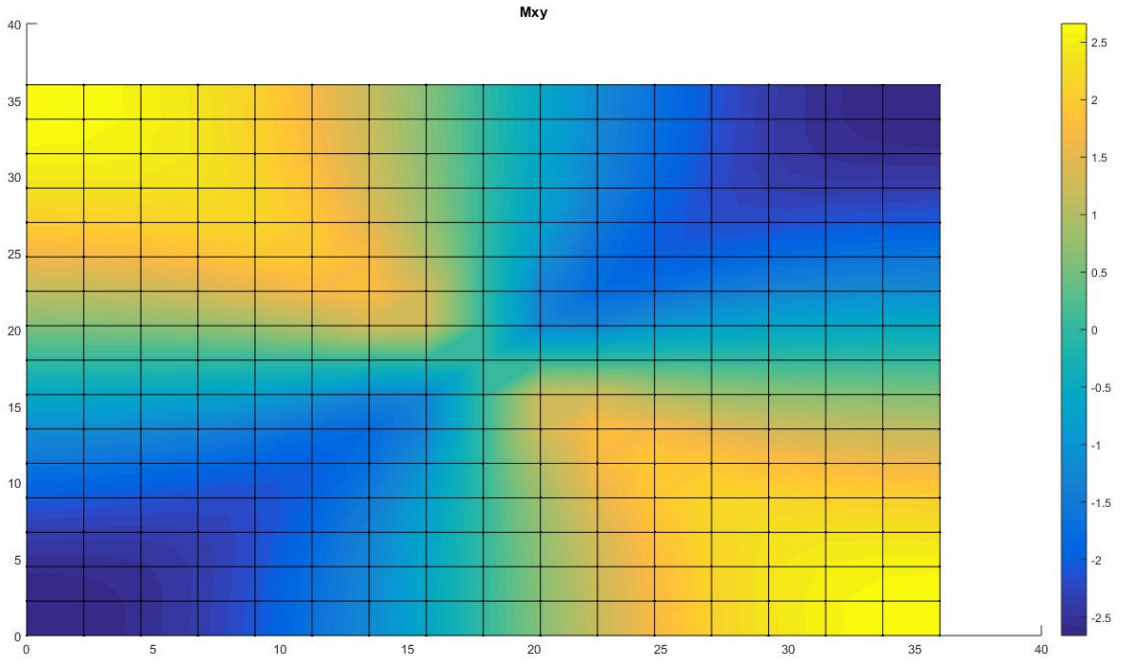


Figura 6.1-46. Momento torçor Mxy. Fonte: Elaborado pelo autor

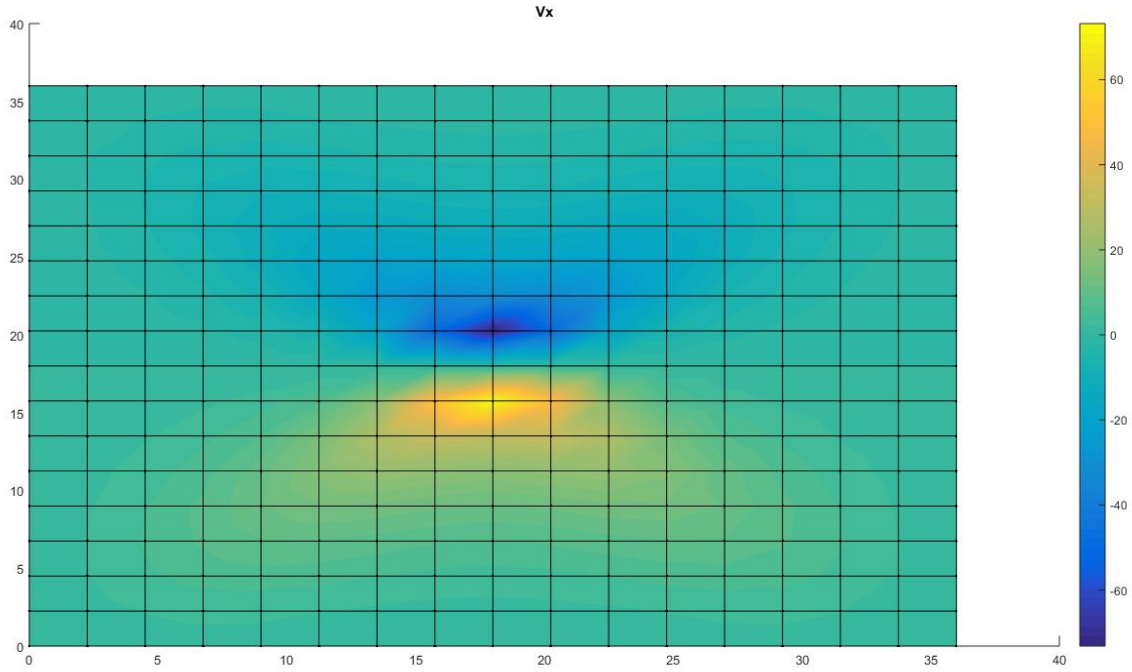


Figura 6.1-47. Esforço cortante Vx. Fonte: Elaborado pelo autor

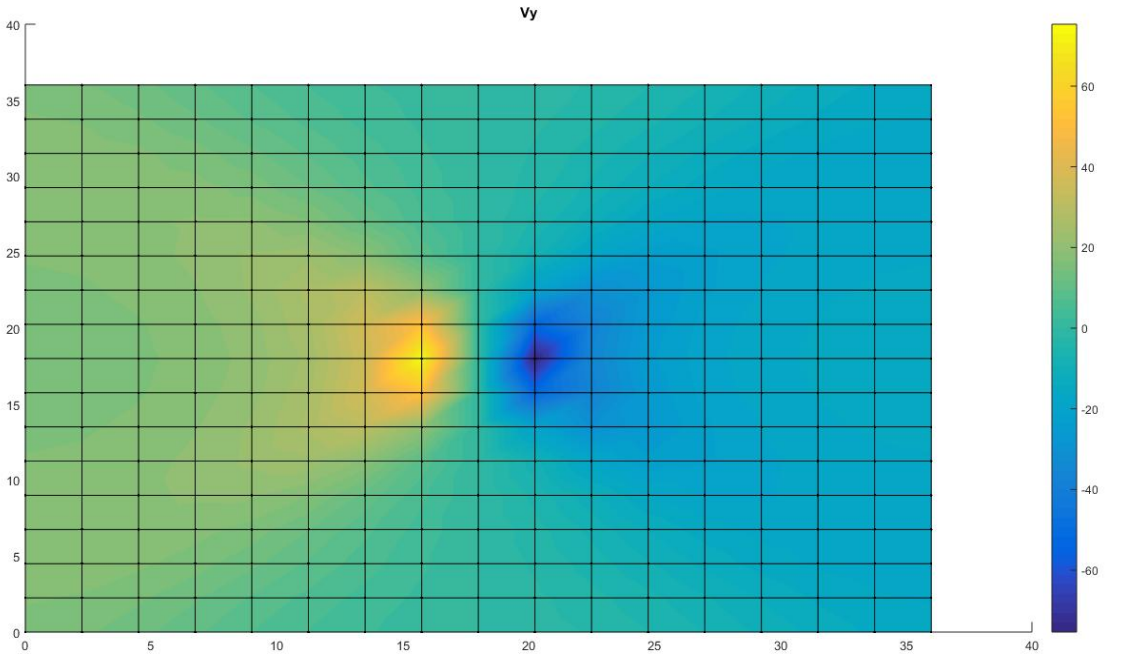


Figura 6.1-48. Esforço cortante Vy. Fonte: Elaborado pelo autor

Tabela 12 – esforços em uma placa duas bordas adjacentes livres e duas bordas engastadas

Tipo de esforço	Valor
Deslocamento máximo	-3.2784e-02 m
Momento fletor positivo máximo (Mx)	1.8800e+01 kN.m
Momento fletor negativo máximo (Mx)	6.5458e-01 kN.m
Momento fletor positivo máximo (My)	1.5243e+01 kN.m
Momento fletor negativo máximo (My)	2.0303e-01 kN.m
Momento torçor positivo máximo (Mxy)	2.6625e+00 kN.m
Momento torçor negativo máximo (Mxy)	-2.6625e+00 kN.m
Esforço cortante positivo máximo (Vx)	7.3178e+01 kN/m
Esforço cortante negativo máximo (Vx)	-7.3178e+01 kN/m
Esforço cortante positivo máximo (Vy)	7.5352e+01 kN/m
Esforço cortante negativo máximo (Vy)	-7.5352e+01 kN/m

Fonte: Elaborado pelo autor

6.2.6 Duas bordas adjacentes livres e duas bordas engastadas

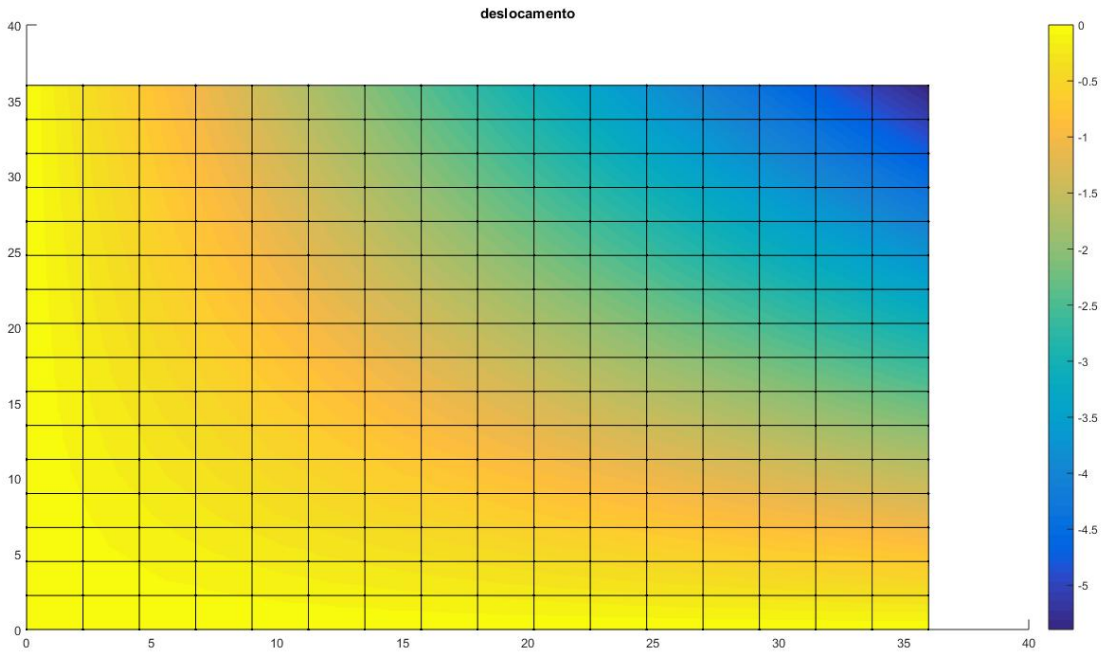


Figura 6.1-43. Deslocamento vertical. Fonte: Elaborado pelo autor

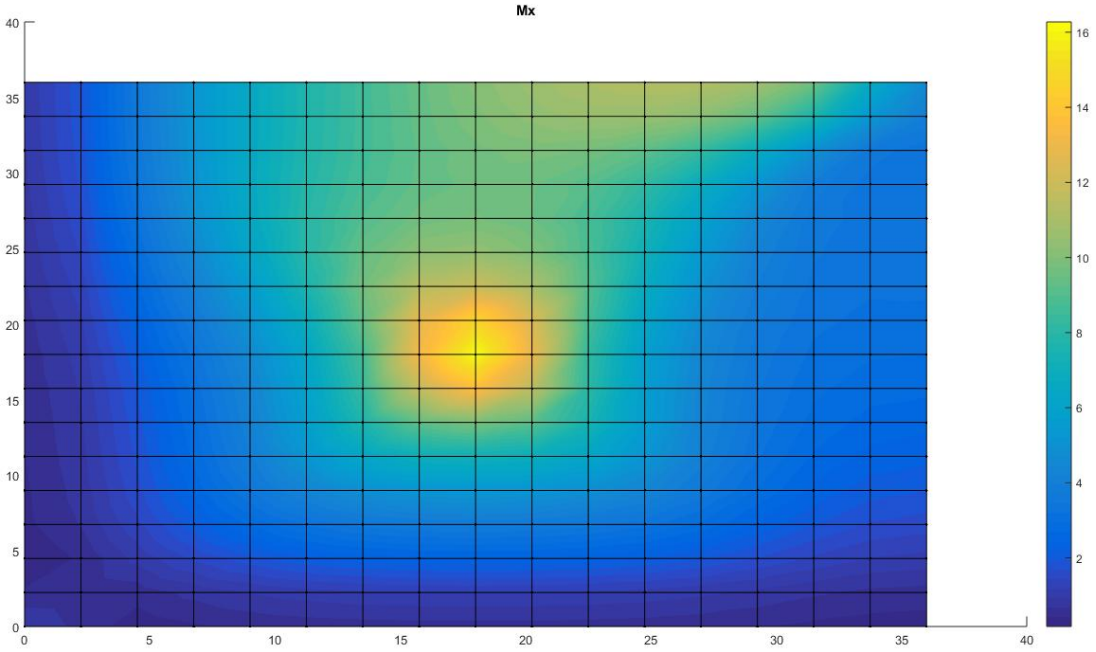


Figura 6.1-44. Momento fletor Mx. Fonte: Elaborado pelo autor

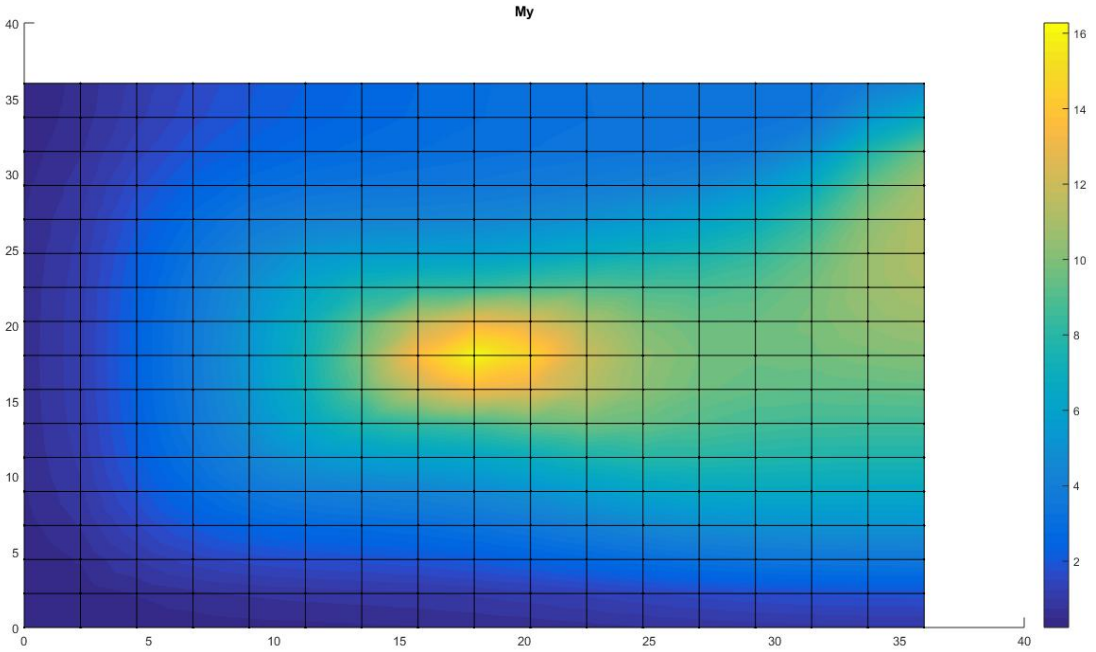


Figura 6.1-45. Momento fletor My. Fonte: Elaborado pelo autor

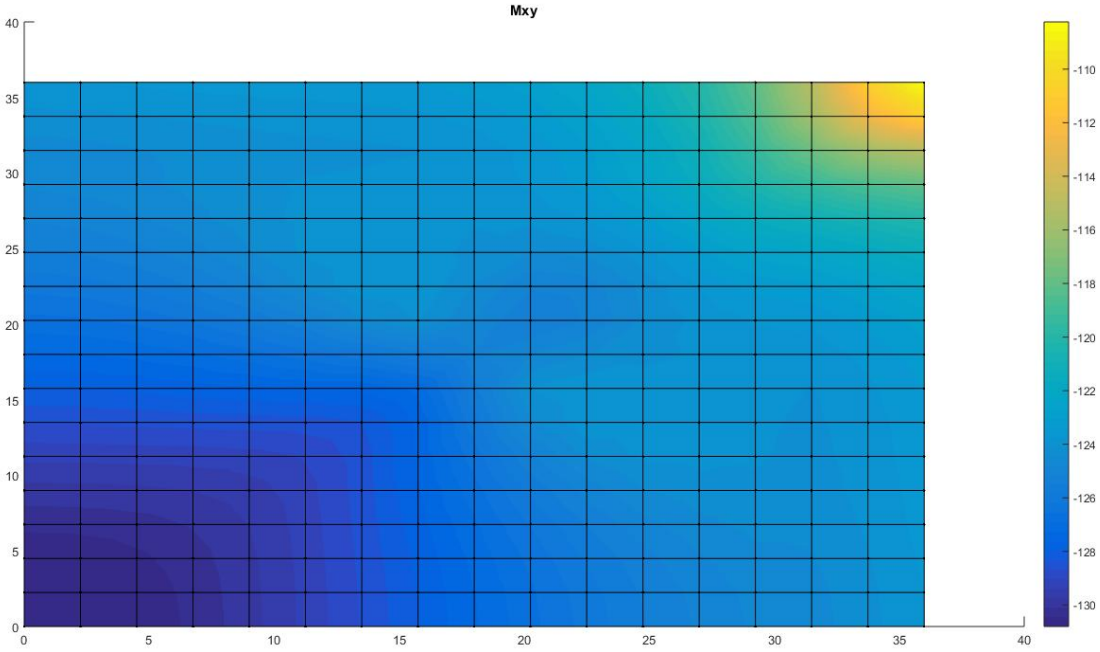


Figura 6.1-46. Momento torçor Mxy. Fonte: Elaborado pelo autor

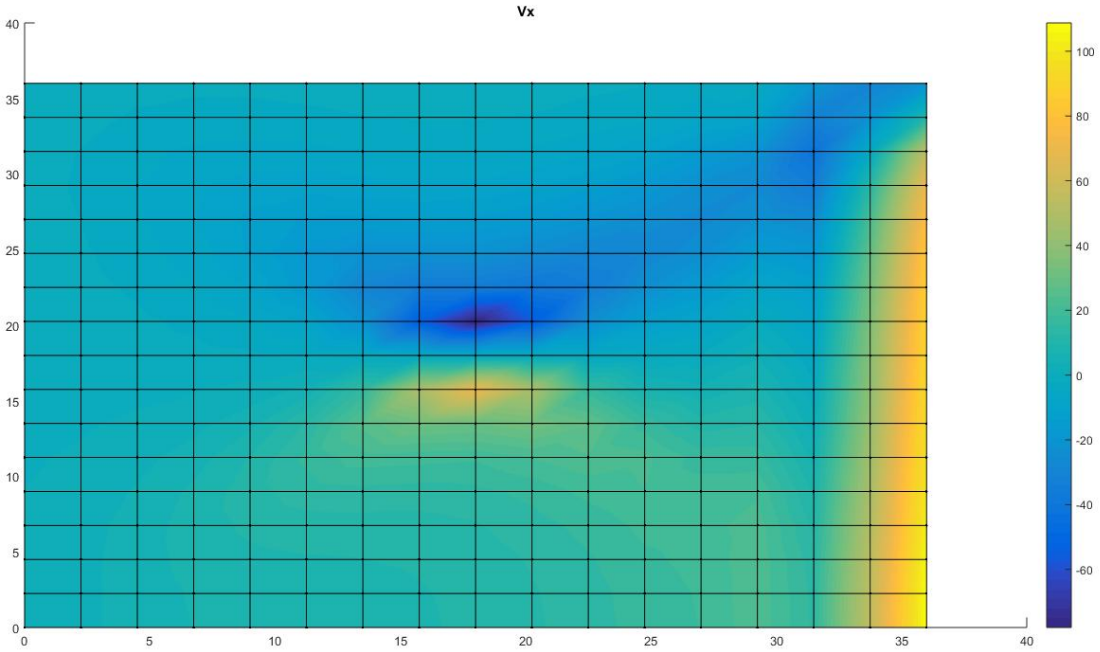


Figura 6.1-47. Esforço cortante Vx. Fonte: Elaborado pelo autor

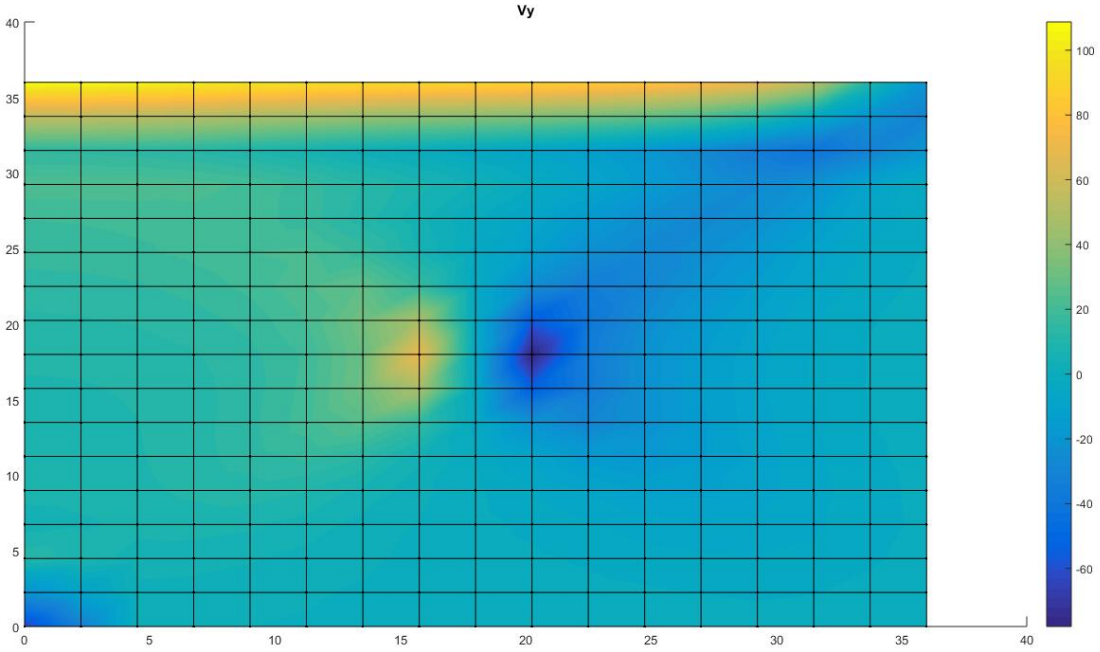


Figura 6.1-48. Esforço cortante Vy. Fonte: Elaborado pelo autor

Tabela 13 – esforços em uma placa duas bordas adjacentes livres e duas bordas engastadas

Tipo de esforço	Valor
Deslocamento máximo	-5.3942e+00 m
Momento fletor positivo máximo (Mx)	1.6274e+01 kN.m
Momento fletor negativo máximo (Mx)	1.7141e-01 kN.m
Momento fletor positivo máximo (My)	1.6274e+01 kN.m
Momento fletor negativo máximo (My)	2.4147e-01 kN.m
Momento torçor positivo máximo (Mxy)	-1.0824e+02 kN.m
Momento torçor negativo máximo (Mxy)	-1.3080e+02 kN.m
Esforço cortante positivo máximo (Vx)	1.0873e+02 kN/m
Esforço cortante negativo máximo (Vx)	-7.7929e+01 kN/m
Esforço cortante positivo máximo (Vy)	1.0873e+02 kN/m
Esforço cortante negativo máximo (Vy)	-7.7929e+01 kN/m

Fonte: Elaborado pelo autor

Conclusão

A teoria das placas de Kirchhoff requer continuidade C^1 para o campo de deslocamentos. Isso resulta numa maior complexidade para derivação de elementos de placa conformantes que satisfazem a condição de continuidade C^1.

