ISSN 1809-5860

ABORDAGEM ORIENTADA A OBJETOS PARA IMPLEMENTAÇÃO COMPUTACIONAL DE ELEMENTOS FINITOS DE PLACAS

Samir Silva Saliba[1], Samuel Silva Penna[2] & Roque Luiz Pitangueira[3]

Resumo

Este artigo apresenta um programa computacional para análise estrutural de placas que disponibiliza vários elementos finitos, baseados nas teorias de Kirchhoff e de Reissner-Mindlin. Essas teorias são discutidas brevemente, destacando-se diferenças e particularidades relevantes para o projeto do software. A implementação do sistema, realizada segundo o paradigma de programação orientada a objetos, é então apresentada através de diagramas Unified Modelling Language (UML) que descrevem as principais classes e interfaces utilizadas. Para fins de ilustração e validação dos recursos disponibilizados, algumas simulações numéricas são apresentadas. Por fim, a importância do sistema para apoiar o estudo do tema e as possibilidades de expansão do mesmo são discutidas.

Palavras-chave: Método dos Elementos Finitos. Teorias de Placas. Programação Orientada a Objetos.

OBJECT-ORIENTED APPROACH TO COMPUTER IMPLEMENTATION OF PLATE FINITE ELEMENTS

Abstract

This article presents a software for structural analysis of plates that provides a variety of finite elements, based on the theories of Kirchhoff and Reissner-Mindlin. These theories are briefly discussed, highlighting differences and particularities relevant to software design. The implementation of the system, conducted under the paradigm of object-oriented programming, is then presented through Unified Modelling Language (UML) diagrams that describe the main classes and interfaces used. For purposes of illustration and validation of the resources available, some numerical simulations are presented. Finally, the importance of the system to study the issue and the possibilities of its expansion are discussed.

Keywords: Finite Element Method. Theories of Plates. Object Oriented Programming.

1 INTRODUÇÃO

Modelos matemáticos, expressos por equações diferenciais, fornecem soluções analíticas para problemas de flexão de placas com configurações geométrica, de carregamento e de contorno específicas. Para geometria, cargas e condições de contorno genéricas, soluções aproximadas podem ser obtidas por meio de modelos numéricos discretos. Entre os modelos discretos, os baseados no Método dos Elementos Finitos (MEF) são os mais utilizados.

Programar computadores para modelos do MEF não é novidade. Entretanto, esta tarefa pode ser realizada de maneira inadequada, resultando em produtos dependentes de sistema operacional, pouco amigáveis, escritos em linguagens de programação não apropriadas, de expansão, distribuição e manutenção difíceis, desenvolvidos por equipes fechadas, com documentação deficiente, entre outras limitações que tornam os programas computacionais tão específicos quanto as referidas

[1]Mestre pelo Programa de Pós-Graduação em Engenharia de Estruturas da UFMG - PROPEES, samirsaliba@gmail.com
[2]Doutorando do PROPEES, spenna@dees.ufmg.br
[3] Professor do PROPEES, roque@dees.ufmg.br

soluções analíticas. Esse cenário confronta-se com o surgimento e aprimoramento de recursos para desenvolvimento de programas computacionais, como o paradigma de programação orientada a objetos, linguagens de programação e de modelagem de sistemas computacionais que o suportam, linguagens de marcação para transferência de dados, sistemas de controle de versões, testes unitários, padrões de projeto de programas computacionais, entre outros. Assim, desenvolver um sistema computacional amigável à mudanças, que atenda desde modelos mais simples até modelos extremamente sofisticados do MEF, sem ter que recomeçar o processo a cada novo aperfeiçoamento, é um desafio na área de métodos computacionais aplicados à engenharia. Enfrentar este desafio é imperativo para que a pesquisa na área disponha de recursos computacionais que, a partir de conceitos já consolidados, possibilitem o aprimoramento progressivo dos modelos através da ampliação de complexidades.

Este artigo apresenta um exemplo desse enfrentamento e ilustrá-o por meio de uma abordagem orientada a objetos (OO) para modelos de placas do MEF. Inicialmente, resume-se brevemente as bases dos modelos do MEF para placas. Tal revisão aponta as principais diferenças e particularidades das teorias de Kirchhoff e de Reissner-Mindlin, visando o projeto de implementação. Em seguida, discute-se o projeto OO que, usufruindo de todo potencial de generalização do MEF, gerou o programa computacional que permite a análise estrutural de placas. Simulações numéricas que validam as implementações e ilustram as possibilidades de análise são, então, apresentadas. Por fim, a importância do sistema para apoiar o estudo do tema é discutida.

2 ELEMENTOS FINITOS DE PLACA BASEADOS NA TEORIA DE KIRCHHOFF

A Teoria de Kirchhoff fundamenta-se em quatro hipóteses simplificadoras, ilustradas na Figura 1: (1) os pontos localizados no plano médio possuem somente deslocamentos verticais (w); (2) todos os pontos contidos numa reta normal ao plano médio possuem o mesmo deslocamento vertical; (3) a tensão normal na direção z (σ_z) é desprezível e (4) retas normais ao plano médio no estado indeformado da placa permanecem retas e normais a este plano, após sua deformação. Partindo dessas hipóteses, é possível obter o campo de deslocamentos da placa, dados pelas equações:

$$\begin{aligned} u(x,y,z) &= -z\theta_x(x,y) = -z\frac{\partial w(x,y)}{\partial x} \\ v(x,y,z) &= -z\theta_y(x,y) = -z\frac{\partial w(x,y)}{\partial y} \\ w(x,y,z) &= w(x,y) \end{aligned} \qquad (1)$$

A Eq. (1) mostra que a descrição cinemática e, após o cálculo das deformações e aplicação de relações constitutivas, a descrição estática de qualquer ponto da placa dependem somente da função $w(x,y)$, que fornece os deslocamentos verticais dos pontos do plano médio. Assim, nos elementos finitos baseados na Teoria de Kirchhoff, uma aproximação para $w(x,y)$ fornece uma descrição completa do problema. A referida aproximação é normalmente escrita em função de valores nodais de $w(x,y)$ e de suas derivadas, gerando elementos finitos de continuidade C^1 ou superior, como os mostrados na Figura 2. O elemento retangular não-conforme MZC (Figura 2(a)), desenvolvido por MELOSH (1961) e ZIENKIEWICZ e CHEUNG (1964), possui três graus de liberdade por nó: um deslocamento vertical (w) e duas rotações (θ_x e θ_y). Na Figura 2(b) é apresentado o elemento retangular conforme BFS elaborado por BOGNER, FOX e SCHMIT (1965), que possui quatro graus de liberdade por nó: um deslocamento vertical (w), duas rotações (θ_x e θ_y) e uma curvatura ($w_{,xy}$). Já o elemento triangular não-conforme CKZ (Figura 2(c)), desenvolvido por CHEUNG *et al*. (1968), possui os mesmos três graus de liberdade do elemento MZC, ou seja, um deslocamento vertical (w) e duas

rotações (θ_x e θ_y). Finalizando, na Figura 2(d) é apresentado o elemento triangular conforme desenvolvido por COWPER *et al*. (1968), cujos graus de liberdade são: um deslocamento vertical (w), duas rotações (θ_x e θ_y) e três curvaturas ($w_{,xx}$, $w_{,yy}$ e $w_{,xy}$). Outros detalhes sobre os quatro elementos mostrados na Figura 2 podem ser obtidos em OÑATE (1995), SALIBA (2007) e WEAVER (1984). Estes são os elementos de Kirchhoff implementados no sistema computacional que será apresentado.

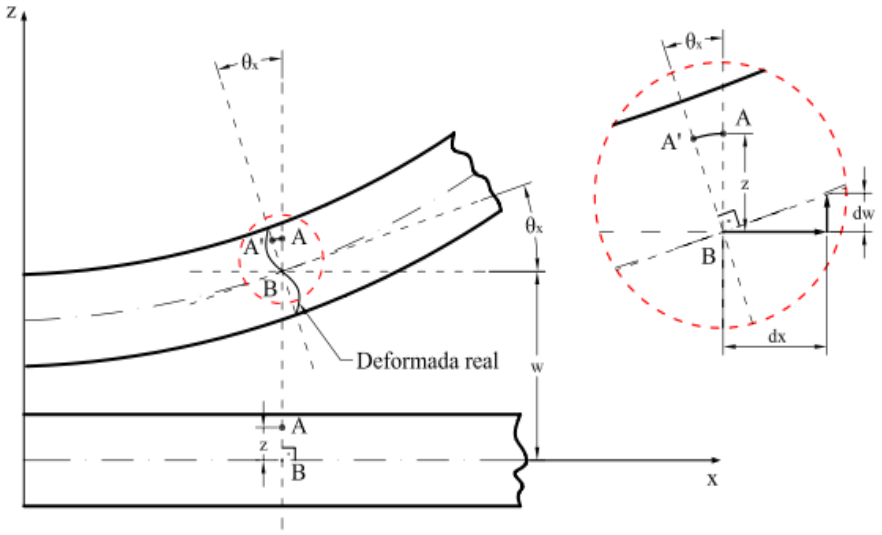


Figura 1 – Campo de deslocamentos segundo a Teoria de Placas de Kirchhoff.

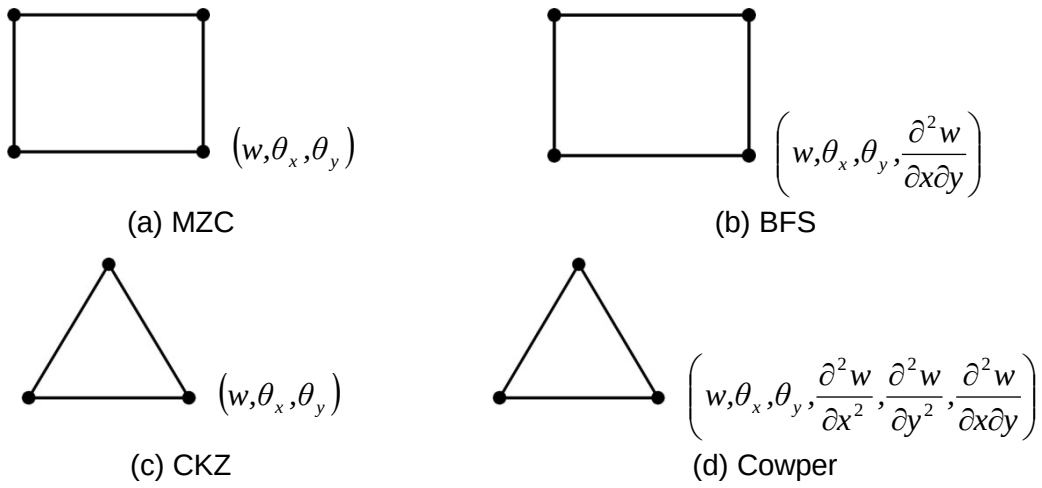


Figura 2 – Elementos finitos baseados na Teoria de Kirchhoff.