Embora a conformidade não seja uma condição necessária para a convergência de um elemento, garante uma boa performance para geometrias arbitrárias, especialmente para elementos quadriláteros (Oñate, 2013).

A teoria das placas de Reissner-Mindlin é a base para derivação sistemática de elementos de placa com continuidade C^0 os quais incluem efeitos de cisalhamento transverso (Oñate, 2013). Praticamente, a única desvantagem dos elementos de placas de Reissner-Mindlin é a aparição do "shear locking" para situações de placas grossas. A integração reduzida dos termos de rigidez ao cisalhamento transverso é um procedimento simples e eficiente para a eliminação do "shear locking", embora possa produzir mecanismos que podem poluir a solução em alguns casos.

Bibliografia

Araújo, J. M. (2009). *Projeto Estrutural de Edifícios de Concreto Armado. Um Exemplo Completo.* Rio Grande: Dunas.

Araújo, J. M. (2010). *Curso de Concreto Armado* (3ª ed., Vol. 2). Rio Grande: Dunas.

Chaves, E. W., & Minguez, R. (2010). *Mecánica Computacional en la Ingeniería com Aplicaciones en MATLAB.* Ciudad Real: Gráficas Calina S.A.

Cook, R. D., Malkus, D. S., Plesha, M. E., & Witt, R. J. (2002). *Concepts and Applications of Finite Element Analysis* (4ª ed.). Madison, Winsconsin, Estados Unidos da América: John Wiley & Sons.

Khennanne, A. (2013). *Introduction to Finite Element Analysis Using MATLAB and Abaqus.* CRC Press Taylor & Francis Group.

Oñate, E. (2009). *Structural Analysis with the Finite Element Method Linear Statics* (Vol. 1). Barcelona, Espanha: Springer.

Oñate, E. (2013). *Structural Analysis with the Finite Element Method Linear Statics* (Vol. 2). Barcelona, Espanha: Springer.

Park, R., & Gamble, W. (2000). *Reinforced Concrete Slabs* (2ª ed.). New York, Estados Unidos da América.

Soriano, H. L. (2003). *Método de Elementos Finitos em análise de Estruturas.* São Paulo: Editora da Universidade de São Paulo.

Szilard, R. (2004). *Theories and Applications of Plate Analysis: Classical, Numeric and Engineering Methods.* New Jersey: Jonh Wiley & Sons, Inc.

Vaz, L. E. (2011). *Método dos Elementos Finitos em Análise de Estruturas.* Rio de Janeiro: Elsevier Ltda.

APÊNDICES

APÊNDICE A: CÓDIGO DA IMPLEMENTAÇÃO DA TEORIA DAS PLACAS DE KIRCHHOFF (EM MATLAB)

Arquivo T_exemplo1.m

```
clear all
clc
% Exemplo 1 (Tarik)
%Dados de entrada de uma placa retangular engastada nos quatro lados

format shortG
% Matriz coordenadas dos nós coord é a matriz conectividade dos
% elementos conec


comprimento=1;           % Comprimento da laje (paralelo a x)
largura=1;               % Largura da laje (paralelo a y)
NEX=20;                  % Número de elementos na direção x
NEY=20;                  % Número de elementos na direção y
nel=0;                   % Número de elementos da estrutura
dhx=comprimento/NEX;     % Comprimento de um elemento finito
dhy=largura/NEY;         % Largura de um elemento finito
Xorigem=-comprimento/2; % Coordenada global inicial x do desenho da
placa
Yorigem=-largura/2;      % Coordenada global inicial y do desenho da
placa
Zorigem=3;               % Coordenada global inicial z do desenho da
placa
nnoel=4;                 % Número de nós de cada elemento
nglno=3;                 % Número de graus de liberdade de cada nó do
elemento
nno=0;                   % Número de nós de toda a estrutura

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Numeração, coordenadas e conectividade dos nós dos elementos da laje
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%             y'
%             |                                  Esquema de numeração
%             |                                  dos nós do elemento.
%             |
%             |
%             n3 &&&&&&&&&&&&&&&&&&&& n4   ---
%             &.......................&    |
%             &.......................&    |
%             &.......................&    |
%             &.......................&    |
%             &.......................&    |
%             &.......................&   dhy
%             &.......................&    |
%             &.......................&    |
%             &.......................&    |
%             &.......................&    |
%             &.......................&    |
%             n1 &&&&&&&&&&&&&&&&&&&& n2 -------------------- x'
%
%             |---------- dhx ----------|
```

```
for i=1:NEX
    for j=1:NEY
        n1=j+(i-1)*(NEY+1);
        coord(n1,:)=[(i-1)*dhx+Xorigem (j-1)*dhy+Yorigem Zorigem];
        n2=j+i*(NEY+1);
        coord(n2,:)=[    i*dhx+Xorigem (j-1)*dhy+Yorigem Zorigem];
        n3=n1+1;
        coord(n3,:)=[(i-1)*dhx+Xorigem     j*dhy+Yorigem Zorigem];
        n4=n2+1;
        coord(n4,:)=[    i*dhx+Xorigem     j*dhy+Yorigem Zorigem];
        nel=nel+1;
        conec(nel,:)=[n1 n2 n4 n3];
        nno=n4;
    end
end


nglel=nno*nglno; % Número de graus de liberdade da estrutura


ex=zeros(nel,4); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ey=zeros(nel,4); % Coordenadas globais dos nós da estrutura
ez=zeros(nel,4); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:nel
    ex(i,:)=[coord(conec(i,1),1) coord(conec(i,2),1)
coord(conec(i,3),1) coord(conec(i,4),1)];
    ey(i,:)=[coord(conec(i,1),2) coord(conec(i,2),2)
coord(conec(i,3),2) coord(conec(i,4),2)];
    ez(i,:)=[coord(conec(i,1),3) coord(conec(i,2),3)
coord(conec(i,3),3) coord(conec(i,4),3)];
end

% gleNum identificação dos graus de liberdade de cada nó da estrutura
% Os graus de liberdade são, nessa ordem:
% vertical, rotação em volta de x e rotação em volta de y
gleNum=zeros(nno,nglno);
n=0;

for i=1:nno
    for j=1:nglno
        n=n+1;
        gleNum(i,j)=n;
    end
end

% Mtopologica Matriz topologica
Mtopologica=zeros(nel,nnoel*nglno+1);

for i=1:nel
   [st]=submatriztopologica(i,nnoel,nglno,conec,gleNum);
   Mtopologica(i,:)=[i st];
   SMtopologica(i,:)=[st];
end

gdlest=max(max(SMtopologica));

K=sparse(gdlest,gdlest);
f=zeros(gdlest,1);
%f(4)=100;
```

```
% Matrix Propriedades do elemento
t=0.01;
E=10.92e6;
v=0.3;

% Propriedades do elementos pe
pe=zeros(nel,3);
for i=1:nel
    pe(i,:)=[t E v];
end

% carga nodal equivalente para carga uniformemente distribuida
eq=zeros(nel,1);
%
for i=1:nel
    eq(i,:)=-1;
end

% rigidez e força de cada elemento e matriz de rigidez global

for i=1:nel

[Ke,fe]=matrizrigidezelementodeplacaACM(ex(i,:),ey(i,:),pe(i,:),eq(i,:
));
    [K,f]=montagemrigidezglobal(Mtopologica(i,:),K,Ke,f,fe);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Condições de contorno
% Os quatro lados engastados
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% gleImp graus de liberdade de cada nó que possui algum impedimento
gleImp=false(nno,nglno);
% gleNum deslocamento prescrito dos graus de liberdade impedidos de
cada nó
gleDPE=zeros(nno,nglno);

m=1;
for i=1:nno
    if coord(i,1) == Xorigem;  % Lado paralelo ao eixo y
        gleImp(i,:)=[1 1 1];   % 1 = grau de liberdade preso, 0 = grau
de liberdade livre
        gleDPE(i,:)=[0 0 0];   % Valor do deslocamento prescrito
    elseif coord(i,1) == Xorigem+comprimento;  % Lado paralelo ao eixo
y
        gleImp(i,:)=[1 1 1];   % 1 = grau de liberdade preso, 0 = grau
de liberdade livre
        gleDPE(i,:)=[0 0 0];   % Valor do deslocamento prescrito
    elseif coord(i,2) == Yorigem;  % Lado paralelo ao eixo x
        gleImp(i,:)=[1 1 1];   % 1 = grau de liberdade preso, 0 = grau
de liberdade livre
        gleDPE(i,:)=[0 0 0];   % Valor do deslocamento prescrito
    elseif coord(i,2) == Yorigem+largura;  % Lado paralelo ao eixo x
        gleImp(i,:)=[1 1 1];   % 1 = grau de liberdade preso, 0 = grau
de liberdade livre
        gleDPE(i,:)=[0 0 0];   % Valor do deslocamento prescrito
    end

    % Identificação dos graus de liberdade impedidos e
```

```
    % definição dos seus deslocamentos prescritos
    for j=1:nglno
        if(gleImp(i,j))
            cc(m,:)=[gleNum(i,j) gleDPE(i,j)];
            m=m+1;
        end
    end
end

% Resolve o sistema de equações e encontra os deslocamentos
% e as reações de apoio
[d,R]=resolvesistema(K,f,cc);
R1=max(R)
R2=min(R)

% Extrai os deslocamentos
[ed]=extraideslocamento(Mtopologica,d);

%% Cálculo dos esforços e plotagem das reações internas e
%% deslocamentos
edv=ed(:,1:3:12);
ESF_Mom=zeros(nel,12);
ESF_Cis=zeros(nel, 8);
ReApoio=zeros(nel, 8);

ESFMx =zeros(nel,4);
ESFMy =zeros(nel,4);
ESFMxy=zeros(nel,4);

ESFVx =zeros(nel,4);
ESFVy =zeros(nel,4);

ReaRx =zeros(nel,4);
ReaRy =zeros(nel,4);

deslVert=d(1:3:nglel,:);

format short e

dez=zeros(nel,4);

dmx=zeros(nel,4);
dmy=zeros(nel,4);
dmxy=zeros(nel,4);

dvx=zeros(nel,4);
dvy=zeros(nel,4);

drx=zeros(nel,4);
dry=zeros(nel,4);

for i=1:nel

[ESF_Cis(i,:),ESF_Mom(i,:),ReApoio(i,:)]=esforconaplaca(ex(i,:),ey(i,:
),pe(i,:),ed(i,:)); % Esforços na placa

    ESFMx (i,:)=ESF_Mom(i,1:3:12);
    ESFMy (i,:)=ESF_Mom(i,2:3:12);
    ESFMxy(i,:)=ESF_Mom(i,3:3:12);
```

```
    ESFVx (i,:)=ESF_Cis(i,1:2:8);
    ESFVy (i,:)=ESF_Cis(i,2:2:8);

    ReaRx (i,:)=ReApoio(i,1:2:8);
    ReaRy (i,:)=ReApoio(i,2:2:8);

    dez (i,:)=[deslVert(conec(i,1),1)...
               deslVert(conec(i,2),1)...
               deslVert(conec(i,3),1)...
               deslVert(conec(i,4),1)];

    dmx (i,:)=[ESFMx(i,1)...
               ESFMx(i,2)...
               ESFMx(i,3)...
               ESFMx(i,4)];
    dmy (i,:)=[ESFMy(i,1)...
               ESFMy(i,2)...
               ESFMy(i,3)...
               ESFMy(i,4)];
    dmxy(i,:)=[ESFMxy(i,1)...
               ESFMxy(i,2)...
               ESFMxy(i,3)...
               ESFMxy(i,4)];

    dvx (i,:)=[ESFVx(i,1)...
               ESFVx(i,2)...
               ESFVx(i,3)...
               ESFVx(i,4)];
    dvy (i,:)=[ESFVy(i,1)...
               ESFVy(i,2)...
               ESFVy(i,3)...
               ESFVy(i,4)];

    drx (i,:)=[ReaRx(i,1)...
               ReaRx(i,2)...
               ReaRx(i,3)...
               ReaRx(i,4)];
    dry (i,:)=[ReaRy(i,1)...
               ReaRy(i,2)...
               ReaRy(i,3)...
               ReaRy(i,4)];

end

deslMax=max(abs(deslVert))

MxMax =max(max(abs(ESFMx )));
MyMax =max(max(abs(ESFMy )));
MxyMax=max(max(abs(ESFMxy)));

MomMax=max(max(abs(ESF_Mom)));
CisMax=max(max(abs(ESF_Cis)));
ReaMax=max(max(abs(ReApoio)));

MxMax=max(max(ESFMx))
MyMax=max(max(ESFMy))

MxyMax=max(max(ESFMxy))
MxyMin=min(min(ESFMxy))
```

```
MxMin=min(min(ESFMx))
MyMin=min(min(ESFMy))

VxMax=max(max(ReaRx))
VyMax=max(max(ReaRy))

VxMin=min(min(ReaRx))
VyMin=min(min(ReaRy))

% Fator de deslocamento gráfico máximo
FatorDes=0.5*(comprimento+largura)/2;

dez =FatorDes*dez /deslMax+Zorigem;

dmx =FatorDes*dmx /MomMax +Zorigem;
dmy =FatorDes*dmy /MomMax +Zorigem;
dmxy=FatorDes*dmxy/MomMax +Zorigem;

dvx =FatorDes*dvx /CisMax +Zorigem;
dvy =FatorDes*dvy /CisMax +Zorigem;

drx =FatorDes*drx /ReaMax +Zorigem;
dry =FatorDes*dry /ReaMax +Zorigem;

x=ex';
y=ey';
z=ez';

figure(1)
title('deslocamento')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dez',edv','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(2)
title('Mx')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmx',ESFMx','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(3)
title('My')
grid on
hold on
```

```
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmy',ESFMy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(4)
title('Mxy')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmxy',ESFMxy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(5)
title('Vx')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,drx',ReaRx','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(6)
title('Vy')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dry',ReaRy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off
```

Arquivo T_exemplo2.m

```
clear all
clc
% Exemplo 2 (Tarik)
%Dados de entrada de uma placa retangular apoiada nos quatro lados

format shortG
% Matriz coordenadas dos nós Coord e a matriz conectividade dos
% elementos conec


comprimento=1;          % Comprimento da laje (paralelo a x)
largura=1;              % Largura da laje (paralelo a y)
NEX=20;                 % Número de elementos na direção x
NEY=20;                 % Número de elementos na direção y
nel=0;                  % Número de elementos da estrutura
dhx=comprimento/NEX;    % Comprimento de um elemento finito
dhy=largura/NEY;        % Largura de um elemento finito
Xorigem=-comprimento/2; % Coordenada global inicial x do desenho da
placa
Yorigem=-largura/2;     % Coordenada global inicial y do desenho da
placa
Zorigem=3;              % Coordenada global inicial z do desenho da
placa
nnoel=4;                % Número de nós de cada elemento
nglno=3;                % Número de graus de liberdade de cada nó do
elemento
nno=0;                  % Número de nós de toda a estrutura

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%
%  Numeração, coordenadas e conectividade dos nós dos elementos da
laje
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

%             y'
%             |                                   Esquema de numeração
%             |                                   dos nós do elemento.
%             |
%             |
%             n3 &&&&&&&&&&&&&&&&&&&&&& n4   ---
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             &.........................&   dhy
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             n1 &&&&&&&&&&&&&&&&&&&&&& n2 ------------------- x'
%
%             |---------- dhx ----------|

for i=1:NEX
```

```
    for j=1:NEY
        n1=j+(i-1)*(NEY+1);
        coord(n1,:)=[(i-1)*dhx+Xorigem (j-1)*dhy+Yorigem Zorigem];
        n2=j+i*(NEY+1);
        coord(n2,:)=[    i*dhx+Xorigem (j-1)*dhy+Yorigem Zorigem];
        n3=n1+1;
        coord(n3,:)=[(i-1)*dhx+Xorigem     j*dhy+Yorigem Zorigem];
        n4=n2+1;
        coord(n4,:)=[    i*dhx+Xorigem     j*dhy+Yorigem Zorigem];
        nel=nel+1;
        conec(nel,:)=[n1 n2 n4 n3];
        nno=n4;
    end
end


nglel=nno*nglno; % Número de graus de liberdade da estrutura


ex=zeros(nel,4); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ey=zeros(nel,4); % Coordenadas globais dos nós da estrutura
ez=zeros(nel,4); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:nel
    ex(i,:)=[coord(conec(i,1),1) coord(conec(i,2),1)
coord(conec(i,3),1) coord(conec(i,4),1)];
    ey(i,:)=[coord(conec(i,1),2) coord(conec(i,2),2)
coord(conec(i,3),2) coord(conec(i,4),2)];
    ez(i,:)=[coord(conec(i,1),3) coord(conec(i,2),3)
coord(conec(i,3),3) coord(conec(i,4),3)];
end

% gleNum identificação dos graus de liberdade de cada nó da estrutura
% Os graus de liberdade são, nessa ordem:
% vertical, rotação em volta de x e rotação em volta de y
gleNum=zeros(nno,nglno);
n=0;

for i=1:nno
    for j=1:nglno
        n=n+1;
        gleNum(i,j)=n;
    end
end

% Mtopologica Matriz topologica
Mtopologica=zeros(nel,nnoel*nglno+1);

for i=1:nel
   [st]=submatriztopologica(i,nnoel,nglno,conec,gleNum);
   Mtopologica(i,:)=[i st];
   SMtopologica(i,:)=[st];
end

gdlest=max(max(SMtopologica));

K=sparse(gdlest,gdlest);
f=zeros(gdlest,1);
%f(4)=100;
```

```
% Matrix Propriedades do elemento
t=0.01;
E=10.92e6;
v=0.3;

% Propriedades do elementos pe
pe=zeros(nel,3);
for i=1:nel
    pe(i,:)=[t E v];
end

% carga nodal equivalente para carga uniformemente distribuida
eq=zeros(nel,1);
%
for i=1:nel
    eq(i,:)=-1;
end

% rigidez e força de cada elemento e matriz de rigidez global

for i=1:nel

[Ke,fe]=matrizrigidezelementodeplacaACM(ex(i,:),ey(i,:),pe(i,:),eq(i,:
));
    [K,f]=montagemrigidezglobal(Mtopologica(i,:),K,Ke,f,fe);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Condições de contorno
% Os quatro lados apoiados
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% gleImp graus de liberdade de cada nó que possui algum impedimento
gleImp=false(nno,nglno);
% gleNum deslocamento prescrito dos graus de liberdade impedidos de
cada nó
gleDPE=zeros(nno,nglno);

m=1;
for i=1:nno
    if coord(i,1) == Xorigem;  % Lado paralelo ao eixo y
        gleImp(i,:)=[1 0 0];   % 1 = grau de liberdade preso, 0 = grau
de liberdade livre
        gleDPE(i,:)=[0 0 0];   % Valor do deslocamento prescrito
    elseif coord(i,1) == Xorigem+comprimento;  % Lado paralelo ao eixo
y
        gleImp(i,:)=[1 0 0];   % 1 = grau de liberdade preso, 0 = grau
de liberdade livre
        gleDPE(i,:)=[0 0 0];   % Valor do deslocamento prescrito
    elseif coord(i,2) == Yorigem;  % Lado paralelo ao eixo x
        gleImp(i,:)=[1 0 0];   % 1 = grau de liberdade preso, 0 = grau
de liberdade livre
        gleDPE(i,:)=[0 0 0];   % Valor do deslocamento prescrito
    elseif coord(i,2) == Yorigem+largura;  % Lado paralelo ao eixo x
        gleImp(i,:)=[1 0 0];   % 1 = grau de liberdade preso, 0 = grau
de liberdade livre
        gleDPE(i,:)=[0 0 0];   % Valor do deslocamento prescrito
    end

    % Identificação dos graus de liberdade impedidos e
    % definição dos seus deslocamentos prescritos
```

```
    for j=1:nglno
        if(gleImp(i,j))
            cc(m,:)=[gleNum(i,j) gleDPE(i,j)];
            m=m+1;
        end
    end
end

% Resolve o sistema de equações e encontra os deslocamentos
% e as reações de apoio
[d,R]=resolvesistema(K,f,cc);
R1=max(R)
R2=min(R)
%Reacoes=[R(1);R(2);R(3);R(10);R(11);R(12)]

% Extrai os deslocamentos
[ed]=extraideslocamento(Mtopologica,d);

%% Cálculo do esforços e plotagem das reações internas e deslocamentos
edv=ed(:,1:3:12);
ESF_Mom=zeros(nel,12);
ESF_Cis=zeros(nel, 8);
ReApoio=zeros(nel, 8);

ESFMx =zeros(nel,4);
ESFMy =zeros(nel,4);
ESFMxy=zeros(nel,4);

ESFVx =zeros(nel,4);
ESFVy =zeros(nel,4);

ReaRx =zeros(nel,4);
ReaRy =zeros(nel,4);

deslVert=d(1:3:nglel,:);

format short e

dez=zeros(nel,4);

dmx=zeros(nel,4);
dmy=zeros(nel,4);
dmxy=zeros(nel,4);

dvx=zeros(nel,4);
dvy=zeros(nel,4);

drx=zeros(nel,4);
dry=zeros(nel,4);

for i=1:nel

[ESF_Cis(i,:),ESF_Mom(i,:),ReApoio(i,:)]=esforconaplaca(ex(i,:),ey(i,:
),pe(i,:),ed(i,:)); % Esforços na placa

    ESFMx (i,:)=ESF_Mom(i,1:3:12);
    ESFMy (i,:)=ESF_Mom(i,2:3:12);
    ESFMxy(i,:)=ESF_Mom(i,3:3:12);

    ESFVx (i,:)=ESF_Cis(i,1:2:8);
```

```
    ESFVy (i,:)=ESF_Cis(i,2:2:8);

    ReaRx (i,:)=ReApoio(i,1:2:8);
    ReaRy (i,:)=ReApoio(i,2:2:8);

    dez (i,:)=[deslVert(conec(i,1),1)...
               deslVert(conec(i,2),1)...
               deslVert(conec(i,3),1)...
               deslVert(conec(i,4),1)];

    dmx (i,:)=[ESFMx(i,1)...
               ESFMx(i,2)...
               ESFMx(i,3)...
               ESFMx(i,4)];
    dmy (i,:)=[ESFMy(i,1)...
               ESFMy(i,2)...
               ESFMy(i,3)...
               ESFMy(i,4)];
    dmxy(i,:)=[ESFMxy(i,1)...
               ESFMxy(i,2)...
               ESFMxy(i,3)...
               ESFMxy(i,4)];

    dvx (i,:)=[ESFVx(i,1)...
               ESFVx(i,2)...
               ESFVx(i,3)...
               ESFVx(i,4)];
    dvy (i,:)=[ESFVy(i,1)...
               ESFVy(i,2)...
               ESFVy(i,3)...
               ESFVy(i,4)];

    drx (i,:)=[ReaRx(i,1)...
               ReaRx(i,2)...
               ReaRx(i,3)...
               ReaRx(i,4)];
    dry (i,:)=[ReaRy(i,1)...
               ReaRy(i,2)...
               ReaRy(i,3)...
               ReaRy(i,4)];

end

deslMax=max(abs(deslVert))

MxMax =max(max(abs(ESFMx )));
MyMax =max(max(abs(ESFMy )));
MxyMax=max(max(abs(ESFMxy)));

MomMax=max(max(abs(ESF_Mom)));
CisMax=max(max(abs(ESF_Cis)));
ReaMax=max(max(abs(ReApoio)));

MxMax=max(max(ESFMx))
MyMax=max(max(ESFMy))

MxyMax=max(max(ESFMxy))
MxyMin=min(min(ESFMxy))

MxMin=min(min(ESFMx))
```

```
MyMin=min(min(ESFMy))

VxMax=max(max(ReaRx))
VyMax=max(max(ReaRy))

VxMin=min(min(ReaRx))
VyMin=min(min(ReaRy))

% Fator de deslocamento gráfico máximo
FatorDes=0.5*(comprimento+largura)/2;

dez =FatorDes*dez /deslMax+Zorigem;

dmx =FatorDes*dmx /MomMax +Zorigem;
dmy =FatorDes*dmy /MomMax +Zorigem;
dmxy=FatorDes*dmxy/MomMax +Zorigem;

dvx =FatorDes*dvx /CisMax +Zorigem;
dvy =FatorDes*dvy /CisMax +Zorigem;

drx =FatorDes*drx /ReaMax +Zorigem;
dry =FatorDes*dry /ReaMax +Zorigem;

x=ex';
y=ey';
z=ez';

figure(1)
title('deslocamento')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dez',edv','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(2)
title('Mx')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmx',ESFMx','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(3)
title('My')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
```

```
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmy',ESFMy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(4)
title('Mxy')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmxy',ESFMxy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(5)
title('Vx')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,drx',ReaRx','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(6)
title('Vy')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dry',ReaRy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off
```

Arquivo T_exemplo3.m

```
clear all
clc
% Exemplo 2 (Tarik)
% Dados de entrada de uma placa retangular engastada em dois lados
opostos
% e duas bordas livres

format shortG
% Matriz coordenadas dos nós Coord e a matriz conectividade dos
% elementos conec


comprimento=1;          % Comprimento da laje (paralelo a x)
largura=1;              % Largura da laje (paralelo a y)
NEX=20;                 % Número de elementos na direção x
NEY=20;                 % Número de elementos na direção y
nel=0;                  % Número de elementos da estrutura
dhx=comprimento/NEX;    % Comprimento de um elemento finito
dhy=largura/NEY;        % Largura de um elemento finito
Xorigem=-comprimento/2; % Coordenada global inicial x do desenho da
placa
Yorigem=-largura/2;     % Coordenada global inicial y do desenho da
placa
Zorigem=3;              % Coordenada global inicial z do desenho da
placa
nnoel=4;                % Número de nós de cada elemento
nglno=3;                % Número de graus de liberdade de cada nó do
elemento
nno=0;                  % Número de nós de toda a estrutura

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%
%  Numeração, coordenadas e conectividade dos nós dos elementos da
laje
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

%             y'
%             |                                  Esquema de numeração
%             |                                  dos nós do elemento.
%             |
%             |
%             n3 &&&&&&&&&&&&&&&&&&&& n4   ---
%             &........................&    |
%             &........................&    |
%             &........................&    |
%             &........................&    |
%             &........................&    |
%             &........................&   dhy
%             &........................&    |
%             &........................&    |
%             &........................&    |
%             &........................&    |
%             &........................&    |
%             n1 &&&&&&&&&&&&&&&&&&&& n2 ------------------- x'
%
%             |---------- dhx ----------|
```

```
for i=1:NEX
    for j=1:NEY
        n1=j+(i-1)*(NEY+1);
        coord(n1,:)=[(i-1)*dhx+Xorigem (j-1)*dhy+Yorigem Zorigem];
        n2=j+i*(NEY+1);
        coord(n2,:)=[    i*dhx+Xorigem (j-1)*dhy+Yorigem Zorigem];
        n3=n1+1;
        coord(n3,:)=[(i-1)*dhx+Xorigem     j*dhy+Yorigem Zorigem];
        n4=n2+1;
        coord(n4,:)=[    i*dhx+Xorigem     j*dhy+Yorigem Zorigem];
        nel=nel+1;
        conec(nel,:)=[n1 n2 n4 n3];
        nno=n4;
    end
end


nglel=nno*nglno; % Número de graus de liberdade da estrutura


ex=zeros(nel,4); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ey=zeros(nel,4); % Coordenadas globais dos nós da estrutura
ez=zeros(nel,4); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:nel
    ex(i,:)=[coord(conec(i,1),1) coord(conec(i,2),1)
coord(conec(i,3),1) coord(conec(i,4),1)];
    ey(i,:)=[coord(conec(i,1),2) coord(conec(i,2),2)
coord(conec(i,3),2) coord(conec(i,4),2)];
    ez(i,:)=[coord(conec(i,1),3) coord(conec(i,2),3)
coord(conec(i,3),3) coord(conec(i,4),3)];
end

% gleNum identificação dos graus de liberdade de cada nó da estrutura
% Os graus de liberdade são, nessa ordem:
% vertical, rotação em volta de x e rotação em volta de y
gleNum=zeros(nno,nglno);
n=0;

for i=1:nno
    for j=1:nglno
        n=n+1;
        gleNum(i,j)=n;
    end
end

% Mtopologica Matriz topologica
Mtopologica=zeros(nel,nnoel*nglno+1);

for i=1:nel
   [st]=submatriztopologica(i,nnoel,nglno,conec,gleNum);
   Mtopologica(i,:)=[i st];
   SMtopologica(i,:)=[st];
end

gdlest=max(max(SMtopologica));

K=sparse(gdlest,gdlest);
f=zeros(gdlest,1);
```

```
%f(4)=100;

% Matrix Propriedades do elemento
t=0.01;
E=10.92e6;
v=0.3;

% Propriedades do elementos pe
pe=zeros(nel,3);
for i=1:nel
    pe(i,:)=[t E v];
end

% carga nodal equivalente para carga uniformemente distribuida
eq=zeros(nel,1);
%
for i=1:nel
    eq(i,:)=-1;
end

% rigidez e força de cada elemento e matriz de rigidez global

for i=1:nel

[Ke,fe]=matrizrigidezelementodeplacaACM(ex(i,:),ey(i,:),pe(i,:),eq(i,:
));
    [K,f]=montagemrigidezglobal(Mtopologica(i,:),K,Ke,f,fe);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Condições de contorno
% Dois lados opostos engastados
% e duas bordas livres
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% gleImp graus de liberdade de cada nó que possui algum impedimento
gleImp=false(nno,nglno);
% gleNum deslocamento prescrito dos graus de liberdade impedidos de
cada nó
gleDPE=zeros(nno,nglno);

m=1;
for i=1:nno
    if coord(i,1) == Xorigem;  % Lado paralelo ao eixo y
        gleImp(i,:)=[1 1 1];   % 1 = grau de liberdade preso, 0 = grau
de liberdade livre
        gleDPE(i,:)=[0 0 0];   % Valor do deslocamento prescrito
    elseif coord(i,1) == Xorigem+comprimento;  % Lado paralelo ao eixo
y
        gleImp(i,:)=[1 1 1];   % 1 = grau de liberdade preso, 0 = grau
de liberdade livre
        gleDPE(i,:)=[0 0 0];   % Valor do deslocamento prescrito
    end

    % Identificação dos graus de liberdade impedidos e
    % definição dos seus deslocamentos prescritos
    for j=1:nglno
        if(gleImp(i,j))
            cc(m,:)=[gleNum(i,j) gleDPE(i,j)];
            m=m+1;
        end
```

```
    end
end

% Resolve o sistema de equações e encontra os deslocamentos
% e as reações de apoio
[d,R]=resolvesistema(K,f,cc);
R1=max(R)
R2=min(R)

% Extrai os deslocamentos
[ed]=extraideslocamento(Mtopologica,d);

%% Cálculo do esforços e plotagem das reações internas e deslocamentos
edv=ed(:,1:3:12);
ESF_Mom=zeros(nel,12);
ESF_Cis=zeros(nel, 8);
ReApoio=zeros(nel, 8);

ESFMx =zeros(nel,4);
ESFMy =zeros(nel,4);
ESFMxy=zeros(nel,4);

ESFVx =zeros(nel,4);
ESFVy =zeros(nel,4);

ReaRx =zeros(nel,4);
ReaRy =zeros(nel,4);

deslVert=d(1:3:nglel,:);

format short e

dez=zeros(nel,4);

dmx=zeros(nel,4);
dmy=zeros(nel,4);
dmxy=zeros(nel,4);

dvx=zeros(nel,4);
dvy=zeros(nel,4);

drx=zeros(nel,4);
dry=zeros(nel,4);

for i=1:nel

[ESF_Cis(i,:),ESF_Mom(i,:),ReApoio(i,:)]=esforconaplaca(ex(i,:),ey(i,:
),pe(i,:),ed(i,:)); % Esforços na placa

    ESFMx (i,:)=ESF_Mom(i,1:3:12);
    ESFMy (i,:)=ESF_Mom(i,2:3:12);
    ESFMxy(i,:)=ESF_Mom(i,3:3:12);

    ESFVx (i,:)=ESF_Cis(i,1:2:8);
    ESFVy (i,:)=ESF_Cis(i,2:2:8);

    ReaRx (i,:)=ReApoio(i,1:2:8);
    ReaRy (i,:)=ReApoio(i,2:2:8);
```

```
    dez (i,:)=[deslVert(conec(i,1),1)...
               deslVert(conec(i,2),1)...
               deslVert(conec(i,3),1)...
               deslVert(conec(i,4),1)];

    dmx (i,:)=[ESFMx(i,1)...
               ESFMx(i,2)...
               ESFMx(i,3)...
               ESFMx(i,4)];
    dmy (i,:)=[ESFMy(i,1)...
               ESFMy(i,2)...
               ESFMy(i,3)...
               ESFMy(i,4)];
    dmxy(i,:)=[ESFMxy(i,1)...
               ESFMxy(i,2)...
               ESFMxy(i,3)...
               ESFMxy(i,4)];

    dvx (i,:)=[ESFVx(i,1)...
               ESFVx(i,2)...
               ESFVx(i,3)...
               ESFVx(i,4)];
    dvy (i,:)=[ESFVy(i,1)...
               ESFVy(i,2)...
               ESFVy(i,3)...
               ESFVy(i,4)];

    drx (i,:)=[ReaRx(i,1)...
               ReaRx(i,2)...
               ReaRx(i,3)...
               ReaRx(i,4)];
    dry (i,:)=[ReaRy(i,1)...
               ReaRy(i,2)...
               ReaRy(i,3)...
               ReaRy(i,4)];

end

deslMax=max(abs(deslVert))

MxMax =max(max(abs(ESFMx )));
MyMax =max(max(abs(ESFMy )));
MxyMax=max(max(abs(ESFMxy)));

MomMax=max(max(abs(ESF_Mom)));
CisMax=max(max(abs(ESF_Cis)));
ReaMax=max(max(abs(ReApoio)));

MxMax=max(max(ESFMx))
MyMax=max(max(ESFMy))

MxyMax=max(max(ESFMxy))
MxyMin=min(min(ESFMxy))

MxMin=min(min(ESFMx))
MyMin=min(min(ESFMy))

VxMax=max(max(ReaRx))
VyMax=max(max(ReaRy))
```

```
VxMin=min(min(ReaRx))
VyMin=min(min(ReaRy))

% Fator de deslocamento gráfico máximo
FatorDes=0.5*(comprimento+largura)/2;

dez =FatorDes*dez /deslMax+Zorigem;

dmx =FatorDes*dmx /MomMax +Zorigem;
dmy =FatorDes*dmy /MomMax +Zorigem;
dmxy=FatorDes*dmxy/MomMax +Zorigem;

dvx =FatorDes*dvx /CisMax +Zorigem;
dvy =FatorDes*dvy /CisMax +Zorigem;

drx =FatorDes*drx /ReaMax +Zorigem;
dry =FatorDes*dry /ReaMax +Zorigem;

x=ex';
y=ey';
z=ez';

figure(1)
title('deslocamento')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dez',edv','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(2)
title('Mx')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmx',ESFMx','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(3)
title('My')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmy',ESFMy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
```

```
axis tight
hold off

figure(4)
title('Mxy')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmxy',ESFMxy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(5)
title('Vx')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,drx',ReaRx','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(6)
title('Vy')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dry',ReaRy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off
```