3 ELEMENTOS FINITOS DE PLACA BASEADOS NA TEORIA DE REISSNER-MINDLIN

A Teoria de Reissner-Mindlin repete as mesmas três primeiras hipóteses da Teoria de Kirchhoff e altera a quarta hipótese para (ver Figura 3): (4) retas normais ao plano médio no estado indeformado da placa permanecem retas mas não necessariamente normais a este plano, após sua deformação. Esta alteração se reflete no campo de deslocamentos da placa, que fica como mostrado na Figura 3 e nas equações:

$$u(x,y,z) = -z\theta_x(x,y)$$
$$v(x,y,z) = -z\theta_y(x,y), \quad \text{com} \quad \theta_x = w_{,x} + \varphi_x \quad \theta_y = w_{,y} + \varphi_y \tag{2}$$
$$w(x,y,z) = w(x,y)$$

A Eq. (2) mostra que, diferentemente da Teoria de Kirchhoff, as rotações θ_x e θ_y são independentes do deslocamento w(x,y). Essa independência permite formular elementos de continuidade C^0, como os mostrados na Figura 4. Em todos esses elementos, aproximações independentes para w(x,y), $\theta_x(x,y)$ e $\theta_y(x,y)$ são facilmente escritas devido à generalização da formulação paramétrica do MEF. Nos elementos Q4, Q8, Q9, T3, T6 e T10, todos os nós possuem três graus de liberdade: um deslocamento vertical e duas rotações. Já o elemento Q9H (Figura 4(d)), chamado de Heterosis e desenvolvido por HUGHES e COHEN (1978), possui os nós periféricos com três graus de liberdade e o nó central com dois, as rotações. Mais detalhes sobre estes elementos podem ser obtidos em OÑATE (1995), SALIBA (2007) e WEAVER (1984). Estes são os elementos de Reissner-Mindlin implementados no sistema computacional que será apresentado.

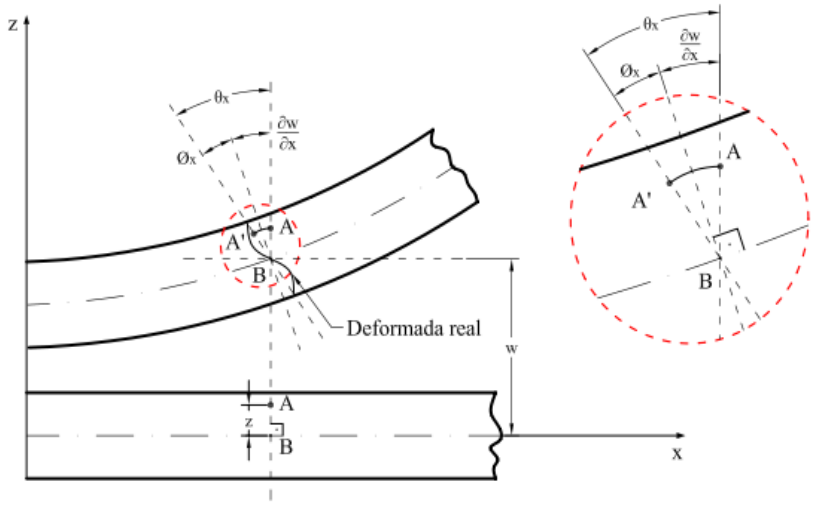


Figura 3 – Campo de deslocamentos segundo a Teoria de Placas de Reissner-Mindlin.

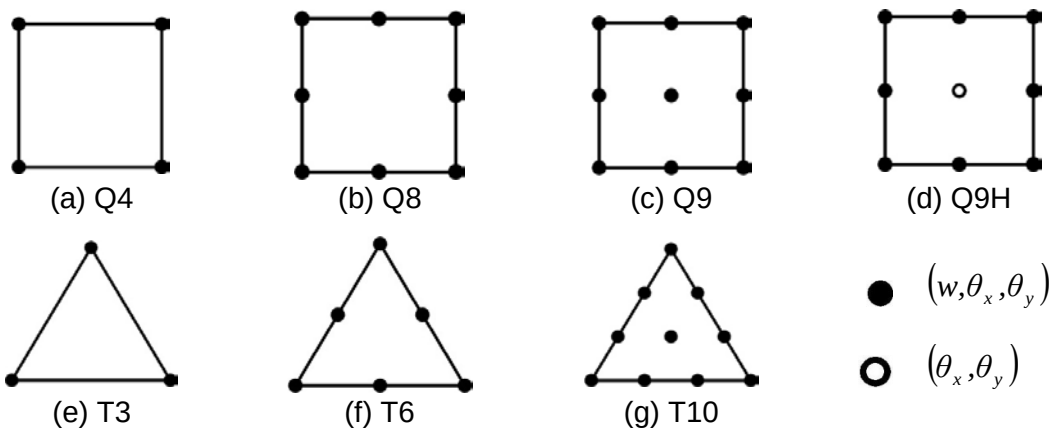


Figura 4 – Elementos Finitos baseados na Teoria de Reissner-Mindlin.