Arquivo T_exemplo4.m

```
clear all
clc
% Exemplo 4 (Tarik)
%Dados de entrada de uma placa retangular apoiada nos quatro cantos

format shortG
% Matriz coordenadas dos nós Coord e a matriz conectividade dos
% elementos conec


comprimento=1;          % Comprimento da laje (paralelo a x)
largura=1;              % Largura da laje (paralelo a y)
NEX=20;                 % Número de elementos na direção x
NEY=20;                 % Número de elementos na direção y
nel=0;                  % Número de elementos da estrutura
dhx=comprimento/NEX;    % Comprimento de um elemento finito
dhy=largura/NEY;        % Largura de um elemento finito
Xorigem=-comprimento/2; % Coordenada global inicial x do desenho da
placa
Yorigem=-largura/2;     % Coordenada global inicial y do desenho da
placa
Zorigem=3;              % Coordenada global inicial z do desenho da
placa
nnoel=4;                % Número de nós de cada elemento
nglno=3;                % Número de graus de liberdade de cada nó do
elemento
nno=0;                  % Número de nós de toda a estrutura

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%
%  Numeração, coordenadas e conectividade dos nós dos elementos da
laje
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

%             y'
%             |                                   Esquema de numeração
%             |                                   dos nós do elemento.
%             |
%             |
%             n3 &&&&&&&&&&&&&&&&&&&&&& n4   ---
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             &.........................&   dhy
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             n1 &&&&&&&&&&&&&&&&&&&&&& n2 ------------------- x'
%
%             |---------- dhx ----------|

for i=1:NEX
```

```
    for j=1:NEY
        n1=j+(i-1)*(NEY+1);
        coord(n1,:)=[(i-1)*dhx+Xorigem (j-1)*dhy+Yorigem Zorigem];
        n2=j+i*(NEY+1);
        coord(n2,:)=[    i*dhx+Xorigem (j-1)*dhy+Yorigem Zorigem];
        n3=n1+1;
        coord(n3,:)=[(i-1)*dhx+Xorigem     j*dhy+Yorigem Zorigem];
        n4=n2+1;
        coord(n4,:)=[    i*dhx+Xorigem     j*dhy+Yorigem Zorigem];
        nel=nel+1;
        conec(nel,:)=[n1 n2 n4 n3];
        nno=n4;
    end
end


nglel=nno*nglno; % Número de graus de liberdade da estrutura


ex=zeros(nel,4); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ey=zeros(nel,4); % Coordenadas globais dos nós da estrutura
ez=zeros(nel,4); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:nel
    ex(i,:)=[coord(conec(i,1),1) coord(conec(i,2),1)
coord(conec(i,3),1) coord(conec(i,4),1)];
    ey(i,:)=[coord(conec(i,1),2) coord(conec(i,2),2)
coord(conec(i,3),2) coord(conec(i,4),2)];
    ez(i,:)=[coord(conec(i,1),3) coord(conec(i,2),3)
coord(conec(i,3),3) coord(conec(i,4),3)];
end

% gleNum identificação dos graus de liberdade de cada nó da estrutura
% Os graus de liberdade são, nessa ordem:
% vertical, rotação em volta de x e rotação em volta de y
gleNum=zeros(nno,nglno);
n=0;

for i=1:nno
    for j=1:nglno
        n=n+1;
        gleNum(i,j)=n;
    end
end

% Mtopologica Matriz topologica
Mtopologica=zeros(nel,nnoel*nglno+1);

for i=1:nel
   [st]=submatriztopologica(i,nnoel,nglno,conec,gleNum);
   Mtopologica(i,:)=[i st];
   SMtopologica(i,:)=[st];
end

gdlest=max(max(SMtopologica));

K=sparse(gdlest,gdlest);
f=zeros(gdlest,1);
%f(4)=100;
```

```
% Matrix Propriedades do elemento
t=0.01;
E=10.92e6;
v=0.3;

% Propriedades do elementos pe
pe=zeros(nel,3);
for i=1:nel
    pe(i,:)=[t E v];
end

% carga nodal equivalente para carga uniformemente distribuida
eq=zeros(nel,1);
%
for i=1:nel
    eq(i,:)=-1;
end

% rigidez e força de cada elemento e matriz de rigidez global

for i=1:nel

[Ke,fe]=matrizrigidezelementodeplacaACM(ex(i,:),ey(i,:),pe(i,:),eq(i,:
));
    [K,f]=montagemrigidezglobal(Mtopologica(i,:),K,Ke,f,fe);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Condições de contorno
% Os quatro cantos apoiados
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% gleImp graus de liberdade de cada nó que possui algum impedimento
gleImp=false(nno,nglno);
% gleNum deslocamento prescrito dos graus de liberdade impedidos de
cada nó
gleDPE=zeros(nno,nglno);

m=1;
for i=1:nno
    if coord(i,:) == [Xorigem             Yorigem          Zorigem] |
...
       coord(i,:) == [Xorigem+comprimento Yorigem          Zorigem] |
...
       coord(i,:) == [Xorigem+comprimento Yorigem+largura  Zorigem] |
...
       coord(i,:) == [Xorigem             Yorigem+largura  Zorigem];
% Lado paralelo ao eixo y
        gleImp(i,:)=[1 0 0];    % 1 = grau de liberdade preso, 0 = grau
de liberdade livre
        gleDPE(i,:)=[0 0 0];    % Valor do deslocamento prescrito
    end

    % Identificação dos graus de liberdade impedidos e
    % definição dos seus deslocamentos prescritos
    for j=1:nglno
        if(gleImp(i,j))
            cc(m,:)=[gleNum(i,j) gleDPE(i,j)];
            m=m+1;
        end
    end
```

```
end

% Resolve o sistema de equações e encontra os deslocamentos
% e as reações de apoio
[d,R]=resolvesistema(K,f,cc);
R1=max(R)
R2=min(R)

% Extrai os deslocamentos
[ed]=extraideslocamento(Mtopologica,d);

%% Cálculo do esforços e plotagem das reações internas e deslocamentos
edv=ed(:,1:3:12);
ESF_Mom=zeros(nel,12);
ESF_Cis=zeros(nel, 8);
ReApoio=zeros(nel, 8);

ESFMx =zeros(nel,4);
ESFMy =zeros(nel,4);
ESFMxy=zeros(nel,4);

ESFVx =zeros(nel,4);
ESFVy =zeros(nel,4);

ReaRx =zeros(nel,4);
ReaRy =zeros(nel,4);

deslVert=d(1:3:nglel,:);

format short e

dez=zeros(nel,4);

dmx=zeros(nel,4);
dmy=zeros(nel,4);
dmxy=zeros(nel,4);

dvx=zeros(nel,4);
dvy=zeros(nel,4);

drx=zeros(nel,4);
dry=zeros(nel,4);

for i=1:nel

[ESF_Cis(i,:),ESF_Mom(i,:),ReApoio(i,:)]=esforconaplaca(ex(i,:),ey(i,:
),pe(i,:),ed(i,:)); % Esforços na placa

    ESFMx (i,:)=ESF_Mom(i,1:3:12);
    ESFMy (i,:)=ESF_Mom(i,2:3:12);
    ESFMxy(i,:)=ESF_Mom(i,3:3:12);

    ESFVx (i,:)=ESF_Cis(i,1:2:8);
    ESFVy (i,:)=ESF_Cis(i,2:2:8);

    ReaRx (i,:)=ReApoio(i,1:2:8);
    ReaRy (i,:)=ReApoio(i,2:2:8);

    dez (i,:)=[deslVert(conec(i,1),1)...
```

```
                deslVert(conec(i,2),1)...
                deslVert(conec(i,3),1)...
                deslVert(conec(i,4),1)];

    dmx (i,:)=[ESFMx(i,1)...
                ESFMx(i,2)...
                ESFMx(i,3)...
                ESFMx(i,4)];
    dmy (i,:)=[ESFMy(i,1)...
                ESFMy(i,2)...
                ESFMy(i,3)...
                ESFMy(i,4)];
    dmxy(i,:)=[ESFMxy(i,1)...
                ESFMxy(i,2)...
                ESFMxy(i,3)...
                ESFMxy(i,4)];

    dvx (i,:)=[ESFVx(i,1)...
                ESFVx(i,2)...
                ESFVx(i,3)...
                ESFVx(i,4)];
    dvy (i,:)=[ESFVy(i,1)...
                ESFVy(i,2)...
                ESFVy(i,3)...
                ESFVy(i,4)];

    drx (i,:)=[ReaRx(i,1)...
                ReaRx(i,2)...
                ReaRx(i,3)...
                ReaRx(i,4)];
    dry (i,:)=[ReaRy(i,1)...
                ReaRy(i,2)...
                ReaRy(i,3)...
                ReaRy(i,4)];

end

deslMax=max(abs(deslVert))

MxMax =max(max(abs(ESFMx )));
MyMax =max(max(abs(ESFMy )));
MxyMax=max(max(abs(ESFMxy)));

MomMax=max(max(abs(ESF_Mom)));
CisMax=max(max(abs(ESF_Cis)));
ReaMax=max(max(abs(ReApoio)));

MxMax=max(max(ESFMx))
MyMax=max(max(ESFMy))

MxyMax=max(max(ESFMxy))
MxyMin=min(min(ESFMxy))

MxMin=min(min(ESFMx))
MyMin=min(min(ESFMy))

VxMax=max(max(ReaRx))
VyMax=max(max(ReaRy))

VxMin=min(min(ReaRx))
```

```
VyMin=min(min(ReaRy))

% Fator de deslocamento gráfico máximo
FatorDes=0.5*(comprimento+largura)/2;

dez =FatorDes*dez /deslMax+Zorigem;

dmx =FatorDes*dmx /MomMax +Zorigem;
dmy =FatorDes*dmy /MomMax +Zorigem;
dmxy=FatorDes*dmxy/MomMax +Zorigem;

dvx =FatorDes*dvx /CisMax +Zorigem;
dvy =FatorDes*dvy /CisMax +Zorigem;

drx =FatorDes*drx /ReaMax +Zorigem;
dry =FatorDes*dry /ReaMax +Zorigem;

x=ex';
y=ey';
z=ez';

figure(1)
title('deslocamento')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dez',edv','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(2)
title('Mx')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmx',ESFMx','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(3)
title('My')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmy',ESFMy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
```

```
hold off

figure(4)
title('Mxy')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmxy',ESFMxy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(5)
title('Vx')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,drx',ReaRx','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(6)
title('Vy')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dry',ReaRy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off
```

Arquivo T_exemplo5.m

```
clear all
clc
% Exemplo 5 (Tarik)
%Dados de entrada de uma placa retangular engastada nos quatro cantos

format shortG
% Matriz coordenadas dos nós Coord e a matriz conectividade dos
% elementos conec


comprimento=1;          % Comprimento da laje (paralelo a x)
largura=1;              % Largura da laje (paralelo a y)
NEX=20;                 % Número de elementos na direção x
NEY=20;                 % Número de elementos na direção y
nel=0;                  % Número de elementos da estrutura
dhx=comprimento/NEX;    % Comprimento de um elemento finito
dhy=largura/NEY;        % Largura de um elemento finito
Xorigem=-comprimento/2; % Coordenada global inicial x do desenho da
placa
Yorigem=-largura/2;     % Coordenada global inicial y do desenho da
placa
Zorigem=3;              % Coordenada global inicial z do desenho da
placa
nnoel=4;                % Número de nós de cada elemento
nglno=3;                % Número de graus de liberdade de cada nó do
elemento
nno=0;                  % Número de nós de toda a estrutura

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%
%  Numeração, coordenadas e conectividade dos nós dos elementos da
laje
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

%             y'
%             |                                  Esquema de numeração
%             |                                  dos nós do elemento.
%             |
%             |
%             n3 &&&&&&&&&&&&&&&&&&&&&& n4   ---
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             &.........................&   dhy
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             &.........................&    |
%             n1 &&&&&&&&&&&&&&&&&&&&&& n2 ------------------- x'
%
%             |---------- dhx ----------|

for i=1:NEX
```

```
    for j=1:NEY
        n1=j+(i-1)*(NEY+1);
        coord(n1,:)=[(i-1)*dhx+Xorigem (j-1)*dhy+Yorigem Zorigem];
        n2=j+i*(NEY+1);
        coord(n2,:)=[    i*dhx+Xorigem (j-1)*dhy+Yorigem Zorigem];
        n3=n1+1;
        coord(n3,:)=[(i-1)*dhx+Xorigem     j*dhy+Yorigem Zorigem];
        n4=n2+1;
        coord(n4,:)=[    i*dhx+Xorigem     j*dhy+Yorigem Zorigem];
        nel=nel+1;
        conec(nel,:)=[n1 n2 n4 n3];
        nno=n4;
    end
end


nglel=nno*nglno; % Número de graus de liberdade da estrutura


ex=zeros(nel,4); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ey=zeros(nel,4); % Coordenadas globais dos nós da estrutura
ez=zeros(nel,4); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:nel
    ex(i,:)=[coord(conec(i,1),1) coord(conec(i,2),1)
coord(conec(i,3),1) coord(conec(i,4),1)];
    ey(i,:)=[coord(conec(i,1),2) coord(conec(i,2),2)
coord(conec(i,3),2) coord(conec(i,4),2)];
    ez(i,:)=[coord(conec(i,1),3) coord(conec(i,2),3)
coord(conec(i,3),3) coord(conec(i,4),3)];
end

% gleNum identificação dos graus de liberdade de cada nó da estrutura
% Os graus de liberdade são, nessa ordem:
% vertical, rotação em volta de x e rotação em volta de y
gleNum=zeros(nno,nglno);
n=0;

for i=1:nno
    for j=1:nglno
        n=n+1;
        gleNum(i,j)=n;
    end
end

% Mtopologica Matriz topologica
Mtopologica=zeros(nel,nnoel*nglno+1);

for i=1:nel
   [st]=submatriztopologica(i,nnoel,nglno,conec,gleNum);
   Mtopologica(i,:)=[i st];
   SMtopologica(i,:)=[st];
end

gdlest=max(max(SMtopologica));

K=sparse(gdlest,gdlest);
f=zeros(gdlest,1);
%f(4)=100;
```

```
% Matrix Propriedades do elemento
t=0.01;
E=10.92e6;
v=0.3;

% Propriedades do elementos pe
pe=zeros(nel,3);
for i=1:nel
    pe(i,:)=[t E v];
end

% carga nodal equivalente para carga uniformemente distribuida
eq=zeros(nel,1);
%
for i=1:nel
    eq(i,:)=-1;
end

% rigidez e força de cada elemento e matriz de rigidez global

for i=1:nel

[Ke,fe]=matrizrigidezelementodeplacaACM(ex(i,:),ey(i,:),pe(i,:),eq(i,:
));
    [K,f]=montagemrigidezglobal(Mtopologica(i,:),K,Ke,f,fe);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Condições de contorno
% Os quatro cantos engastados
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% gleImp graus de liberdade de cada nó que possui algum impedimento
gleImp=false(nno,nglno);
% gleNum deslocamento prescrito dos graus de liberdade impedidos de
cada nó
gleDPE=zeros(nno,nglno);

m=1;
for i=1:nno
    if coord(i,:) == [Xorigem             Yorigem          Zorigem] |
...
       coord(i,:) == [Xorigem+comprimento Yorigem          Zorigem] |
...
       coord(i,:) == [Xorigem+comprimento Yorigem+largura  Zorigem] |
...
       coord(i,:) == [Xorigem             Yorigem+largura  Zorigem];
% Lado paralelo ao eixo y
        gleImp(i,:)=[1 1 1];    % 1 = grau de liberdade preso, 0 = grau
de liberdade livre
        gleDPE(i,:)=[0 0 0];    % Valor do deslocamento prescrito
    end

    % Identificação dos graus de liberdade impedidos e
    % definição dos seus deslocamentos prescritos
    for j=1:nglno
        if(gleImp(i,j))
            cc(m,:)=[gleNum(i,j) gleDPE(i,j)];
            m=m+1;
        end
    end
```

```
end

% Resolve o sistema de equações e encontra os deslocamentos
% e as reações de apoio
[d,R]=resolvesistema(K,f,cc);
R1=max(R)
R2=min(R)
%Reacoes=[R(1);R(2);R(3);R(10);R(11);R(12)]

% Extrai os deslocamentos
[ed]=extraideslocamento(Mtopologica,d);

%% Cálculo do esforços e plotagem das reações internas e deslocamentos
edv=ed(:,1:3:12);
ESF_Mom=zeros(nel,12);
ESF_Cis=zeros(nel, 8);
ReApoio=zeros(nel, 8);

ESFMx =zeros(nel,4);
ESFMy =zeros(nel,4);
ESFMxy=zeros(nel,4);

ESFVx =zeros(nel,4);
ESFVy =zeros(nel,4);

ReaRx =zeros(nel,4);
ReaRy =zeros(nel,4);

deslVert=d(1:3:nglel,:);

format short e

dez=zeros(nel,4);

dmx=zeros(nel,4);
dmy=zeros(nel,4);
dmxy=zeros(nel,4);

dvx=zeros(nel,4);
dvy=zeros(nel,4);

drx=zeros(nel,4);
dry=zeros(nel,4);

for i=1:nel

[ESF_Cis(i,:),ESF_Mom(i,:),ReApoio(i,:)]=esforconaplaca(ex(i,:),ey(i,:
),pe(i,:),ed(i,:)); % Esforços na placa

    ESFMx (i,:)=ESF_Mom(i,1:3:12);
    ESFMy (i,:)=ESF_Mom(i,2:3:12);
    ESFMxy(i,:)=ESF_Mom(i,3:3:12);

    ESFVx (i,:)=ESF_Cis(i,1:2:8);
    ESFVy (i,:)=ESF_Cis(i,2:2:8);

    ReaRx (i,:)=ReApoio(i,1:2:8);
    ReaRy (i,:)=ReApoio(i,2:2:8);
```

```
    dez (i,:)=[deslVert(conec(i,1),1)...
               deslVert(conec(i,2),1)...
               deslVert(conec(i,3),1)...
               deslVert(conec(i,4),1)];

    dmx (i,:)=[ESFMx(i,1)...
               ESFMx(i,2)...
               ESFMx(i,3)...
               ESFMx(i,4)];
    dmy (i,:)=[ESFMy(i,1)...
               ESFMy(i,2)...
               ESFMy(i,3)...
               ESFMy(i,4)];
    dmxy(i,:)=[ESFMxy(i,1)...
               ESFMxy(i,2)...
               ESFMxy(i,3)...
               ESFMxy(i,4)];

    dvx (i,:)=[ESFVx(i,1)...
               ESFVx(i,2)...
               ESFVx(i,3)...
               ESFVx(i,4)];
    dvy (i,:)=[ESFVy(i,1)...
               ESFVy(i,2)...
               ESFVy(i,3)...
               ESFVy(i,4)];

    drx (i,:)=[ReaRx(i,1)...
               ReaRx(i,2)...
               ReaRx(i,3)...
               ReaRx(i,4)];
    dry (i,:)=[ReaRy(i,1)...
               ReaRy(i,2)...
               ReaRy(i,3)...
               ReaRy(i,4)];

end

deslMax=max(abs(deslVert))

MxMax =max(max(abs(ESFMx )));
MyMax =max(max(abs(ESFMy )));
MxyMax=max(max(abs(ESFMxy)));

MomMax=max(max(abs(ESF_Mom)));
CisMax=max(max(abs(ESF_Cis)));
ReaMax=max(max(abs(ReApoio)));

MxMax=max(max(ESFMx))
MyMax=max(max(ESFMy))

MxyMax=max(max(ESFMxy))
MxyMin=min(min(ESFMxy))

MxMin=min(min(ESFMx))
MyMin=min(min(ESFMy))

VxMax=max(max(ReaRx))
VyMax=max(max(ReaRy))
```

```
VxMin=min(min(ReaRx))
VyMin=min(min(ReaRy))

% Fator de deslocamento gráfico máximo
FatorDes=0.5*(comprimento+largura)/2;

dez =FatorDes*dez /deslMax+Zorigem;

dmx =FatorDes*dmx /MomMax +Zorigem;
dmy =FatorDes*dmy /MomMax +Zorigem;
dmxy=FatorDes*dmxy/MomMax +Zorigem;

dvx =FatorDes*dvx /CisMax +Zorigem;
dvy =FatorDes*dvy /CisMax +Zorigem;

drx =FatorDes*drx /ReaMax +Zorigem;
dry =FatorDes*dry /ReaMax +Zorigem;

x=ex';
y=ey';
z=ez';

figure(1)
title('deslocamento')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dez',edv','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(2)
title('Mx')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmx',ESFMx','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(3)
title('My')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmy',ESFMy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
```

```
axis tight
hold off

figure(4)
title('Mxy')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmxy',ESFMxy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(5)
title('Vx')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,drx',ReaRx','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(6)
title('Vy')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dry',ReaRy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off
```

Arquivo T_exemplo6.m

```
clear all
clc
% Exemplo 6 (Tarik)
% Dados de entrada de uma placa retangular engastada em dois lados
% adjacentes e duas bordas livres

format shortG
% Matriz coordenadas dos nós Coord e a matriz conectividade dos
% elementos conec


comprimento=1;           % Comprimento da laje (paralelo a x)
largura=1;               % Largura da laje (paralelo a y)
NEX=20;                  % Número de elementos na direção x
NEY=20;                  % Número de elementos na direção y
nel=0;                   % Número de elementos da estrutura
dhx=comprimento/NEX;     % Comprimento de um elemento finito
dhy=largura/NEY;         % Largura de um elemento finito
Xorigem=-comprimento/2; % Coordenada global inicial x do desenho da
placa
Yorigem=-largura/2;      % Coordenada global inicial y do desenho da
placa
Zorigem=3;               % Coordenada global inicial z do desenho da
placa
nnoel=4;                 % Número de nós de cada elemento
nglno=3;                 % Número de graus de liberdade de cada nó do
elemento
nno=0;                   % Número de nós de toda a estrutura

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%
%  Numeração, coordenadas e conectividade dos nós dos elementos da
laje
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

%             y'
%             |                                  Esquema de numeração
%             |                                  dos nós do elemento.
%             |
%             |
%             n3 &&&&&&&&&&&&&&&&&&&& n4   ---
%             &........................&    |
%             &........................&    |
%             &........................&    |
%             &........................&    |
%             &........................&    |
%             &........................&   dhy
%             &........................&    |
%             &........................&    |
%             &........................&    |
%             &........................&    |
%             &........................&    |
%             n1 &&&&&&&&&&&&&&&&&&&& n2 ------------------- x'
%
%             |---------- dhx ----------|
```

```
for i=1:NEX
    for j=1:NEY
        n1=j+(i-1)*(NEY+1);
        coord(n1,:)=[(i-1)*dhx+Xorigem (j-1)*dhy+Yorigem Zorigem];
        n2=j+i*(NEY+1);
        coord(n2,:)=[    i*dhx+Xorigem (j-1)*dhy+Yorigem Zorigem];
        n3=n1+1;
        coord(n3,:)=[(i-1)*dhx+Xorigem     j*dhy+Yorigem Zorigem];
        n4=n2+1;
        coord(n4,:)=[    i*dhx+Xorigem     j*dhy+Yorigem Zorigem];
        nel=nel+1;
        conec(nel,:)=[n1 n2 n4 n3];
        nno=n4;
    end
end


nglel=nno*nglno; % Número de graus de liberdade da estrutura


ex=zeros(nel,4); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ey=zeros(nel,4); % Coordenadas globais dos nós da estrutura
ez=zeros(nel,4); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:nel
    ex(i,:)=[coord(conec(i,1),1) coord(conec(i,2),1)
coord(conec(i,3),1) coord(conec(i,4),1)];
    ey(i,:)=[coord(conec(i,1),2) coord(conec(i,2),2)
coord(conec(i,3),2) coord(conec(i,4),2)];
    ez(i,:)=[coord(conec(i,1),3) coord(conec(i,2),3)
coord(conec(i,3),3) coord(conec(i,4),3)];
end

% gleNum identificação dos graus de liberdade de cada nó da estrutura
% Os graus de liberdade são, nessa ordem:
% vertical, rotação em volta de x e rotação em volta de y
gleNum=zeros(nno,nglno);
n=0;

for i=1:nno
    for j=1:nglno
        n=n+1;
        gleNum(i,j)=n;
    end
end

% Mtopologica Matriz topologica
Mtopologica=zeros(nel,nnoel*nglno+1);

for i=1:nel
   [st]=submatriztopologica(i,nnoel,nglno,conec,gleNum);
   Mtopologica(i,:)=[i st];
   SMtopologica(i,:)=[st];
end

gdlest=max(max(SMtopologica));

K=sparse(gdlest,gdlest);
f=zeros(gdlest,1);
%f(4)=100;
```

```
% Matrix Propriedades do elemento
t=0.01;
E=10.92e6;
v=0.3;

% Propriedades do elementos pe
pe=zeros(nel,3);
for i=1:nel
    pe(i,:)=[t E v];
end

% carga nodal equivalente para carga uniformemente distribuida
eq=zeros(nel,1);
%
for i=1:nel
    eq(i,:)=-1;
end

% rigidez e força de cada elemento e matriz de rigidez global

for i=1:nel

[Ke,fe]=matrizrigidezelementodeplacaACM(ex(i,:),ey(i,:),pe(i,:),eq(i,:
));
    [K,f]=montagemrigidezglobal(Mtopologica(i,:),K,Ke,f,fe);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Condições de contorno
% Dois lados adjacentes engastados
% e duas bordas livres
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% gleImp graus de liberdade de cada nó que possui algum impedimento
gleImp=false(nno,nglno);
% gleNum deslocamento prescrito dos graus de liberdade impedidos de
cada nó
gleDPE=zeros(nno,nglno);

m=1;
for i=1:nno
    if coord(i,1) == Xorigem;  % Lado paralelo ao eixo y
        gleImp(i,:)=[1 1 1];   % 1 = grau de liberdade preso, 0 = grau
de liberdade livre
        gleDPE(i,:)=[0 0 0];   % Valor do deslocamento prescrito
    end

    if coord(i,2) == Yorigem;  % Lado paralelo ao eixo y
        gleImp(i,:)=[1 1 1];   % 1 = grau de liberdade preso, 0 = grau
de liberdade livre
        gleDPE(i,:)=[0 0 0];   % Valor do deslocamento prescrito
    end

    % Identificação dos graus de liberdade impedidos e
    % definição dos seus deslocamentos prescritos
    for j=1:nglno
        if(gleImp(i,j))
            cc(m,:)=[gleNum(i,j) gleDPE(i,j)];
            m=m+1;
        end
```

```
    end
end

% Resolve o sistema de equações e encontra os deslocamentos
% e as reações de apoio
[d,R]=resolvesistema(K,f,cc);
R1=max(R)
R2=min(R)
%Reacoes=[R(1);R(2);R(3);R(10);R(11);R(12)]

% Extrai os deslocamentos
[ed]=extraideslocamento(Mtopologica,d);

%% Cálculo do esforços e plotagem das reações internas e deslocamentos
edv=ed(:,1:3:12);
ESF_Mom=zeros(nel,12);
ESF_Cis=zeros(nel, 8);
ReApoio=zeros(nel, 8);

ESFMx =zeros(nel,4);
ESFMy =zeros(nel,4);
ESFMxy=zeros(nel,4);

ESFVx =zeros(nel,4);
ESFVy =zeros(nel,4);

ReaRx =zeros(nel,4);
ReaRy =zeros(nel,4);

deslVert=d(1:3:nglel,:);

format short e

dez=zeros(nel,4);

dmx=zeros(nel,4);
dmy=zeros(nel,4);
dmxy=zeros(nel,4);

dvx=zeros(nel,4);
dvy=zeros(nel,4);

drx=zeros(nel,4);
dry=zeros(nel,4);

for i=1:nel

[ESF_Cis(i,:),ESF_Mom(i,:),ReApoio(i,:)]=esforconaplaca(ex(i,:),ey(i,:
),pe(i,:),ed(i,:)); % Esforços na placa

    ESFMx (i,:)=ESF_Mom(i,1:3:12);
    ESFMy (i,:)=ESF_Mom(i,2:3:12);
    ESFMxy(i,:)=ESF_Mom(i,3:3:12);

    ESFVx (i,:)=ESF_Cis(i,1:2:8);
    ESFVy (i,:)=ESF_Cis(i,2:2:8);

    ReaRx (i,:)=ReApoio(i,1:2:8);
    ReaRy (i,:)=ReApoio(i,2:2:8);
```

```
    dez (i,:)=[deslVert(conec(i,1),1)...
               deslVert(conec(i,2),1)...
               deslVert(conec(i,3),1)...
               deslVert(conec(i,4),1)];

    dmx (i,:)=[ESFMx(i,1)...
               ESFMx(i,2)...
               ESFMx(i,3)...
               ESFMx(i,4)];
    dmy (i,:)=[ESFMy(i,1)...
               ESFMy(i,2)...
               ESFMy(i,3)...
               ESFMy(i,4)];
    dmxy(i,:)=[ESFMxy(i,1)...
               ESFMxy(i,2)...
               ESFMxy(i,3)...
               ESFMxy(i,4)];

    dvx (i,:)=[ESFVx(i,1)...
               ESFVx(i,2)...
               ESFVx(i,3)...
               ESFVx(i,4)];
    dvy (i,:)=[ESFVy(i,1)...
               ESFVy(i,2)...
               ESFVy(i,3)...
               ESFVy(i,4)];

    drx (i,:)=[ReaRx(i,1)...
               ReaRx(i,2)...
               ReaRx(i,3)...
               ReaRx(i,4)];
    dry (i,:)=[ReaRy(i,1)...
               ReaRy(i,2)...
               ReaRy(i,3)...
               ReaRy(i,4)];

end

deslMax=max(abs(deslVert))

MxMax =max(max(abs(ESFMx )));
MyMax =max(max(abs(ESFMy )));
MxyMax=max(max(abs(ESFMxy)));

MomMax=max(max(abs(ESF_Mom)));
CisMax=max(max(abs(ESF_Cis)));
ReaMax=max(max(abs(ReApoio)));

MxMax=max(max(ESFMx))
MyMax=max(max(ESFMy))

MxyMax=max(max(ESFMxy))
MxyMin=min(min(ESFMxy))

MxMin=min(min(ESFMx))
MyMin=min(min(ESFMy))

VxMax=max(max(ReaRx))
VyMax=max(max(ReaRy))
```

```
VxMin=min(min(ReaRx))
VyMin=min(min(ReaRy))

% Fator de deslocamento gráfico máximo
FatorDes=0.5*(comprimento+largura)/2;

dez =FatorDes*dez /deslMax+Zorigem;

dmx =FatorDes*dmx /MomMax +Zorigem;
dmy =FatorDes*dmy /MomMax +Zorigem;
dmxy=FatorDes*dmxy/MomMax +Zorigem;

dvx =FatorDes*dvx /CisMax +Zorigem;
dvy =FatorDes*dvy /CisMax +Zorigem;

drx =FatorDes*drx /ReaMax +Zorigem;
dry =FatorDes*dry /ReaMax +Zorigem;

x=ex';
y=ey';
z=ez';

figure(1)
title('deslocamento')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dez',edv','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(2)
title('Mx')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmx',ESFMx','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(3)
title('My')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmy',ESFMy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
```

```
axis equal
axis tight
hold off

figure(4)
title('Mxy')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dmxy',ESFMxy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(5)
title('Vx')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,drx',ReaRx','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off

figure(6)
title('Vy')
grid on
hold on
xlabel('x','FontSize',16,'FontWeight','bold','Color','r')
ylabel('y','FontSize',16,'FontWeight','bold','Color','g')
zlabel('z','FontSize',16,'FontWeight','bold','Color','b')
fill3(x,y,z,'r','EdgeAlpha',0.3,'FaceAlpha',0.0,'EdgeColor','r')
fill3(x,y,dry',ReaRy','EdgeAlpha',0.3,'LineWidth',0.1);
colorbar;
axis equal
axis tight
hold off
```

Função esforconaplaca.m

```
function [ESFCis,ESFMom,ReApoio]=esforconaplaca(ex,ey,pe,ed)

t=pe(1);            % Espessura da placa
E=pe(2);            % Módulo de elasticidade
v=pe(3);            % Coeficiente de Poisson
dhx=ex(3)-ex(1);    % Comprimento do elemento
dhy=ey(3)-ey(1);    % Largura do elemento

% Matriz inversa da matriz de polinômios de Pascal
InvA=[1   0   0   0   0   0   0   0   0   0   0   0;
      0   0  -1   0   0   0   0   0   0   0   0   0;
      0   1   0   0   0   0   0   0   0   0   0   0;
      -(3/dhx^2)    0   2/dhx   3/dhx^2   0   1/dhx   0   0   0   0   0
0;
      -(1/(dhx*dhy))  -(1/dhx)   1/dhy   1/(dhx*dhy)   1/dhx   0   -
(1/(dhx*dhy))   0   0   1/(dhx*dhy)   0   -(1/dhy);
      -(3/dhy^2)  -(2/dhy)   0   0   0   0   0   0   0   3/dhy^2   -
(1/dhy)   0;
      2/dhx^3   0   -(1/dhx^2)   -(2/dhx^3)   0   -(1/dhx^2)   0   0   0
0   0   0;
      3/(dhx^2*dhy)   0   -(2/(dhx*dhy))   -(3/(dhx^2*dhy))   0   -
(1/(dhx*dhy))   3/(dhx^2*dhy)   0   1/(dhx*dhy)   -(3/(dhx^2*dhy))   0
2/(dhx*dhy);
      3/(dhx*dhy^2)   2/(dhx*dhy)   0   -(3/(dhx*dhy^2))   -
(2/(dhx*dhy))   0   3/(dhx*dhy^2)   -(1/(dhx*dhy))   0   -
(3/(dhx*dhy^2))   1/(dhx*dhy)   0;
      2/dhy^3   1/dhy^2   0   0   0   0   0   0   0   -(2/dhy^3)
1/dhy^2   0;
      -(2/(dhx^3*dhy))   0   1/(dhx^2*dhy)   2/(dhx^3*dhy)   0
1/(dhx^2*dhy)   -(2/(dhx^3*dhy))   0   -(1/(dhx^2*dhy))   2/(dhx^3*dhy)
0   -(1/(dhx^2*dhy));
      -(2/(dhx*dhy^3))   -(1/(dhx*dhy^2))   0   2/(dhx*dhy^3)
1/(dhx*dhy^2)   0   -(2/(dhx*dhy^3))   1/(dhx*dhy^2)   0
2/(dhx*dhy^3)   -(1/(dhx*dhy^2))   0];

% Coeficientes do campo de deslocamentos
Coef=InvA*ed';

% Vetor de deformações generalizadas por flexão e torção
DefFlex=[0   0   2   0   0   6*0     0       0       0      6*0*0
0;              %----------------------
         0   0   0   0   0   2   0       0       2*0     6*0    0
6*0*0;          % x=0,   y=0   (Nó 1)
         0   0   0   0   2   0   0       4*0     4*0     0      6*0^2
6*0^2;          %----------------------
         0   0   0   2   0   0   6*dhx   2*0     0       0
6*dhx*0    0;               %----------------------
         0   0   0   0   0   2   0       0       2*dhx   6*0    0
6*dhx*0;        % x=dhx, y=0   (Nó 2)
         0   0   0   0   2   0   0       4*dhx   4*0     0
6*dhx^2    6*0^2;           %----------------------
         0   0   0   2   0   0   6*dhx   2*dhy   0       0
6*dhx*dhy  0;               %----------------------
         0   0   0   0   0   2   0       0       2*dhx   6*dhy  0
6*dhx*dhy;      % x=dhx, y=dhy (Nó 4)
         0   0   0   0   2   0   0       4*dhx   4*dhy   0
6*dhx^2    6*dhy^2;         %----------------------
         0   0   0   2   0   0   6*0     2*dhy   0       0
6*0*dhy    0;               %----------------------
```

```
          0   0   0   0   0   2   0       0       2*0     6*dhy  0
6*0*dhy;        % x=0,   y=dhy (Nó 3)
          0   0   0   0   2   0   0       4*0     4*dhy   0      6*0^2
6*dhy^2]*Coef;  %----------------------

% Matriz constitutiva generalizada à flexão para material homogêneo
D=E*t^3/(12*(1-v^2))*[1 v 0;
                      v 1 0;
                      0 0 (1-v)/2];

% Momentos fletores (Mx, My e Mxy)
M1=-D*DefFlex( 1:3 ,:);   % Nó 1
M2=-D*DefFlex( 4:6 ,:);   % Nó 2
M3=-D*DefFlex( 7:9 ,:);   % Nó 4*
M4=-D*DefFlex(10:12,:);   % Nó 3
ESFMom=[M1;M2;M3;M4]';

% Matriz de deformações por cisalhamento suplementar
DefCisT=[ 0   0   0   0   0   0   6   0   0   0       0       0;
%----------------------
          0   0   0   0   0   0   0   0   0   6       0       0;
% x=0,   y=0   (Nó 1)
          0   0   0   0   0   0   0   0   2   0       0       0;
%
          0   0   0   0   0   0   0   2   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   6       0   6*dhx;
% x=dhx, y=0   (Nó 2)
          0   0   0   0   0   0   0   0   2   0       0       0;
%
          0   0   0   0   0   0   0   2   0   0   6*dhx       0;
%----------------------
          0   0   0   0   0   0   6   0   0   0   6*dhy       0;
%----------------------
          0   0   0   0   0   0   0   0   0   6       0   6*dhx;
% x=dhx, y=dhy (Nó 4)
          0   0   0   0   0   0   0   0   2   0       0   6*dhy;
%
          0   0   0   0   0   0   0   2   0   0   6*dhx       0;
%----------------------
          0   0   0   0   0   0   6   0   0   0   6*dhy       0;
%----------------------
          0   0   0   0   0   0   0   0   0   6       0       0;
% x=0,   y=dhy (Nó 3)
          0   0   0   0   0   0   0   0   2   0       0   6*dhy;
%
          0   0   0   0   0   0   0   2   0   0       0       0]*Coef;
%----------------------

% Matriz constitutiva generalizada ao cisalhamento suplementar para
material homogêneo
D=E*t^3/(12*(1-v^2))*[1   0   1   0;
                      0   1   0   1];

% Esforços de cisalhamento suplementar (Vx, Vy)
V1=-D*DefCisT( 1:4 ,:);   % Nó 1
V2=-D*DefCisT( 5:8 ,:);   % Nó 2
V3=-D*DefCisT( 9:12,:);   % Nó 4*
V4=-D*DefCisT(13:16,:);   % Nó 3
ESFCis=[V1;V2;V3;V4]';
```

```
% Matriz constitutiva com parcela de torção para material homogêneo
D=E*t^3/(12*(1-v^2))*[ 1   0   2-v   0;
                       0   1    0   2-v];

DefReA=[ 0   0   0   0   0   0   6   0   0   0        0        0;
%----------------------
         0   0   0   0   0   0   0   0   0   6        0        0;
% x=0,   y=0   (Nó 1)
         0   0   0   0   0   0   0   0   2   0        0        0;
%
         0   0   0   0   0   0   0   2   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   6        0   6*dhx;
% x=dhx, y=0   (Nó 2)
         0   0   0   0   0   0   0   0   2   0        0        0;
%
         0   0   0   0   0   0   0   2   0   0   6*dhx        0;
%----------------------
         0   0   0   0   0   0   6   0   0   0   6*dhy        0;
%----------------------
         0   0   0   0   0   0   0   0   0   6        0   6*dhx;
% x=dhx, y=dhy (Nó 4)
         0   0   0   0   0   0   0   0   2   0        0   6*dhy;
%
         0   0   0   0   0   0   0   2   0   0   6*dhx        0;
%----------------------
         0   0   0   0   0   0   6   0   0   0   6*dhy        0;
%----------------------
         0   0   0   0   0   0   0   0   0   6        0        0;
% x=0,   y=dhy (Nó 3)
         0   0   0   0   0   0   0   0   2   0        0   6*dhy;
%
         0   0   0   0   0   0   0   2   0   0        0        0]*Coef;
%----------------------

% Reações de apoio (Rx, Ry)
R1=-D*DefReA( 1:4 ,:);   % Nó 1
R2=-D*DefReA( 5:8 ,:);   % Nó 2
R3=-D*DefReA( 9:12,:);   % Nó 4*
R4=-D*DefReA(13:16,:);   % Nó 3
ReApoio=[R1;R2;R3;R4]';
%-------------------------end--------------------------------
```

Função extraideslocamento.m

```
function [ed]=extraideslocamento(Mtopologica,d)


    [nel,n]=size(Mtopologica);
%
    st=Mtopologica(:,2:n);
%
    for i = 1:nel
        ed(i,1:(n-1))=d(st(i,:))';
    end
%
%--------------------------end-------------------------------
```

Função matrizrigidezelementodeplacaACM.m

```
function [Ke,fe]=matrizrigidezelementodeplacaACM(ex,ey,pe,eq)


t=pe(1); % Espessura da placa
E=pe(2); % Módulo de elasticidade
v=pe(3); % Coeficiente de Poisson

a=ex(3)-ex(1); % Lado dhx do elemento de placa
b=ey(3)-ey(1); % Lado dhy do elemento de placa

% Coeficiente de rigidez à flexão da placa
D=E*t^3/(12*(1-v^2));
%
Keq=[ (2*D*(10*a^4 + 10*b^4 + 7*a^2*b^2 - 2*a^2*b^2*v))/(5*a^3*b^3),
(D*(4*v + 1))/(5*a) + (2*D*a)/b^2,      - (D*(4*v + 1))/(5*b) -
(2*D*b)/a^2,  (2*D*(5*a^4 - 10*b^4 - 7*a^2*b^2 +
2*a^2*b^2*v))/(5*a^3*b^3),          (D*a)/b^2 - (D*(4*v + 1))/(5*a),
(D*(v - 1))/(5*b) - (2*D*b)/a^2,  -(2*D*(5*a^4 + 5*b^4 - 7*a^2*b^2 +
2*a^2*b^2*v))/(5*a^3*b^3),            (D*(v - 1))/(5*a) + (D*a)/b^2,
- (D*(v - 1))/(5*b) - (D*b)/a^2, -(2*D*(10*a^4 - 5*b^4 + 7*a^2*b^2 -
2*a^2*b^2*v))/(5*a^3*b^3),          (2*D*a)/b^2 - (D*(v - 1))/(5*a),
(D*(4*v + 1))/(5*b) - (D*b)/a^2;
                                  (D*(4*v + 1))/(5*a) + (2*D*a)/b^2,
(4*D*a)/(3*b) - (4*D*b*(v - 1))/(15*a),
-D*v,                               (D*a)/b^2 - (D*(4*v + 1))/(5*a),
(2*D*a)/(3*b) + (2*D*b*(2*v - 2))/(15*a),
0,                                - (D*(v - 1))/(5*a) - (D*a)/b^2,
(D*a)/(3*b) - (D*b*(v - 1))/(15*a),
0,                                (D*(v - 1))/(5*a) - (2*D*a)/b^2,
(2*D*a)/(3*b) + (D*b*(v - 1))/(15*a),
0;
                                - (D*(4*v + 1))/(5*b) - (2*D*b)/a^2,
-D*v,   (4*D*b)/(3*a) - (4*D*a*(v - 1))/(15*b),
(2*D*b)/a^2 - (D*(v - 1))/(5*b),
0,     (2*D*b)/(3*a) + (D*a*(v - 1))/(15*b),
(D*(v - 1))/(5*b) + (D*b)/a^2,
0,       (D*b)/(3*a) - (D*a*(v - 1))/(15*b),
(D*(4*v + 1))/(5*b) - (D*b)/a^2,
0, (2*D*b)/(3*a) + (2*D*a*(2*v - 2))/(15*b);
       (2*D*(5*a^4 - 10*b^4 - 7*a^2*b^2 + 2*a^2*b^2*v))/(5*a^3*b^3),
(D*a)/b^2 - (D*(4*v + 1))/(5*a),          (2*D*b)/a^2 - (D*(v -
```

```
1))/(5*b), (2*D*(10*a^4 + 10*b^4 + 7*a^2*b^2 -
2*a^2*b^2*v))/(5*a^3*b^3),        (D*(4*v + 1))/(5*a) + (2*D*a)/b^2,
(D*(4*v + 1))/(5*b) + (2*D*b)/a^2, -(2*D*(10*a^4 - 5*b^4 + 7*a^2*b^2 -
2*a^2*b^2*v))/(5*a^3*b^3),          (2*D*a)/b^2 - (D*(v - 1))/(5*a),
(D*b)/a^2 - (D*(4*v + 1))/(5*b),  -(2*D*(5*a^4 + 5*b^4 - 7*a^2*b^2 +
2*a^2*b^2*v))/(5*a^3*b^3),            (D*(v - 1))/(5*a) + (D*a)/b^2,
(D*(v - 1))/(5*b) + (D*b)/a^2;
                                     (D*a)/b^2 - (D*(4*v + 1))/(5*a),
(2*D*a)/(3*b) + (2*D*b*(2*v - 2))/(15*a),
0,                                (D*(4*v + 1))/(5*a) + (2*D*a)/b^2,
(4*D*a)/(3*b) - (4*D*b*(v - 1))/(15*a),
D*v,                               (D*(v - 1))/(5*a) - (2*D*a)/b^2,
(2*D*a)/(3*b) + (D*b*(v - 1))/(15*a),
0,                                - (D*(v - 1))/(5*a) - (D*a)/b^2,
(D*a)/(3*b) - (D*b*(v - 1))/(15*a),
0;
                                     (D*(v - 1))/(5*b) - (2*D*b)/a^2,
0,     (2*D*b)/(3*a) + (D*a*(v - 1))/(15*b),
(D*(4*v + 1))/(5*b) + (2*D*b)/a^2,
D*v,   (4*D*b)/(3*a) - (4*D*a*(v - 1))/(15*b),
(D*b)/a^2 - (D*(4*v + 1))/(5*b),
0, (2*D*b)/(3*a) + (2*D*a*(2*v - 2))/(15*b),
- (D*(v - 1))/(5*b) - (D*b)/a^2,
0,       (D*b)/(3*a) - (D*a*(v - 1))/(15*b);
       -(2*D*(5*a^4 + 5*b^4 - 7*a^2*b^2 + 2*a^2*b^2*v))/(5*a^3*b^3),
- (D*(v - 1))/(5*a) - (D*a)/b^2,            (D*(v - 1))/(5*b) +
(D*b)/a^2, -(2*D*(10*a^4 - 5*b^4 + 7*a^2*b^2 -
2*a^2*b^2*v))/(5*a^3*b^3),          (D*(v - 1))/(5*a) - (2*D*a)/b^2,
(D*b)/a^2 - (D*(4*v + 1))/(5*b), (2*D*(10*a^4 + 10*b^4 + 7*a^2*b^2 -
2*a^2*b^2*v))/(5*a^3*b^3),      - (D*(4*v + 1))/(5*a) - (2*D*a)/b^2,
(D*(4*v + 1))/(5*b) + (2*D*b)/a^2,  (2*D*(5*a^4 - 10*b^4 - 7*a^2*b^2 +
2*a^2*b^2*v))/(5*a^3*b^3),          (D*(4*v + 1))/(5*a) - (D*a)/b^2,
(2*D*b)/a^2 - (D*(v - 1))/(5*b);
                                       (D*(v - 1))/(5*a) + (D*a)/b^2,
(D*a)/(3*b) - (D*b*(v - 1))/(15*a),
0,                                  (2*D*a)/b^2 - (D*(v - 1))/(5*a),
(2*D*a)/(3*b) + (D*b*(v - 1))/(15*a),
0,                               - (D*(4*v + 1))/(5*a) - (2*D*a)/b^2,
(4*D*a)/(3*b) - (4*D*b*(v - 1))/(15*a),
-D*v,                              (D*(4*v + 1))/(5*a) - (D*a)/b^2,
(2*D*a)/(3*b) + (2*D*b*(2*v - 2))/(15*a),
0;
                                     - (D*(v - 1))/(5*b) - (D*b)/a^2,
0,       (D*b)/(3*a) - (D*a*(v - 1))/(15*b),
(D*b)/a^2 - (D*(4*v + 1))/(5*b),
0, (2*D*b)/(3*a) + (2*D*a*(2*v - 2))/(15*b),
(D*(4*v + 1))/(5*b) + (2*D*b)/a^2,
-D*v,   (4*D*b)/(3*a) - (4*D*a*(v - 1))/(15*b),
(D*(v - 1))/(5*b) - (2*D*b)/a^2,
0,     (2*D*b)/(3*a) + (D*a*(v - 1))/(15*b);
      -(2*D*(10*a^4 - 5*b^4 + 7*a^2*b^2 - 2*a^2*b^2*v))/(5*a^3*b^3),
(D*(v - 1))/(5*a) - (2*D*a)/b^2,           (D*(4*v + 1))/(5*b) -
(D*b)/a^2,  -(2*D*(5*a^4 + 5*b^4 - 7*a^2*b^2 +
2*a^2*b^2*v))/(5*a^3*b^3),          - (D*(v - 1))/(5*a) - (D*a)/b^2,
- (D*(v - 1))/(5*b) - (D*b)/a^2,  (2*D*(5*a^4 - 10*b^4 - 7*a^2*b^2 +
2*a^2*b^2*v))/(5*a^3*b^3),          (D*(4*v + 1))/(5*a) - (D*a)/b^2,
(D*(v - 1))/(5*b) - (2*D*b)/a^2, (2*D*(10*a^4 + 10*b^4 + 7*a^2*b^2 -
2*a^2*b^2*v))/(5*a^3*b^3),      - (D*(4*v + 1))/(5*a) - (2*D*a)/b^2,
- (D*(4*v + 1))/(5*b) - (2*D*b)/a^2;
                                     (2*D*a)/b^2 - (D*(v - 1))/(5*a),
(2*D*a)/(3*b) + (D*b*(v - 1))/(15*a),
0,                                (D*(v - 1))/(5*a) + (D*a)/b^2,
(D*a)/(3*b) - (D*b*(v - 1))/(15*a),
```

```
0,                                  (D*(4*v + 1))/(5*a) - (D*a)/b^2,
(2*D*a)/(3*b) + (2*D*b*(2*v - 2))/(15*a),
0,                               - (D*(4*v + 1))/(5*a) - (2*D*a)/b^2,
(4*D*a)/(3*b) - (4*D*b*(v - 1))/(15*a),
D*v;
                                       (D*(4*v + 1))/(5*b) - (D*b)/a^2,
0, (2*D*b)/(3*a) + (2*D*a*(2*v - 2))/(15*b),
(D*(v - 1))/(5*b) + (D*b)/a^2,
0,       (D*b)/(3*a) - (D*a*(v - 1))/(15*b),
(2*D*b)/a^2 - (D*(v - 1))/(5*b),
0,     (2*D*b)/(3*a) + (D*a*(v - 1))/(15*b),
- (D*(4*v + 1))/(5*b) - (2*D*b)/a^2,
D*v,   (4*D*b)/(3*a) - (4*D*a*(v - 1))/(15*b)];

if nargin==4
    R1=eq*a*b/4;
    R2=eq*a*b^2/24;
    R3=eq*b*a^2/24;

    % Vetor de forças nodais equivalentes
    feq(:,1)=[R1  R2 -R3  R1  R2  R3  R1 -R2  R3  R1 -R2 -R3]';
    fe=feq;
end
Ke=Keq;
%--------------------------end-------------------------------
```