A teoria de Reissner-Mindlin se destaca por ser aplicável em placas de qualquer espessura. Entretanto, sua aplicação em placas finas requer cuidados especiais, para que os elementos sejam

capazes de representar casos críticos de deformação de cisalhamento (SORIANO, 2003). No programa computacional que será apresentado, adotou-se: (1) Integração Reduzida, que consiste na redução do número de pontos utilizados para integrar a matriz de rigidez do elemento; (2) Integração Seletiva, que também requer a redução do número de pontos de integração, porém, tal redução é feita somente para a integração da parcela da matriz de rigidez referente ao cisalhamento; (3) Deformação de Cisalhamento Imposta, que consiste na imposição de campo de deformação de cisalhamento que substitui o original.

4 ABORDAGEM ORIENTADA A OBJETOS PARA ELEMENTOS FINITOS DE PLACA

A implementação computacional dos vários elementos finitos de placa anteriormente citados foi realizada no *INSANE (INteractive Structural ANalysis Environment)*, um sistema de código aberto que vem sendo desenvolvido desde o ano 2002 (PITANGUEIRA *et al*., 2008). O *INSANE* possui várias aplicações independentes que se relacionam através de arquivos persistidos em formato XML – eXtensible Markup Language – (RAY, 2000). Estas aplicações foram implementadas em JAVA, seguindo rigorosamente as premissas do paradigma de Programação Orientada a Objetos, o que gerou um sistema modular, de fácil manutenção e expansão.

4.1 Generalização para modelos discretos e problemas diversos

O núcleo numérico do *INSANE* é o responsável pela obtenção dos resultados de diferentes modelos discretos de análise estrutural. Ele é formado por interfaces e classes que representam as diversas abstrações de uma resolução numérica de modelos discretos, cada qual com sua hierarquia de classes responsável por cumprir o seu devido papel no processamento. O diagrama *Unified Modelling Language* (LARMAN, 2000) (UML) da Figura 5 mostra que todo o núcleo numérico é baseado nas relações entre as interfaces *Assembler*, *Model* e *Persistence*, além da classe abstrata *Solution.*

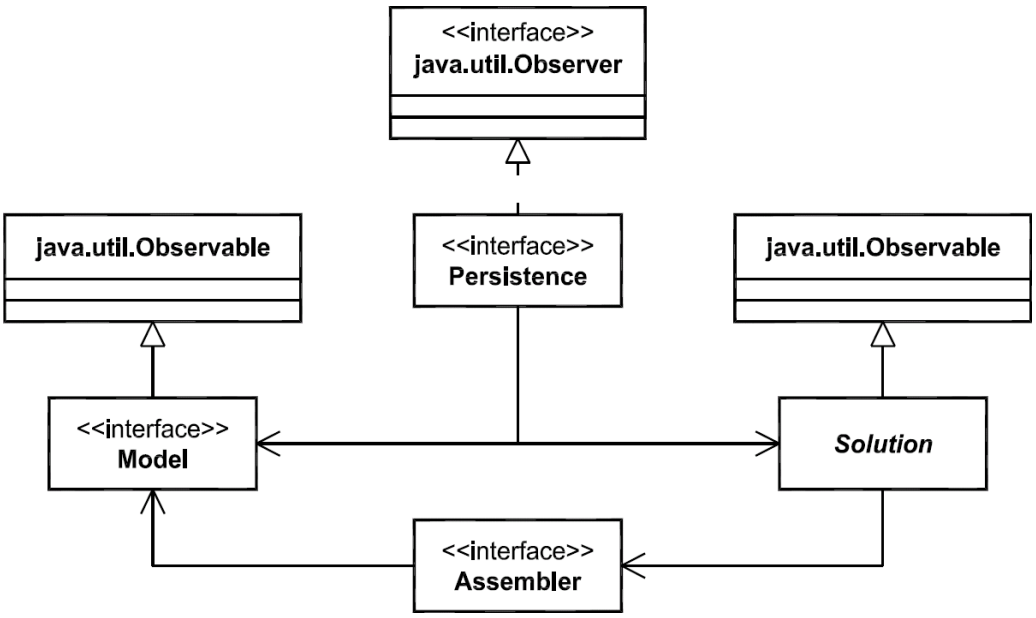


Figura 5 – Organização do núcleo numérico do *INSANE.*

A interface *Assembler* é a responsável por montar o sistema matricial de segunda ordem com o qual é possível representar diversos tipos de problemas discretos:

$$\mathbf{A\,X'' + B\,X' + C\,X = R - F} \tag{3}$$

em que **X** é o vetor de variáveis de estado do problema; **X'** e **X''** são os vetores com, respectivamente, a primeira e a segunda variação temporal da variável de estado; **A**, **B** e **C** são as matrizes dos coeficientes, que podem ou não depender da variável de estado e suas derivadas; e **R** e **F** representam os termos independentes do sistema de equações.