Função montagemrigidezglobal.m

```
function [K,f]=montagemrigidezglobal(Mtopologica,K,Ke,f,fe)

    [nel,n]=size(Mtopologica);
    % nel numero de elemento da estrutura
    % n numero da coluna da matriz topologica
    st=Mtopologica(:,2:n);
    % st submatriz topologica que contem só o grau de liberdade da
    % estrutura
    for i = 1:nel
      K(st(i,:),st(i,:)) = K(st(i,:),st(i,:))+Ke;
      if nargin==5
         f(st(i,:))=f(st(i,:))+fe;
      end
    end
%--------------------------end-----------------------------
```

Função resolvesistema.m

```
function [d,R]=resolvesistema(K,f,cc)

  if nargin==2 ;
     d=K\f ;
  elseif nargin==3;
     [nd,nd]=size(K);
% gdlt grau de liberdade total
     gdlt=[1:nd]';
%
     d=zeros(size(gdlt));
     R=zeros(size(gdlt));
% gdlp grau de liberdade preescrito
     gdlp=cc(:,1);
```

```
% dp deslocamento preescrito
     dp=cc(:,2);
% elimina do grau de liberdade total o grau de liberdade preescrito
     gdlt(gdlp)=[];
% dl deslocamento livre
     dl=K(gdlt,gdlt)\(f(gdlt)-K(gdlt,gdlp)*dp);
%
     d(gdlp)=dp;
     d(gdlt)=dl;
  end
     R=K*d-f;
```

Função submatriztopologica.m

```
function [st]=submatriztopologica(i,nnoel,nglno,conec,gle)
% Elimina a coluna do número do elemento da matriz topológica.

 l=0;

    for k=1:nnoel
        for j=1:nglno
            l=l+1;
            st(l)=gle(conec(i,k),j);
        end
    end
```

Arquivo Sim_MatRigidezACM_Kirch.m

```
% ================================================
% Cálculo simbólico da matriz de rigidez de um
% elemento de placa do tipo ACM segundo a teoria
% de placas de Kirchhoff
% ================================================
clc
clear all

%% Declara as variáveis simbólicas a serem utilizadas
syms a b D v x y;

%% Campo de deslocamentos
w = [1 x y x^2 x*y y^2 x^3 x^2*y x*y^2 y^3 x^3*y x*y^3];

% Matriz de deslocamentos generalizados
Desloc=[w;diff(w, y);-diff(w, x)];

%% Particularização do vetor de deslocamentos em cada nó do elemento
A=sym(zeros(12));
A( 1:3 ,:)=subs(Desloc, [x y], [0 0]);
A( 4:6 ,:)=subs(Desloc, [x y], [a 0]);
A( 7:9 ,:)=subs(Desloc, [x y], [a b]);
A(10:12,:)=subs(Desloc, [x y], [0 b]);

A
%% Forma matricial do vetor de deformações independentes
G=[diff(w, x, 2); diff(w, y, 2); 2*diff(diff(w, x), y)]

%% Matriz constitutiva generalidada à flexão
```

```
Dk=D*[ 1, v,      0     ;
       v, 1,      0     ;
       0, 0, (1 - v)/2];

%% Cálculo da matriz de rigidez. O resultado será impresso na janela
de comando
MATRIZ_RIGIDEZ_ACM_Kirch=...
int(int(transpose(inv(A))*transpose(G)*Dk*G*inv(A), x, 0, a), y, 0, b)

% Fim do arquivo
```

APÊNDICE B: CÓDIGO DA IMPLEMENTAÇÃO DA TEORIA DAS PLACAS DE REISSNER-MINDLIN (EM MATLAB)

Arquivo exemplo01q4.m

```
% Fonte: (Khennanne, 2013)

% Exemplo 1

% Este programa foi adaptado para utilizar um elemento de quatro nós,
% a partir de um código implementado para um elemento quadrilátero de
8
% nós.

% O programa calcula os esforços e os deslocamentos de uma placa de
% Reissner-Mindlin ENGASTADA nos quatro CANTOS com um carregamento
nodal
% vertical aplicado no centro da placa. Esta implementação discretiza
toda
% a placa.

%
% Make these variables global so they can be shared by other functions
%
clc
clear all
%
global nnd nel nne nodof eldof n ngpb ngps
global geom connec deeb dees nf load dim
%
format long g
%
% To cchange the size of the problem or change the elastic properties
% ALTER the PlateQ8_input_module.m
%
dim=2;
nne=4;
nodof=3;
eldof=nne*nodof;
%
% Plate_Q8_input_module
Length=36.; % Length of the in x-direction
Width=36.; % Width of the model in y-direction
NXE=16; % Number of rows in the x direction
NYE=16; % Number of rows in the y direction
dhx=Length/NXE; % Element size in the x direction
dhy=Width/NYE; % Element size in the y direction
X_origin=0. ; % x origin of the global coordinate system
Y_origin=0. ; % y origin of the global coordinate system
%
thick=0.25; % Thickness of plate
ngpb=2; % number of Gauss points bending
ngps=1; % number of Gauss points for shear
%
Q4_mesh % Generate the mesh
%
E=30.e+6; % Elastic modulus in kN/m2
vu=0.3; % Poisson's ratio
%
% Form the matrix of elastic properties
%
```

```
deeb=formdeeb(E,vu,thick); % Matrix of elastic properties for plate
bending
dees=formdees(E,vu,thick); % Matrix of elastic properties for plate
shear
%
% Boundary conditions
%
nf=ones(nnd, nodof); % Initialize the matrix nf to 1
%
for i=1:nnd
    if geom(i,:) == [0 0]
        nf(i,1)=0 ; % Restrain in direction w
        nf(i,2)=0 ; % Restrain rotation theta_x (around y)
        nf(i,3)=0 ; % Restrain rotation theta_y (around x)
    elseif geom(i,:) == [Length 0]
        nf(i,1)=0 ; % Restrain in direction w
        nf(i,2)=0 ; % Restrain rotation theta_x (around y)
        nf(i,3)=0 ; % Restrain rotation theta_y (around x)
    elseif geom(i,:) == [Length Width]
        nf(i,1)=0 ; % Restrain in direction w
        nf(i,2)=0 ; % Restrain rotation theta_x (around y)
        nf(i,3)=0 ; % Restrain rotation theta_y (around x)
    elseif geom(i,:) == [0 Width]
        nf(i,1)=0 ; % Restrain in direction w
        nf(i,2)=0 ; % Restrain rotation theta_x (around y)
        nf(i,3)=0 ; % Restrain rotation theta_y (around x)
    end
end
%
% Counting of the free degrees of freedom
%
n=0;
for i=1:nnd
    for j=1:nodof
        if nf(i,j) ~=0
            n=n+1;
            nf(i,j)=n;
        end
    end
end
disp ('Total number of active degrees of freedom')
n
%
% loading
%
load=zeros(nnd, 3);
%
for i=1:nnd
    if geom(i,1) == Length/2 && geom(i,2) == Width/2
        load(i,1)=-1000; % Vertical load of 1000 lb on the center node
    end
end
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%% End of input%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Assemble the global force vector
%
fg=zeros(n,1);
for i=1: nnd
    for j=1:nodof
        if nf(i,j) ~=0
```

```
            fg(nf(i,j))=load(i,j);
        end
    end
end
%
% Form the matrix containing the abscissas and the weights of Gauss
points
%
sampb=gauss(ngpb);
samps=gauss(ngps);
%
% Numerical integration and assembly of the global stiffness matrix
%
% initialize the global stiffness matrix to zero
kk=sparse(n, n);
%
for i=1:nel
    [coord,g]=platelem_q4(i) ; % coordinates of the nodes of element
i,
                               % and its steering vector
    keb=zeros(eldof,eldof) ; % Initialize the element bending
                             % stiffness matrix to zero
    %
    % Integrate element bending stiffness and assemble it in global
matrix
    %
    for ig=1: ngpb
        wi=sampb(ig,2);
        for jg=1: ngpb
            wj=sampb(jg,2);
            [der,fun]=fmlin(sampb, ig,jg); % Derivative of shape
functions
                                           % in local coordinates
            jac=der*coord; % Compute Jacobian matrix
            d=det(jac); % Compute the determinant of
                        % Jacobian matrix
            jac1=inv(jac); % Compute inverse of the Jacobian
            deriv=jac1*der; % Derivative of shape functions
                            % in global coordinates
            beeb=formbeeb(deriv,nne,eldof); % Form matrix [Bb]
            bees=formbees(deriv,fun,nne,eldof); % Form matrix [Bs]
            keb=keb + d*wi*wj*beeb'*deeb*beeb + ...
                (5/6)*d*wi*wj*bees'*dees*bees;
        end
    end
    kk=form_KK(kk,keb, g); % assemble global stiffness matrix
end
%
%
%%%%%%%%%%%%%%%%%%%%%% End of assembly
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
delta=kk\fg; % solve for unknown displacements
format short e
disp('node w_disp x_slope y_slope ') %
for i=1: nnd %
    if nf(i,1) == 0 %
        w_disp=0.; %
    else
        w_disp=delta(nf(i,1)); %
    end
    %
```

```
    if nf(i,2) == 0 %
        x_slope=0.; %
    else
        x_slope=delta(nf(i,2)); %
    end
    %
    if nf(i,3) == 0 %
        y_slope=0.; %
    else
        y_slope=delta(nf(i,3)); %
    end
    disp([i w_disp x_slope y_slope]) % Display displacements of each
node
    DISP(i,:)=[ w_disp x_slope y_slope];
end
%
%
%
ngp=1; % Calculate moments and shear forces
       % the center of each element
samp=gauss(ngp);
%
for i=1:nel
    [coord,g]=platelem_q4(i); % coordinates of the nodes of element i,
                              % and its steering vector
    eld=zeros(eldof,1); % Initialize element displacement to zero
    for m=1:eldof %
        if g(m) == 0 %
            eld(m)=0.; %
        else %
            eld(m)=delta(g(m)); % Retrieve element displacement from
the
                                % global displacement vector
        end
    end
    %
    for ig=1: ngp
        wi=samp(ig,2);
        for jg=1: ngp
            wj=samp(jg,2);
            [der,fun]=fmlin(samp, ig,jg); % Derivative of shape
functions
                                          % in local coordinates
            jac=der*coord; % Compute Jacobian matrix
            d=det(jac); % Compute the determinant of
                        % Jacobian matrix
            jac1=inv(jac); % Compute inverse of the Jacobian
            deriv=jac1*der; % Derivative of shape functions
                            % in global coordinates

            beeb=formbeeb(deriv,nne,eldof); % Form matrix [B_b]
            chi_b=beeb*eld ; % compute bending curvatures
            Moment=deeb*chi_b ; % Compute moments
            bees=formbees(deriv,fun,nne,eldof); % Form matrix [B_s]
            chi_s=bees*eld ; % compute shear curvatures
            Shear=dees*chi_s ; % Compute shera forces
        end
    end
    Element_Forces(i,:)=[Moment' Shear'];
end
%
W=DISP(:,1);
```

```
[MX, MY, MXY, QX, QY]=Forces_at_nodes_plate(Element_Forces);
%
deslMax=min(W)
MxMax=max(MX)
MxMin=min(MX)

MyMax=max(MY)
MyMin=min(MY)

MxyMax=max(MXY)
MxyMin=min(MXY)

VxMax=max(QX)
VxMin=min(QX)

VyMax=max(QY)
VyMin=min(QY)

figure(1)
title('deslocamento')
cmin=min(W);
cmax=max(W);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',W,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(2)
title('Mx')
cmin=min(MX);
cmax=max(MX);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MX,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(3)
title('My')
cmin=min(MY);
cmax=max(MY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MY,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(4)
title('Mxy')
cmin=min(MXY);
cmax=max(MXY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MXY,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(5)
title('Vx')
cmin=min(QX);
cmax=max(QX);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',QX,...
      'Facecolor','interp','Marker','.');
```

```
colorbar;

figure(6)
title('Vy')
cmin=min(QY);
cmax=max(QY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',QY,...
      'Facecolor','interp','Marker','.');
colorbar;
```

Arquivo exemplo02q4.m

```
% Fonte: (Khennanne, 2013)

% Exemplo 2

% Este programa foi adaptado para utilizar um elemento de quatro nós,
% a partir de um código implementado para um elemento quadrilátero de
8
% nós.

% O programa calcula os esforços e os deslocamentos de uma placa de
% Reissner-Mindlin APOIADA nos quatro CANTOS com um carregamento nodal
% vertical aplicado no centro da placa. Esta implementação discretiza
toda
% a placa.

%
% Make these variables global so they can be shared by other functions
%
clc
clear all
%
global nnd nel nne nodof eldof n ngpb ngps
global geom connec deeb dees nf load dim
%
format long g
%
% To cchange the size of the problem or change the elastic properties
% ALTER the PlateQ8_input_module.m
%
dim=2;
nne=4;
nodof=3;
eldof=nne*nodof;
%
% Plate_Q8_input_module
Length=36.; % Length of the in x-direction
Width=36.; % Width of the model in y-direction
NXE=16; % Number of rows in the x direction
NYE=16; % Number of rows in the y direction
dhx=Length/NXE; % Element size in the x direction
dhy=Width/NYE; % Element size in the y direction
X_origin=0. ; % x origin of the global coordinate system
Y_origin=0. ; % y origin of the global coordinate system
%
thick=0.25; % Thickness of plate
ngpb=2; % number of Gauss points bending
ngps=1; % number of Gauss points for shear
%
```

```
Q4_mesh % Generate the mesh
%
E=30.e+6; % Elastic modulus in kN/m2
vu=0.3; % Poisson's ratio
%
% Form the matrix of elastic properties
%
deeb=formdeeb(E,vu,thick); % Matrix of elastic properties for plate
bending
dees=formdees(E,vu,thick); % Matrix of elastic properties for plate
shear
%
% Boundary conditions
%
nf=ones(nnd, nodof); % Initialize the matrix nf to 1
%
for i=1:nnd
    if geom(i,:) == [0 0]
        nf(i,1)=0 ; % Restrain in direction w
    elseif geom(i,:) == [Length 0]
        nf(i,1)=0 ; % Restrain in direction w
    elseif geom(i,:) == [Length Width]
        nf(i,1)=0 ; % Restrain in direction w
    elseif geom(i,:) == [0 Width]
        nf(i,1)=0 ; % Restrain in direction w
    end
end
%
% Counting of the free degrees of freedom
%
n=0;
for i=1:nnd
    for j=1:nodof
        if nf(i,j) ~=0
            n=n+1;
            nf(i,j)=n;
        end
    end
end
disp ('Total number of active degrees of freedom')
n
%
% loading
%
load=zeros(nnd, 3);
%
for i=1:nnd
    if geom(i,1) == Length/2 && geom(i,2) == Width/2
        load(i,1)=-1000; % Vertical load of 1000 lb on the center node
    end
end
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%% End of input%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Assemble the global force vector
%
fg=zeros(n,1);
for i=1: nnd
    for j=1:nodof
        if nf(i,j) ~=0
            fg(nf(i,j))=load(i,j);
```

```
        end
    end
end
%
% Form the matrix containing the abscissas and the weights of Gauss
points
%
sampb=gauss(ngpb);
samps=gauss(ngps);
%
% Numerical integration and assembly of the global stiffness matrix
%
% initialize the global stiffness matrix to zero
kk=sparse(n, n);
%
for i=1:nel
    [coord,g]=platelem_q4(i) ; % coordinates of the nodes of element
i,
                               % and its steering vector
    keb=zeros(eldof,eldof) ; % Initialize the element bending
                             % stiffness matrix to zero
    %
    % Integrate element bending stiffness and assemble it in global
matrix
    %
    for ig=1: ngpb
        wi=sampb(ig,2);
        for jg=1: ngpb
            wj=sampb(jg,2);
            [der,fun]=fmlin(sampb, ig,jg); % Derivative of shape
functions
                                           % in local coordinates
            jac=der*coord; % Compute Jacobian matrix
            d=det(jac); % Compute the determinant of
                        % Jacobian matrix
            jac1=inv(jac); % Compute inverse of the Jacobian
            deriv=jac1*der; % Derivative of shape functions
                            % in global coordinates
            beeb=formbeeb(deriv,nne,eldof); % Form matrix [Bb]
            bees=formbees(deriv,fun,nne,eldof); % Form matrix [Bs]
            keb=keb + d*wi*wj*beeb'*deeb*beeb + ...
                (5/6)*d*wi*wj*bees'*dees*bees;
        end
    end
    kk=form_KK(kk,keb, g); % assemble global stiffness matrix
end
%
%
%%%%%%%%%%%%%%%%%%%%%%% End of assembly
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
delta=kk\fg; % solve for unknown displacements
format short e
disp('node w_disp x_slope y_slope ') %
for i=1: nnd %
    if nf(i,1) == 0 %
        w_disp=0.; %
    else
        w_disp=delta(nf(i,1)); %
    end
    %
    if nf(i,2) == 0 %
```

```
        x_slope=0.; %
    else
        x_slope=delta(nf(i,2)); %
    end
    %
    if nf(i,3) == 0 %
        y_slope=0.; %
    else
        y_slope=delta(nf(i,3)); %
    end
    disp([i w_disp x_slope y_slope]) % Display displacements of each
node
    DISP(i,:)=[ w_disp x_slope y_slope];
end
%
%
%
ngp=1; % Calculate moments and shear forces
       % the center of each element
samp=gauss(ngp);
%
for i=1:nel
    [coord,g]=platelem_q4(i); % coordinates of the nodes of element i,
                              % and its steering vector
    eld=zeros(eldof,1); % Initialize element displacement to zero
    for m=1:eldof %
        if g(m) == 0 %
            eld(m)=0.; %
        else %
            eld(m)=delta(g(m)); % Retrieve element displacement from
the
                                % global displacement vector
        end
    end
    %
    for ig=1: ngp
        wi=samp(ig,2);
        for jg=1: ngp
            wj=samp(jg,2);
            [der,fun]=fmlin(samp, ig,jg); % Derivative of shape
functions
                                          % in local coordinates
            jac=der*coord; % Compute Jacobian matrix
            d=det(jac); % Compute the determinant of
                        % Jacobian matrix
            jac1=inv(jac); % Compute inverse of the Jacobian
            deriv=jac1*der; % Derivative of shape functions
                            % in global coordinates

            beeb=formbeeb(deriv,nne,eldof); % Form matrix [B_b]
            chi_b=beeb*eld ; % compute bending curvatures
            Moment=deeb*chi_b ; % Compute moments
            bees=formbees(deriv,fun,nne,eldof); % Form matrix [B_s]
            chi_s=bees*eld ; % compute shear curvatures
            Shear=dees*chi_s ; % Compute shera forces
        end
    end
    Element_Forces(i,:)=[Moment' Shear'];
end
%
W=DISP(:,1);
[MX, MY, MXY, QX, QY]=Forces_at_nodes_plate(Element_Forces);
```

```
%
deslMax=min(W)
MxMax=max(MX)
MxMin=min(MX)

MyMax=max(MY)
MyMin=min(MY)

MxyMax=max(MXY)
MxyMin=min(MXY)

VxMax=max(QX)
VxMin=min(QX)

VyMax=max(QY)
VyMin=min(QY)

figure(1)
title('deslocamento')
cmin=min(W);
cmax=max(W);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',W,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(2)
title('Mx')
cmin=min(MX);
cmax=max(MX);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MX,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(3)
title('My')
cmin=min(MY);
cmax=max(MY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MY,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(4)
title('Mxy')
cmin=min(MXY);
cmax=max(MXY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MXY,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(5)
title('Vx')
cmin=min(QX);
cmax=max(QX);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',QX,...
      'Facecolor','interp','Marker','.');
colorbar;
```

```
figure(6)
title('Vy')
cmin=min(QY);
cmax=max(QY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',QY,...
      'Facecolor','interp','Marker','.');
colorbar;
```