A classe abstrata *Solution* (Figura 5) é quem desencadeia o processo de solução, possuindo os recursos necessários para resolver este sistema matricial, seja o sistema linear ou não-linear. *Model* possui os dados relativos ao modelo discreto a ser analisado e fornece para *Assembler* todas as informações necessárias para montar a equação do modelo, que será resolvida por *Solution*. Tanto *Model* como *Solution* se comunicam com a interface *Persistence*, que trata os dados de entrada e, principalmente, persiste os dados de saída para as demais aplicações, sempre que observa alterações no estado do modelo discreto. Este processo de observação de alterações ocorre segundo o padrão de projeto de software *Observer-Observable* (GAMA *et al*., 1995), que é um mecanismo de propagação de mudanças. Quando um objeto dito *observador* (que implementa a interface *java.util.Observer*) é criado, ele é inscrito na lista de observadores dos objetos ditos *observados* (que estendem a interface *java.util.Observable*). Quando alguma mudança ocorre no estado de um objeto observado, é disparado o mecanismo de propagação de mudanças, que se encarrega de notificar os objetos observadores para se atualizarem. Isto garante a consistência e a comunicação entre o componente observador (*Persistence*) e os componentes observados (*Solution* e *Model*).

4.2 Especialização para modelos discretos do MEF

A interface *Assembler* (Figura 6) possui os métodos necessários para montar as matrizes e vetores do modelo discreto. Esta interface é implementada pela classe *FemAssembler*, visando especializá-la para os diversos tipos de problemas que podem ser modelados através do Método dos Elementos Finitos. Conforme mostra a Figura 6, a classe *FemAssembler* tem como atributo um objeto do tipo *Model*, que é o modelo de elementos finitos para o qual deve montar a equação. Cominicando-se com *Model* e implementando os agoritmos do método da rigidez direta, *FemAssembler* é capaz de responder à classe *Solution* por todas as parcelas da Eq. (3).

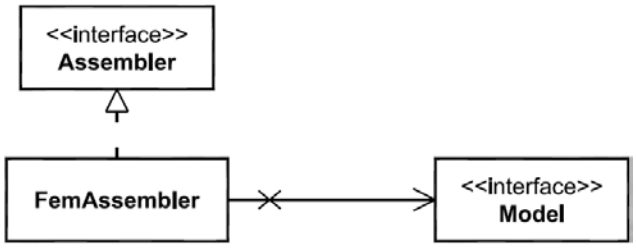


Figura 6 – Especialização da Interface *Assembler.*

O modelo discreto a ser analisado é representado no projeto orientado a objetos do núcleo numérico pela interface *Model* (Figura 6). A classe que implementa *Model*, para representar da forma mais geral possível um modelo discreto, é constituída por listas de objetos inerentes a este modelo e contém métodos de acesso e manipulação destas listas. Conforme mostra a Figura 7, *Model* é implementada pela classe *FemModel*, que representa o modelo do MEF propriamente dito.

Ainda conforme a Figura 7, um objeto *FemModel* possui listas de nós, elementos, funções de forma, carregamentos, materiais, modelos constitutivos, dentre outras. *FemModel* estende também a

classe *Observable*, pois é observada pela persistência. Para ilustrar o relacionamento entre as várias entidades de *FemModel*, apresenta-se na Figura 8 o diagrama de sequência correspondente à montagem da matriz C de um modelo de elementos finitos. Independentemente do significado de C para o problema em questão – matriz de rigidez no caso de um problema de análise de tensão da mecânica dos sólidos – sua montagem segue os passos ilustrados na figura. Como se pode observar na figura, a obtenção da matriz C para um problema em particular depende da especialização de quatro classes principais: *Element*, *ProblemDriver*, *Shape* e *AnalysisModel*. Na seção a seguir, mostra-se como estas quatro classes são especializadas para um problema de análise de placas.

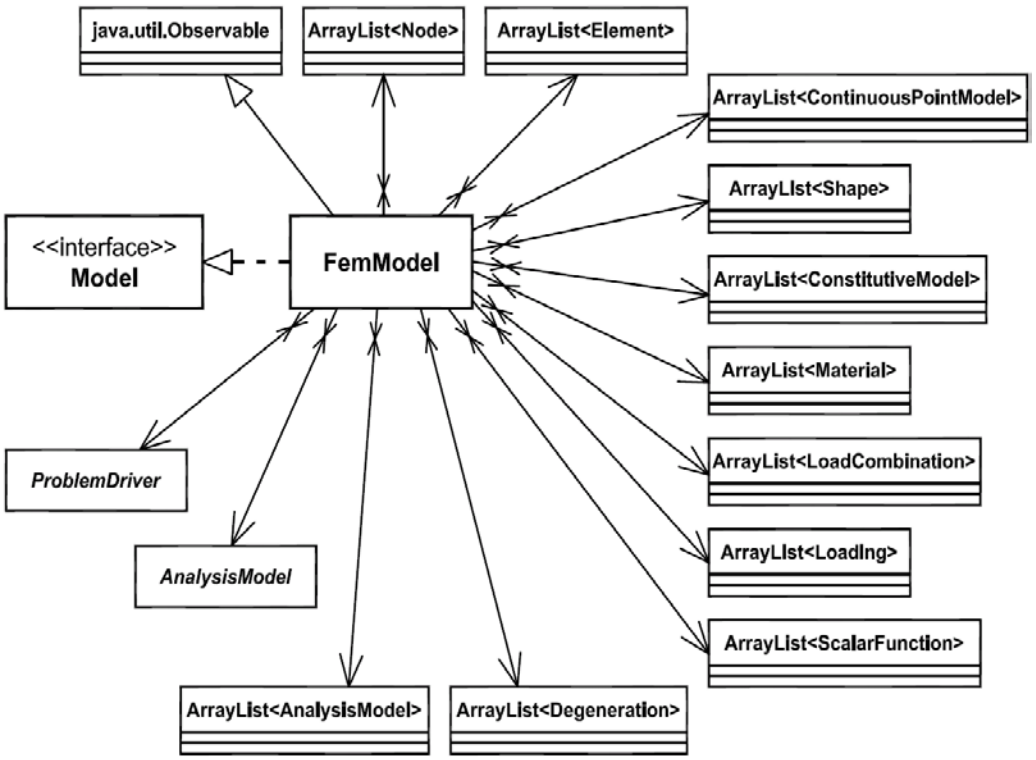


Figura 7 – Especialização da interface *Model.*

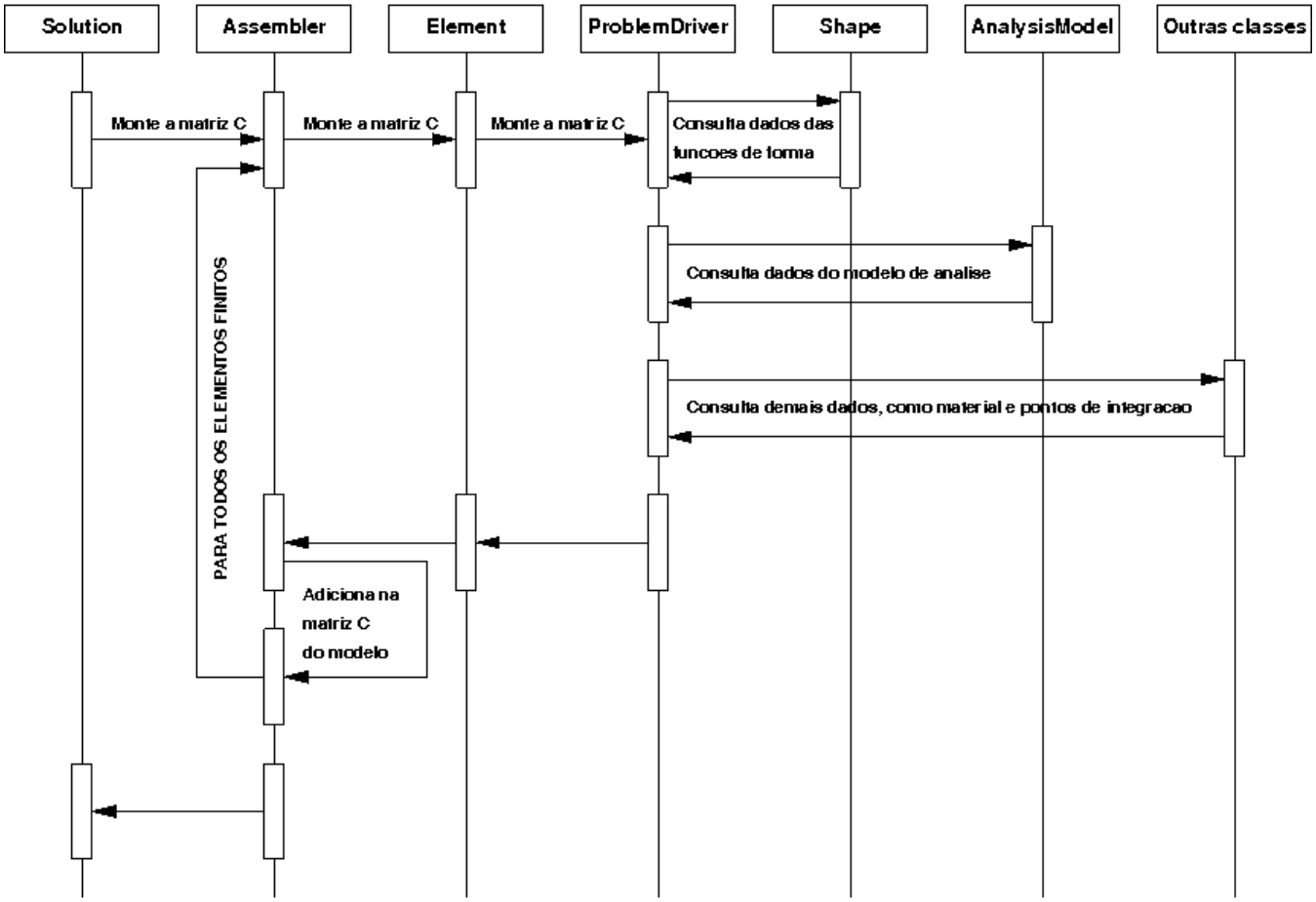


Figura 8 – Diagrama de sequência para montagem da matriz C.

4.3 Especialização para elementos finitos de placas

Como observado anteriormente (Figura 8), a generalização para montagem das diversas matrizes associadas a determinado problema depende das classes abstratas *Element*, *ProblemDriver*, *Shape* e *AnalysisModel*.