Arquivo exemplo03q4.m

```
% Fonte: (Khennanne, 2013)

% Exemplo 3

% Este programa foi adaptado para utilizar um elemento de quatro nós,
% a partir de um código implementado para um elemento quadrilátero de
8
% nós.

% O programa calcula os esforços e os deslocamentos de uma placa de
% Reissner-Mindlin ENGASTADA nos quatro LADOS com um carregamento
nodal
% vertical aplicado no centro da placa. Esta implementação discretiza
toda
% a placa.

%
% Make these variables global so they can be shared by other functions
%
clc
clear all
%
global nnd nel nne nodof eldof n ngpb ngps
global geom connec deeb dees nf load dim
%
format long g
%
% To cchange the size of the problem or change the elastic properties
% ALTER the PlateQ8_input_module.m
%
dim=2;
nne=4;
nodof=3;
eldof=nne*nodof;
%
% Plate_Q8_input_module
Length=36.; % Length of the in x-direction
Width=36.; % Width of the model in y-direction
NXE=16; % Number of rows in the x direction
NYE=16; % Number of rows in the y direction
dhx=Length/NXE; % Element size in the x direction
dhy=Width/NYE; % Element size in the y direction
X_origin=0. ; % x origin of the global coordinate system
Y_origin=0. ; % y origin of the global coordinate system
%
thick=0.25; % Thickness of plate
ngpb=2; % number of Gauss points bending
ngps=1; % number of Gauss points for shear
%
```

```
Q4_mesh % Generate the mesh
%
E=30.e+6; % Elastic modulus in kN/m2
vu=0.3; % Poisson's ratio
%
% Form the matrix of elastic properties
%
deeb=formdeeb(E,vu,thick); % Matrix of elastic properties for plate
bending
dees=formdees(E,vu,thick); % Matrix of elastic properties for plate
shear
%
% Boundary conditions
%
nf=ones(nnd, nodof); % Initialize the matrix nf to 1
%
for i=1:nnd
    if geom(i,1) == 0
        nf(i,1)=0 ; % Restrain in direction w
        nf(i,3)=0 ; % Restrain rotation theta_y (around x)
    elseif geom(i,2) == 0
        nf(i,1)=0. ; % Restrain displacement w
        nf(i,2)=0. ; % Restrain rotation theta_x (around y)
    elseif geom(i,1) == Length
        nf(i,1)=0 ; % Restrain in direction w
        nf(i,3)=0. ; % Restrain rotation theta_y (around x)
    elseif geom(i,2) == Width
        nf(i,1)=0 ; % Restrain in direction w
        nf(i,2)=0. ; % Restrain rotation theta_x (around y)
    end
end
%
% Counting of the free degrees of freedom
%
n=0;
for i=1:nnd
    for j=1:nodof
        if nf(i,j) ~=0
            n=n+1;
            nf(i,j)=n;
        end
    end
end
disp ('Total number of active degrees of freedom')
n
%
% loading
%
load=zeros(nnd, 3);
%
for i=1:nnd
    if geom(i,1) == Length/2 && geom(i,2) == Width/2
        load(i,1)=-1000; % Vertical load of 1000 lb on the center node
    end
end
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%% End of input%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Assemble the global force vector
%
fg=zeros(n,1);
```

```
for i=1: nnd
    for j=1:nodof
        if nf(i,j) ~=0
            fg(nf(i,j))=load(i,j);
        end
    end
end
%
% Form the matrix containing the abscissas and the weights of Gauss
points
%
sampb=gauss(ngpb);
samps=gauss(ngps);
%
% Numerical integration and assembly of the global stiffness matrix
%
% initialize the global stiffness matrix to zero
kk=sparse(n, n);
%
for i=1:nel
    [coord,g]=platelem_q4(i) ; % coordinates of the nodes of element
i,
                               % and its steering vector
    keb=zeros(eldof,eldof) ; % Initialize the element bending
                             % stiffness matrix to zero
    %
    % Integrate element bending stiffness and assemble it in global
matrix
    %
    for ig=1: ngpb
        wi=sampb(ig,2);
        for jg=1: ngpb
            wj=sampb(jg,2);
            [der,fun]=fmlin(sampb, ig,jg); % Derivative of shape
functions
                                           % in local coordinates
            jac=der*coord; % Compute Jacobian matrix
            d=det(jac); % Compute the determinant of
                        % Jacobian matrix
            jac1=inv(jac); % Compute inverse of the Jacobian
            deriv=jac1*der; % Derivative of shape functions
                            % in global coordinates
            beeb=formbeeb(deriv,nne,eldof); % Form matrix [Bb]
            bees=formbees(deriv,fun,nne,eldof); % Form matrix [Bs]
            keb=keb + d*wi*wj*beeb'*deeb*beeb + ...
                (5/6)*d*wi*wj*bees'*dees*bees;
        end
    end
    kk=form_KK(kk,keb, g); % assemble global stiffness matrix
end
%
%
%%%%%%%%%%%%%%%%%%%%%% End of assembly
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
delta=kk\fg; % solve for unknown displacements
format short e
disp('node w_disp x_slope y_slope ') %
for i=1: nnd %
    if nf(i,1) == 0 %
        w_disp=0.; %
    else
```

```
        w_disp=delta(nf(i,1)); %
    end
    %
    if nf(i,2) == 0 %
        x_slope=0.; %
    else
        x_slope=delta(nf(i,2)); %
    end
    %
    if nf(i,3) == 0 %
        y_slope=0.; %
    else
        y_slope=delta(nf(i,3)); %
    end
    disp([i w_disp x_slope y_slope]) % Display displacements of each
node
    DISP(i,:)=[ w_disp x_slope y_slope];
end
%
%
%
ngp=1; % Calculate moments and shear forces
       % the center of each element
samp=gauss(ngp);
%
for i=1:nel
    [coord,g]=platelem_q4(i); % coordinates of the nodes of element i,
                              % and its steering vector
    eld=zeros(eldof,1); % Initialize element displacement to zero
    for m=1:eldof %
        if g(m) == 0 %
            eld(m)=0.; %
        else %
            eld(m)=delta(g(m)); % Retrieve element displacement from
the
                                % global displacement vector
        end
    end
    %
    for ig=1: ngp
        wi=samp(ig,2);
        for jg=1: ngp
            wj=samp(jg,2);
            [der,fun]=fmlin(samp, ig,jg); % Derivative of shape
functions
                                          % in local coordinates
            jac=der*coord; % Compute Jacobian matrix
            d=det(jac); % Compute the determinant of
                        % Jacobian matrix
            jac1=inv(jac); % Compute inverse of the Jacobian
            deriv=jac1*der; % Derivative of shape functions
                            % in global coordinates

            beeb=formbeeb(deriv,nne,eldof); % Form matrix [B_b]
            chi_b=beeb*eld ; % compute bending curvatures
            Moment=deeb*chi_b ; % Compute moments
            bees=formbees(deriv,fun,nne,eldof); % Form matrix [B_s]
            chi_s=bees*eld ; % compute shear curvatures
            Shear=dees*chi_s ; % Compute shera forces
        end
    end
    Element_Forces(i,:)=[Moment' Shear'];
```

```
end
%
W=DISP(:,1);
[MX, MY, MXY, QX, QY]=Forces_at_nodes_plate(Element_Forces);
%
deslMax=min(W)
MxMax=max(MX)
MxMin=min(MX)

MyMax=max(MY)
MyMin=min(MY)

MxyMax=max(MXY)
MxyMin=min(MXY)

VxMax=max(QX)
VxMin=min(QX)

VyMax=max(QY)
VyMin=min(QY)

figure(1)
title('deslocamento')
cmin=min(W);
cmax=max(W);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',W,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(2)
title('Mx')
cmin=min(MX);
cmax=max(MX);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MX,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(3)
title('My')
cmin=min(MY);
cmax=max(MY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MY,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(4)
title('Mxy')
cmin=min(MXY);
cmax=max(MXY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MXY,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(5)
title('Vx')
cmin=min(QX);
cmax=max(QX);
```

```
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',QX,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(6)
title('Vy')
cmin=min(QY);
cmax=max(QY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',QY,...
      'Facecolor','interp','Marker','.');
colorbar;
```

Arquivo elemento04q4.m

```
% Fonte: (Khennanne, 2013)

% Exemplo 4

% Este programa foi adaptado para utilizar um elemento de quatro nós,
% a partir de um código implementado para um elemento quadrilátero de
8
% nós.

% O programa calcula os esforços e os deslocamentos de uma placa de
% Reissner-Mindlin APOIADA nos quatro LADOS com um carregamento nodal
% vertical aplicado no centro da placa. Esta implementação discretiza
toda
% a placa.

%
% Make these variables global so they can be shared by other functions
%
clc
clear all
%
global nnd nel nne nodof eldof n ngpb ngps
global geom connec deeb dees nf load dim
%
format long g
%
% To cchange the size of the problem or change the elastic properties
% ALTER the PlateQ8_input_module.m
%
dim=2;
nne=4;
nodof=3;
eldof=nne*nodof;
%
% Plate_Q8_input_module
Length=36.; % Length of the in x-direction
Width=36.; % Width of the model in y-direction
NXE=16; % Number of rows in the x direction
NYE=16; % Number of rows in the y direction
dhx=Length/NXE; % Element size in the x direction
dhy=Width/NYE; % Element size in the y direction
X_origin=0. ; % x origin of the global coordinate system
Y_origin=0. ; % y origin of the global coordinate system
%
```

```
thick=0.25; % Thickness of plate
ngpb=2; % number of Gauss points bending
ngps=1; % number of Gauss points for shear
%
Q4_mesh % Generate the mesh
%
E=30.e+6; % Elastic modulus in kN/m2
vu=0.3; % Poisson's ratio
%
% Form the matrix of elastic properties
%
deeb=formdeeb(E,vu,thick); % Matrix of elastic properties for plate
bending
dees=formdees(E,vu,thick); % Matrix of elastic properties for plate
shear
%
% Boundary conditions
%
nf=ones(nnd, nodof); % Initialize the matrix nf to 1
%
for i=1:nnd
    if geom(i,1) == 0
        nf(i,1)=0 ; % Restrain in direction w
    elseif geom(i,2) == 0
        nf(i,1)=0. ; % Restrain displacement w
    elseif geom(i,1) == Length
        nf(i,1)=0 ; % Restrain in direction w
    elseif geom(i,2) == Width
        nf(i,1)=0 ; % Restrain in direction w
    end
end
%
% Counting of the free degrees of freedom
%
n=0;
for i=1:nnd
    for j=1:nodof
        if nf(i,j) ~=0
            n=n+1;
            nf(i,j)=n;
        end
    end
end
disp ('Total number of active degrees of freedom')
n
%
% loading
%
load=zeros(nnd, 3);
%
for i=1:nnd
    if geom(i,1) == Length/2 && geom(i,2) == Width/2
        load(i,1)=-1000; % Vertical load of 1000 lb on the center node
    end
end
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%% End of input%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Assemble the global force vector
%
fg=zeros(n,1);
```

```
for i=1: nnd
    for j=1:nodof
        if nf(i,j) ~=0
            fg(nf(i,j))=load(i,j);
        end
    end
end
%
% Form the matrix containing the abscissas and the weights of Gauss
points
%
sampb=gauss(ngpb);
samps=gauss(ngps);
%
% Numerical integration and assembly of the global stiffness matrix
%
% initialize the global stiffness matrix to zero
kk=sparse(n, n);
%
for i=1:nel
    [coord,g]=platelem_q4(i) ; % coordinates of the nodes of element
i,
                               % and its steering vector
    keb=zeros(eldof,eldof) ; % Initialize the element bending
                             % stiffness matrix to zero
    %
    % Integrate element bending stiffness and assemble it in global
matrix
    %
    for ig=1: ngpb
        wi=sampb(ig,2);
        for jg=1: ngpb
            wj=sampb(jg,2);
            [der,fun]=fmlin(sampb, ig,jg); % Derivative of shape
functions
                                           % in local coordinates
            jac=der*coord; % Compute Jacobian matrix
            d=det(jac); % Compute the determinant of
                        % Jacobian matrix
            jac1=inv(jac); % Compute inverse of the Jacobian
            deriv=jac1*der; % Derivative of shape functions
                            % in global coordinates
            beeb=formbeeb(deriv,nne,eldof); % Form matrix [Bb]
            bees=formbees(deriv,fun,nne,eldof); % Form matrix [Bs]
            keb=keb + d*wi*wj*beeb'*deeb*beeb + ...
                (5/6)*d*wi*wj*bees'*dees*bees;
        end
    end
    kk=form_KK(kk,keb, g); % assemble global stiffness matrix
end
%
%
%%%%%%%%%%%%%%%%%%%%%% End of assembly
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
delta=kk\fg; % solve for unknown displacements
format short e
disp('node w_disp x_slope y_slope ') %
for i=1: nnd %
    if nf(i,1) == 0 %
        w_disp=0.; %
    else
```

```
        w_disp=delta(nf(i,1)); %
    end
    %
    if nf(i,2) == 0 %
        x_slope=0.; %
    else
        x_slope=delta(nf(i,2)); %
    end
    %
    if nf(i,3) == 0 %
        y_slope=0.; %
    else
        y_slope=delta(nf(i,3)); %
    end
    disp([i w_disp x_slope y_slope]) % Display displacements of each
node
    DISP(i,:)=[ w_disp x_slope y_slope];
end
%
%
%
ngp=1; % Calculate moments and shear forces
       % the center of each element
samp=gauss(ngp);
%
for i=1:nel
    [coord,g]=platelem_q4(i); % coordinates of the nodes of element i,
                              % and its steering vector
    eld=zeros(eldof,1); % Initialize element displacement to zero
    for m=1:eldof %
        if g(m) == 0 %
            eld(m)=0.; %
        else %
            eld(m)=delta(g(m)); % Retrieve element displacement from
the
                                % global displacement vector
        end
    end
    %
    for ig=1: ngp
        wi=samp(ig,2);
        for jg=1: ngp
            wj=samp(jg,2);
            [der,fun]=fmlin(samp, ig,jg); % Derivative of shape
functions
                                          % in local coordinates
            jac=der*coord; % Compute Jacobian matrix
            d=det(jac); % Compute the determinant of
                        % Jacobian matrix
            jac1=inv(jac); % Compute inverse of the Jacobian
            deriv=jac1*der; % Derivative of shape functions
                            % in global coordinates

            beeb=formbeeb(deriv,nne,eldof); % Form matrix [B_b]
            chi_b=beeb*eld ; % compute bending curvatures
            Moment=deeb*chi_b ; % Compute moments
            bees=formbees(deriv,fun,nne,eldof); % Form matrix [B_s]
            chi_s=bees*eld ; % compute shear curvatures
            Shear=dees*chi_s ; % Compute shera forces
        end
    end
    Element_Forces(i,:)=[Moment' Shear'];
```

```
end
%
W=DISP(:,1);
[MX, MY, MXY, QX, QY]=Forces_at_nodes_plate(Element_Forces);
%
deslMax=min(W)
MxMax=max(MX)
MxMin=min(MX)

MyMax=max(MY)
MyMin=min(MY)

MxyMax=max(MXY)
MxyMin=min(MXY)

VxMax=max(QX)
VxMin=min(QX)

VyMax=max(QY)
VyMin=min(QY)

figure(1)
title('deslocamento')
cmin=min(W);
cmax=max(W);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',W,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(2)
title('Mx')
cmin=min(MX);
cmax=max(MX);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MX,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(3)
title('My')
cmin=min(MY);
cmax=max(MY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MY,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(4)
title('Mxy')
cmin=min(MXY);
cmax=max(MXY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MXY,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(5)
title('Vx')
cmin=min(QX);
cmax=max(QX);
```

```
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',QX,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(6)
title('Vy')
cmin=min(QY);
cmax=max(QY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',QY,...
      'Facecolor','interp','Marker','.');
colorbar;
```