A Figura 9 mostra o diagrama de herança da classe abstrata *Element,* que representa os diversos tipos de elementos finitos. A especialização da mesma para elementos obtidos por meio da formulação paramétrica do MEF é feita com a classe *ParametricElement*. Devido ao procedimento para posicionar os pontos de integração no domínio do elemento, a classe *ParametricElement* é especializada em classes que representam elementos paramétricos unidimensionais (classe *Bar*), elementos bidimensionais (classes *Quadrilateral* e *Triangular*) e elementos tridimensionais (classes *Tetrahedral* e *Hexahedral*). Os elementos finitos de placas de Reissner-Mindlin (Figura 4) são, portanto, do tipo *Quadrilateral* ou *Triangular*. A Figura 9 mostra outra especialização da classe *Element*, relevante para problemas de placas. Trata-se da classe *ThinPlateElement,* que representa os elementos finitos de placas finas baseados na Teoria de kirchhoff, mostrados na Figura 2.

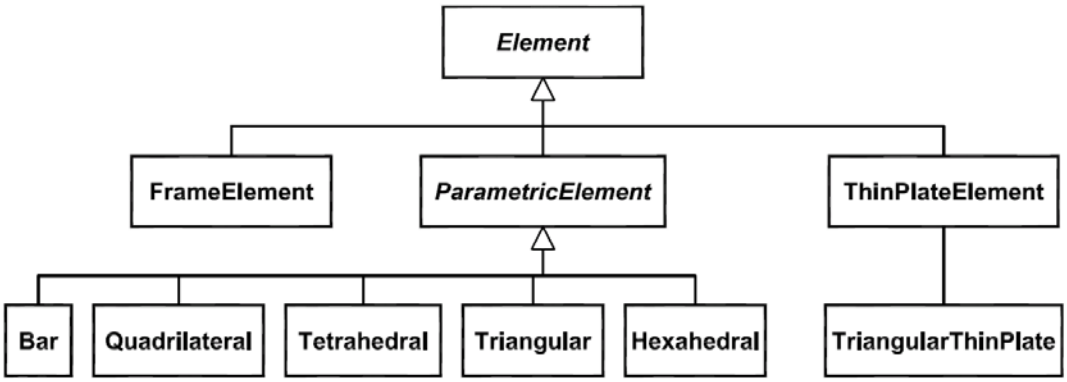


Figura 9 – Especialização da classe *Element.*

A classe *Element* é uma abstração matemática com atributos que permitem sua especialização para diferentes contextos. Mesmo com a especialização em três níveis de herança, mostrada na Figura 9, a classe *Element*, por si só, não destingue para qual tipo de problema um elemento finito está sendo usado. Por este motivo possui como atributo um objeto *ProblemDriver*, que armazena as informações referentes ao tipo de problema que o elemento modela. Conforme mostram as Figuras 8 e 10, a especialização da classe abstrata *ProblemDriver* permite informar as parcelas de cada elemento na equação do modelo, sejam elas incrementais ou totais, dando significado às matrizes e vetores da equação 3. Diversas classes são derivadas de *ProblemDriver*, representando os diversos tipos de problemas que podem ser solucionados a partir de modelos discretos. A subclasse *Parametric*, que estende *SolidMech*, que por sua vez estende *ProblemDriver*, é responsável por obter a matriz de rigidez do elemento de forma numérica, como nos elementos de Reissner-Mindlin da Figura 4, enquanto que a classe *KirchhoffThinPlate* permite a obtenção da matriz de rigidez analiticamente, para os elementos de Kirchhoff da Figura 2. No último nível da especialização, derivam-se as classes *ReissnerMindlinThinPlate* e *RMThinPlateSubstitutiveShearStrain*. Estas classes referem-se aos modelos que usam as técnicas anti-bloqueio, respectivamente, de integração seletiva e de deformação de cisalhamento imposta.

A classe *Element* (Figura 9), por si só, também não destingue se um elemento triangular é de três, seis ou dez nós, nem se um elemento quadrilateral é de quatro, oito ou nove nós. Assim, *Element* possui um objeto do tipo *Shape* como atributo. Conforme mostram as Figuras 8 e 11, por meio da classe abstrata *Shape,* é possível especializar a aproximação da variável de estado do problema, para um elemento, calculando-se as funções de forma e suas derivadas. Primeiramente, a classe *Shape* se especializa de acordo com o sistema de coordenadas utilizado: cartesiano ou natural. Da classe *NaturalCoordsShape* derivam-se classes para elementos cujas funções de forma são escritas em coordenadas naturais. Os elementos de placas de Reissner-Mindlin (Figura 4) podem ser quadrilaterais ou triangulares, logo, são representados por, respectivamente, *NCSQuadrilateral* e *NCSTriangular*. Para utilização da técnica de deformação de cisalhamento imposta, é necessário conhecer os pontos onde serão avaliadas as deformações de cisalhamento, os denominados pontos de colocação. Para tal existe a interface *Colocation,* implementada pelas classes *RMCIT6, RMCIQ4, RMCIQ8 e RMCIQ9, com o objetivo de especificar o posicionamento dos pontos de colocação nos elementos triangular de 6 nós e quadrilaterais de 4, 8 e 9 nós, respectivamente.* Já as classes *AnalyticCowper*, *AnalyticCKZ*, *AnalyticMZC* e *AnalyticBFS* são empregadas para os elementos de Kirchhoff (Figura 2).