Arquivo exemplo05q4.m

```
% Fonte: (Khennanne, 2013)

% Exemplo 5

% Este programa foi adaptado para utilizar um elemento de quatro nós,
% a partir de um código implementado para um elemento quadrilátero de
8
% nós.

% O programa calcula os esforços e os deslocamentos de uma placa de
% Reissner-Mindlin ENGASTADA EM DOIS LADOS OPOSTOS E LIVRE NOS OUTROS
DOIS
% com um carregamento nodal vertical aplicado no centro da placa.
% Esta implementação discretiza toda a placa.

%
% Make these variables global so they can be shared by other functions
%
clc
clear all
%
global nnd nel nne nodof eldof n ngpb ngps
global geom connec deeb dees nf load dim
%
format long g
%
% To cchange the size of the problem or change the elastic properties
% ALTER the PlateQ8_input_module.m
%
dim=2;
nne=4;
nodof=3;
eldof=nne*nodof;
%
% Plate_Q8_input_module
Length=36.; % Length of the in x-direction
Width=36.; % Width of the model in y-direction
NXE=16; % Number of rows in the x direction
NYE=16; % Number of rows in the y direction
dhx=Length/NXE; % Element size in the x direction
dhy=Width/NYE; % Element size in the y direction
X_origin=0. ; % x origin of the global coordinate system
Y_origin=0. ; % y origin of the global coordinate system
%
thick=0.25; % Thickness of plate
```

```
ngpb=2; % number of Gauss points bending
ngps=1; % number of Gauss points for shear
%
Q4_mesh % Generate the mesh
%
E=30.e+6; % Elastic modulus in kN/m2
vu=0.3; % Poisson's ratio
%
% Form the matrix of elastic properties
%
deeb=formdeeb(E,vu,thick); % Matrix of elastic properties for plate
bending
dees=formdees(E,vu,thick); % Matrix of elastic properties for plate
shear
%
% Boundary conditions
%
nf=ones(nnd, nodof); % Initialize the matrix nf to 1
%
for i=1:nnd
    if geom(i,1) == 0
        nf(i,1)=0 ; % Restrain in direction w
        nf(i,3)=0 ; % Restrain rotation theta_y (around x)
    elseif geom(i,1) == Length
        nf(i,1)=0 ; % Restrain in direction w
        nf(i,3)=0. ; % Restrain rotation theta_y (around x)
    end
end
%
% Counting of the free degrees of freedom
%
n=0;
for i=1:nnd
    for j=1:nodof
        if nf(i,j) ~=0
            n=n+1;
            nf(i,j)=n;
        end
    end
end
disp ('Total number of active degrees of freedom')
n
%
% loading
%
load=zeros(nnd, 3);
%
for i=1:nnd
    if geom(i,1) == Length/2 && geom(i,2) == Width/2
        load(i,1)=-1000; % Vertical load of 1000 lb on the center node
    end
end
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%% End of input%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Assemble the global force vector
%
fg=zeros(n,1);
for i=1: nnd
    for j=1:nodof
        if nf(i,j) ~=0
```

```
            fg(nf(i,j))=load(i,j);
        end
    end
end
%
% Form the matrix containing the abscissas and the weights of Gauss
points
%
sampb=gauss(ngpb);
samps=gauss(ngps);
%
% Numerical integration and assembly of the global stiffness matrix
%
% initialize the global stiffness matrix to zero
kk=sparse(n, n);
%
for i=1:nel
    [coord,g]=platelem_q4(i) ; % coordinates of the nodes of element
i,
                               % and its steering vector
    keb=zeros(eldof,eldof) ; % Initialize the element bending
                             % stiffness matrix to zero
    %
    % Integrate element bending stiffness and assemble it in global
matrix
    %
    for ig=1: ngpb
        wi=sampb(ig,2);
        for jg=1: ngpb
            wj=sampb(jg,2);
            [der,fun]=fmlin(sampb, ig,jg); % Derivative of shape
functions
                                           % in local coordinates
            jac=der*coord; % Compute Jacobian matrix
            d=det(jac); % Compute the determinant of
                        % Jacobian matrix
            jac1=inv(jac); % Compute inverse of the Jacobian
            deriv=jac1*der; % Derivative of shape functions
                            % in global coordinates
            beeb=formbeeb(deriv,nne,eldof); % Form matrix [Bb]
            bees=formbees(deriv,fun,nne,eldof); % Form matrix [Bs]
            keb=keb + d*wi*wj*beeb'*deeb*beeb + ...
                (5/6)*d*wi*wj*bees'*dees*bees;
        end
    end
    kk=form_KK(kk,keb, g); % assemble global stiffness matrix
end
%
%
%%%%%%%%%%%%%%%%%%%%%% End of assembly
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
delta=kk\fg; % solve for unknown displacements
format short e
disp('node w_disp x_slope y_slope ') %
for i=1: nnd %
    if nf(i,1) == 0 %
        w_disp=0.; %
    else
        w_disp=delta(nf(i,1)); %
    end
    %
```

```
    if nf(i,2) == 0 %
        x_slope=0.; %
    else
        x_slope=delta(nf(i,2)); %
    end
    %
    if nf(i,3) == 0 %
        y_slope=0.; %
    else
        y_slope=delta(nf(i,3)); %
    end
    disp([i w_disp x_slope y_slope]) % Display displacements of each
node
    DISP(i,:)=[ w_disp x_slope y_slope];
end
%
%
%
ngp=1; % Calculate moments and shear forces
       % the center of each element
samp=gauss(ngp);
%
for i=1:nel
    [coord,g]=platelem_q4(i); % coordinates of the nodes of element i,
                              % and its steering vector
    eld=zeros(eldof,1); % Initialize element displacement to zero
    for m=1:eldof %
        if g(m) == 0 %
            eld(m)=0.; %
        else %
            eld(m)=delta(g(m)); % Retrieve element displacement from
the
                                % global displacement vector
        end
    end
    %
    for ig=1: ngp
        wi=samp(ig,2);
        for jg=1: ngp
            wj=samp(jg,2);
            [der,fun]=fmlin(samp, ig,jg); % Derivative of shape
functions
                                          % in local coordinates
            jac=der*coord; % Compute Jacobian matrix
            d=det(jac); % Compute the determinant of
                        % Jacobian matrix
            jac1=inv(jac); % Compute inverse of the Jacobian
            deriv=jac1*der; % Derivative of shape functions
                            % in global coordinates

            beeb=formbeeb(deriv,nne,eldof); % Form matrix [B_b]
            chi_b=beeb*eld ; % compute bending curvatures
            Moment=deeb*chi_b ; % Compute moments
            bees=formbees(deriv,fun,nne,eldof); % Form matrix [B_s]
            chi_s=bees*eld ; % compute shear curvatures
            Shear=dees*chi_s ; % Compute shera forces
        end
    end
    Element_Forces(i,:)=[Moment' Shear'];
end
%
W=DISP(:,1);
```

```
[MX, MY, MXY, QX, QY]=Forces_at_nodes_plate(Element_Forces);
%
deslMax=min(W)
MxMax=max(MX)
MxMin=min(MX)

MyMax=max(MY)
MyMin=min(MY)

MxyMax=max(MXY)
MxyMin=min(MXY)

VxMax=max(QX)
VxMin=min(QX)

VyMax=max(QY)
VyMin=min(QY)

figure(1)
title('deslocamento')
cmin=min(W);
cmax=max(W);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',W,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(2)
title('Mx')
cmin=min(MX);
cmax=max(MX);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MX,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(3)
title('My')
cmin=min(MY);
cmax=max(MY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MY,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(4)
title('Mxy')
cmin=min(MXY);
cmax=max(MXY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MXY,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(5)
title('Vx')
cmin=min(QX);
cmax=max(QX);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',QX,...
      'Facecolor','interp','Marker','.');
```

```
colorbar;

figure(6)
title('Vy')
cmin=min(QY);
cmax=max(QY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',QY,...
      'Facecolor','interp','Marker','.');
colorbar;
```

Arquivo exemplo06q4.m

```
% Fonte: (Khennanne, 2013)

% Exemplo 6

% Este programa foi adaptado para utilizar um elemento de quatro nós,
% a partir de um código implementado para um elemento quadrilátero de
8
% nós.

% O programa calcula os esforços e os deslocamentos de uma placa de
% Reissner-Mindlin ENGASTADA EM DOIS LADOS ADJACENTES E LIVRE NOS
OUTROS DOIS
% com um carregamento nodal vertical aplicado no centro da placa.
% Esta implementação discretiza toda a placa.

%
% Make these variables global so they can be shared by other functions
%
clc
clear all
%
global nnd nel nne nodof eldof n ngpb ngps
global geom connec deeb dees nf load dim
%
format long g
%
% To cchange the size of the problem or change the elastic properties
% ALTER the PlateQ8_input_module.m
%
dim=2;
nne=4;
nodof=3;
eldof=nne*nodof;
%
% Plate_Q8_input_module
Length=36.; % Length of the in x-direction
Width=36.; % Width of the model in y-direction
NXE=16; % Number of rows in the x direction
NYE=16; % Number of rows in the y direction
dhx=Length/NXE; % Element size in the x direction
dhy=Width/NYE; % Element size in the y direction
X_origin=0. ; % x origin of the global coordinate system
Y_origin=0. ; % y origin of the global coordinate system
%
thick=0.25; % Thickness of plate
ngpb=2; % number of Gauss points bending
ngps=1; % number of Gauss points for shear
%
```

```
Q4_mesh % Generate the mesh
%
E=30.e+6; % Elastic modulus in kN/m2
vu=0.3; % Poisson's ratio
%
% Form the matrix of elastic properties
%
deeb=formdeeb(E,vu,thick); % Matrix of elastic properties for plate
bending
dees=formdees(E,vu,thick); % Matrix of elastic properties for plate
shear
%
% Boundary conditions
%
nf=ones(nnd, nodof); % Initialize the matrix nf to 1
%
for i=1:nnd
    if geom(i,1) == 0
        nf(i,1)=0 ; % Restrain in direction w
        nf(i,3)=0 ; % Restrain rotation theta_y (around x)
    elseif geom(i,2) == 0
        nf(i,1)=0 ; % Restrain in direction w
        nf(i,2)=0. ; % Restrain rotation theta_x (around y)
    end
end
%
% Counting of the free degrees of freedom
%
n=0;
for i=1:nnd
    for j=1:nodof
        if nf(i,j) ~=0
            n=n+1;
            nf(i,j)=n;
        end
    end
end
disp ('Total number of active degrees of freedom')
n
%
% loading
%
load=zeros(nnd, 3);
%
for i=1:nnd
    if geom(i,1) == Length/2 && geom(i,2) == Width/2
        load(i,1)=-1000; % Vertical load of 1000 kN on the center node
    end
end
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%% End of input%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Assemble the global force vector
%
fg=zeros(n,1);
for i=1: nnd
    for j=1:nodof
        if nf(i,j) ~=0
            fg(nf(i,j))=load(i,j);
        end
    end
```

```
end
%
% Form the matrix containing the abscissas and the weights of Gauss
points
%
sampb=gauss(ngpb);
samps=gauss(ngps);
%
% Numerical integration and assembly of the global stiffness matrix
%
% initialize the global stiffness matrix to zero
kk=sparse(n, n);
%
for i=1:nel
    [coord,g]=platelem_q4(i) ; % coordinates of the nodes of element
i,
                               % and its steering vector
    keb=zeros(eldof,eldof) ; % Initialize the element bending
                             % stiffness matrix to zero
    %
    % Integrate element bending stiffness and assemble it in global
matrix
    %
    for ig=1: ngpb
        wi=sampb(ig,2);
        for jg=1: ngpb
            wj=sampb(jg,2);
            [der,fun]=fmlin(sampb, ig,jg); % Derivative of shape
functions
                                           % in local coordinates
            jac=der*coord; % Compute Jacobian matrix
            d=det(jac); % Compute the determinant of
                        % Jacobian matrix
            jac1=inv(jac); % Compute inverse of the Jacobian
            deriv=jac1*der; % Derivative of shape functions
                            % in global coordinates
            beeb=formbeeb(deriv,nne,eldof); % Form matrix [Bb]
            bees=formbees(deriv,fun,nne,eldof); % Form matrix [Bs]
            keb=keb + d*wi*wj*beeb'*deeb*beeb + ...
                (5/6)*d*wi*wj*bees'*dees*bees;
        end
    end
    kk=form_KK(kk,keb, g); % assemble global stiffness matrix
end
%
%
%%%%%%%%%%%%%%%%%%%%%%% End of assembly
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
delta=kk\fg; % solve for unknown displacements
format short e
disp('node w_disp x_slope y_slope ') %
for i=1: nnd %
    if nf(i,1) == 0 %
        w_disp=0.; %
    else
        w_disp=delta(nf(i,1)); %
    end
    %
    if nf(i,2) == 0 %
        x_slope=0.; %
    else
```

```
        x_slope=delta(nf(i,2)); %
    end
    %
    if nf(i,3) == 0 %
        y_slope=0.; %
    else
        y_slope=delta(nf(i,3)); %
    end
    disp([i w_disp x_slope y_slope]) % Display displacements of each
node
    DISP(i,:)=[ w_disp x_slope y_slope];
end
%
%
%
ngp=1; % Calculate moments and shear forces
       % the center of each element
samp=gauss(ngp);
%
for i=1:nel
    [coord,g]=platelem_q4(i); % coordinates of the nodes of element i,
                              % and its steering vector
    eld=zeros(eldof,1); % Initialize element displacement to zero
    for m=1:eldof %
        if g(m) == 0 %
            eld(m)=0.; %
        else %
            eld(m)=delta(g(m)); % Retrieve element displacement from
the
                                % global displacement vector
        end
    end
    %
    for ig=1: ngp
        wi=samp(ig,2);
        for jg=1: ngp
            wj=samp(jg,2);
            [der,fun]=fmlin(samp, ig,jg); % Derivative of shape
functions
                                          % in local coordinates
            jac=der*coord; % Compute Jacobian matrix
            d=det(jac); % Compute the determinant of
                        % Jacobian matrix
            jac1=inv(jac); % Compute inverse of the Jacobian
            deriv=jac1*der; % Derivative of shape functions
                            % in global coordinates

            beeb=formbeeb(deriv,nne,eldof); % Form matrix [B_b]
            chi_b=beeb*eld ; % compute bending curvatures
            Moment=deeb*chi_b ; % Compute moments
            bees=formbees(deriv,fun,nne,eldof); % Form matrix [B_s]
            chi_s=bees*eld ; % compute shear curvatures
            Shear=dees*chi_s ; % Compute shera forces
        end
    end
    Element_Forces(i,:)=[Moment' Shear'];
end
%
W=DISP(:,1);
[MX, MY, MXY, QX, QY]=Forces_at_nodes_plate(Element_Forces);
%
deslMax=min(W)
```

```
MxMax=max(MX)
MxMin=min(MX)

MyMax=max(MY)
MyMin=min(MY)

MxyMax=max(MXY)
MxyMin=min(MXY)

VxMax=max(QX)
VxMin=min(QX)

VyMax=max(QY)
VyMin=min(QY)

figure(1)
title('deslocamento')
cmin=min(W);
cmax=max(W);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',W,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(2)
title('Mx')
cmin=min(MX);
cmax=max(MX);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MX,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(3)
title('My')
cmin=min(MY);
cmax=max(MY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MY,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(4)
title('Mxy')
cmin=min(MXY);
cmax=max(MXY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',MXY,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(5)
title('Vx')
cmin=min(QX);
cmax=max(QX);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',QX,...
      'Facecolor','interp','Marker','.');
colorbar;

figure(6)
```

```
title('Vy')
cmin=min(QY);
cmax=max(QY);
caxis([cmin cmax]);
patch('Faces',connec, 'Vertices', geom, 'FaceVertexCData',QY,...
      'Facecolor','interp','Marker','.');
colorbar;
```

Função elem_q4.m

```
% Fonte: (Khennanne, 2013)

function[coord,g] = elem_q4(i)
%
% This function returns the coordinates of the nodes of
% element i and its steering vector g
%
global nnd nel nne nodof eldof n ngp
global geom connec dee nf load
%
l=0;
coord=zeros(nne,nodof);
for k=1: nne
    for j=1:nodof
        coord(k,j)=geom(connec(i,k),j);
        l=l+1;
        g(l)=nf(connec(i,k),j);
    end
end
%
% End function elem_q4
```

Função fmlin.m

```
% Fonte: (Khennanne, 2013)

function[der,fun] = fmlin(samp, ig,jg)
%
% This function returns the vector of the shape function and their
% derivatives with respect to xi and eta
%
eta=samp(ig,1);
neta=samp(jg,1);
%

fun=[((1-eta)*(1-neta))/4;
     ((1+eta)*(1-neta))/4;
     ((1+eta)*(1+neta))/4;
     ((1-eta)*(1+neta))/4];

der=[   neta/4 - 1/4,   eta/4 - 1/4;
        1/4 - neta/4, - eta/4 - 1/4;
        neta/4 + 1/4,   eta/4 + 1/4;
      - neta/4 - 1/4,   1/4 - eta/4]';
end
% end function fmlin
```

Função Forces_at_nodes_plate.m

```
% Fonte: (Khennanne, 2013)

function[MX, MY, MXY, QX, QY]=Forces_at_nodes_plate(Element_Forces)
%
% This function averages the stresses at the nodes
%
global nnd nel nne connec
%
for k = 1:nnd
    mx = 0. ; my = 0.; mxy = 0.; qx = 0.; qy = 0.;
    ne = 0;
    for iel = 1:nel;
        for jel=1:nne;
            if connec(iel,jel) == k;
                ne=ne+1;
                mx = mx + Element_Forces(iel,1);
                my = my + Element_Forces(iel,2);
                mxy = mxy + Element_Forces(iel,3);
                qx = qx + Element_Forces(iel,4);
                qy = qy + Element_Forces(iel,5);
            end
        end
    end
    MX(k,1) = mx/ne;
    MY(k,1) = my/ne;
    MXY(k,1) = mxy/ne;
    QX(k,1) = qx/ne;
    QY(k,1) = qy/ne;
end
```

Função form_KK.m

```
% Fonte: (Khennanne, 2013)

function[KK]=form_KK(KK, kg, g)
%
% This function assembles the global stiffness matrix
%
global eldof
%
% This function assembles the global stiffness matrix
%
for i=1:eldof
    if g(i) ~= 0
        for j=1: eldof
            if g(j) ~= 0
                KK(g(i),g(j))= KK(g(i),g(j)) + kg(i,j);
            end
        end
    end
end
%
%%%%%%%%%%%%% end function form_KK %%%%%%%%%%%%%%%%%
```

Função formbeeb.m

```
% Fonte: (Khennanne, 2013)

function[beeb] = formbeeb(deriv,nne,eldof)
%
% This function assembles the matrix [beeb] from the
% derivatives of the shape functions in global coordinates
% for a thick plate element (bending action)
%
beeb=zeros(3,eldof);
for m=1:nne
    k=3*m;
    j=k-1;
    x=deriv(1,m);
    beeb(1,j)=x;
    beeb(3,k)=x;
    y=deriv(2,m);
    beeb(2,k)=y;
    beeb(3,j)=y;
end
%
% End function formbeeb
```

Função formbees.m

```
% Fonte: (Khennanne, 2013)

function[bees] = formbees(deriv,fun, nne,eldof)
%
% This function assembles the matrix [bees] from the
% derivatives of the shape functions in global coordinates
% for the shear action in a plate element
%
bees=zeros(2,eldof);
for m=1:nne
    k=3*m;
    j=k-1;
    i=k-2;
    x=deriv(1,m); y=deriv(2,m);
    bees(2,i)=-x;
    bees(1,i)=-y;
    bees(1,k) = fun(m);
    bees(2,j) = fun(m);
end
%
% End function formbees
```

Função formdeeb.m

```
% Fonte: (Khennanne, 2013)

function[deeb] = formdeeb(E,vu,thick)
%
% This function forms the elasticity matrix for a bending
% action in a plate element
%
```

```
DR= E*(thick^3)/(12*(1.-vu*vu));
%
deeb=DR*[1 vu 0. ;...
         vu 1 0. ;...
         0. 0. (1.-vu)/2] ;
%
% end function fromdeeb
```

Função formdees.m

```
% Fonte: (Khennanne, 2013)

function[dees] = formdees(E,vu,thick)
%
% This function forms the elasticity matrix for the shear
% action in a thick plate element
%
G= E/(2*(1.+vu));
%
dees=G*[thick 0 ;...
         0 thick];
%
% end function fromdees
```

Função gauss.m

```
% Fonte: (Khennanne, 2013)

function[samp]=gauss(ngp)
%
% This function returns the abscissas and weights of the Gauss
% points for ngp equal up to 4
%
%
samp=zeros(ngp,2);
%
if ngp==1
    samp=[0  2];
elseif ngp==2
    samp=[-1/sqrt(3) 1;...
           1/sqrt(3) 1];
elseif ngp==3
    samp= [-0.2*sqrt(15) 5/9;...
            0            8/9;...
            0.2*sqrt(15) 5/9];
elseif ngp==4
    samp= [-0.861136311594053 0.347854845137454; ...
           -0.339981043584856 0.652145154862546; ...
            0.339981043584856 0.652145154862546; ...
            0.861136311594053 0.347854845137454];
end
%
% End function Gauss
```

Função plateelem_q4.m

```
% Fonte: (Khennanne, 2013)
```

```
function[coord,g] = platelem_q4(i)
%
% This function returns the coordinates of the nodes of element i
% and its steering vector
%
global nne nodof geom connec nf dim
%
coord=zeros(nne,dim);
for k=1: nne
    for j=1:dim
        coord(k,j)=geom(connec(i,k),j);
    end
end
%
l=0;
for k=1: nne
    for j=1:nodof
        l=l+1;
        g(l)=nf(connec(i,k),j);
    end
end
%
% End function platelem_q8
```

Rotina Q4_mesh.m

```
% Fonte: (Khennanne, 2013)

% This module generates a mesh of linear quadrilateral elements
%
global nnd nel nne nodof eldof n
global geom connec dee nf Nodal_loads
global Length Width NXE NYE X_origin Y_origin dhx dhy
%
%
nnd = 0;
k = 0;
for i = 1:NXE
    for j=1:NYE
        k = k + 1;
        n1 = j + (i-1)*(NYE + 1);
        geom(n1,:) = [(i-1)*dhx - X_origin (j-1)*dhy - Y_origin ];
        n2 = j + i*(NYE+1);
        geom(n2,:) = [i*dhx - X_origin (j-1)*dhy - Y_origin ];
        n3 = n1 + 1;
        geom(n3,:) = [(i-1)*dhx - X_origin j*dhy - Y_origin ];
        n4 = n2 + 1;
        geom(n4,:) = [i*dhx- X_origin j*dhy - Y_origin ];
        nel = k;
        connec(nel,:) = [n1 n2 n4 n3];
        nnd = n4;
    end
end
%
```