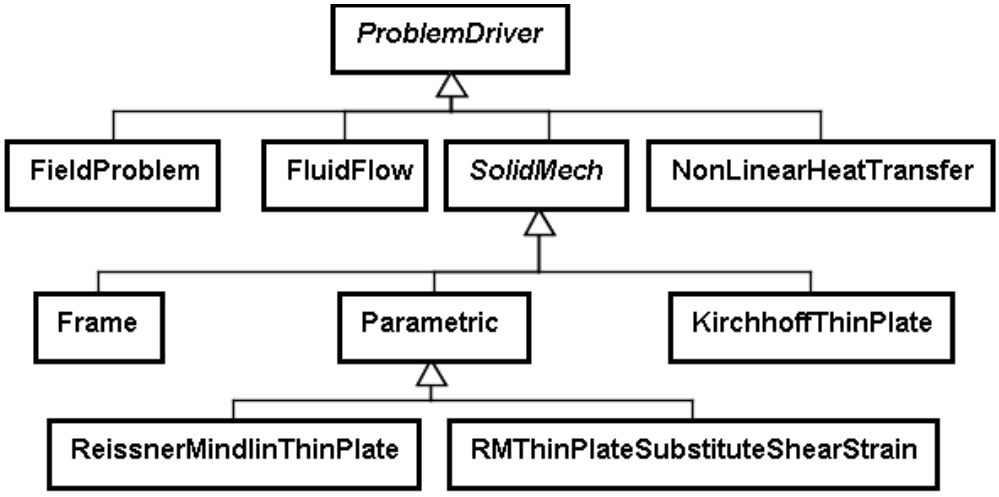


Figura 10 – Especialização da classe *ProblemDriver.*

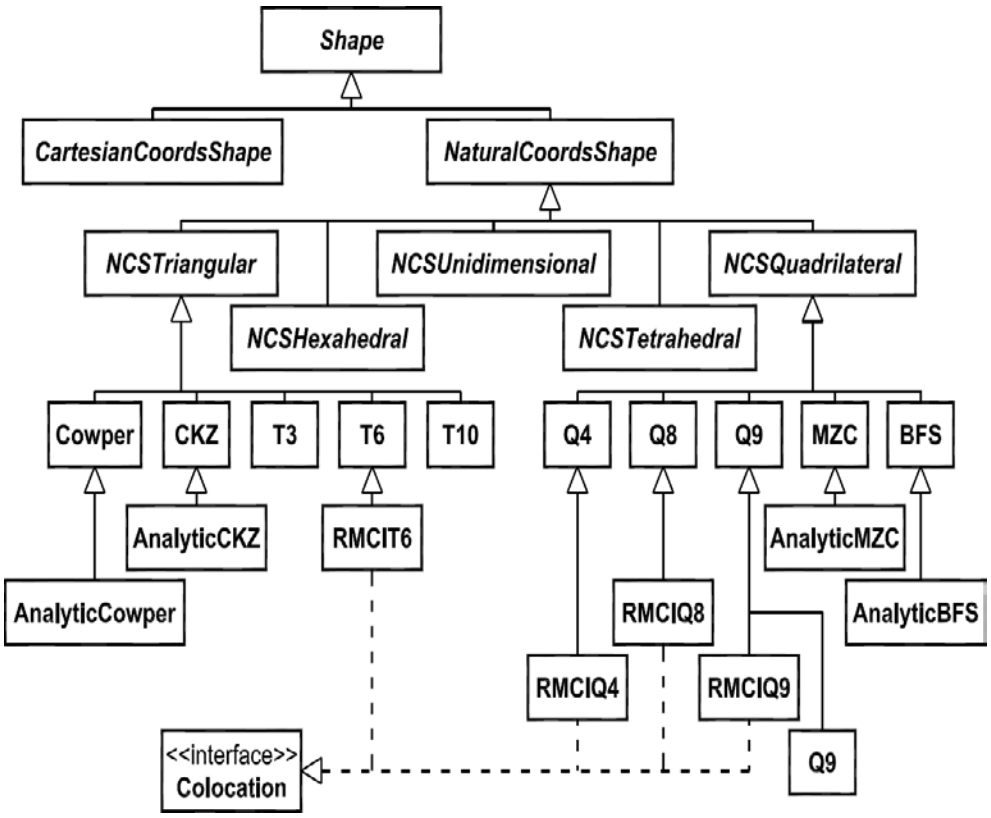


Figura 11 – Especialização da classe *Shape.*

A classe abstrata *Element* (Figura 10) também é incapaz de destinguir se um elemento triangular ou quadrilateral está sendo usado na modelagem de um problema de estado plano de tensão, axissimétrico ou de placa. Esta habilidade é dada à classe *Element* por meio de uma referência à classe *AnalysisModel*, cuja hierarquia está mostrada na Figura 12. Cabem aos objetos desta classe informar, dentre outras grandezas, a quantidade e a natureza dos graus de liberdade de cada nó e das deformações e tensões generalizadas de cada ponto interno do elemento finito. No caso dos elementos de placas, esta tarefa é devida aos objetos das subclasses KirchhoffPlate e ReissnerMindlinPlate e às classes delas derivadas.

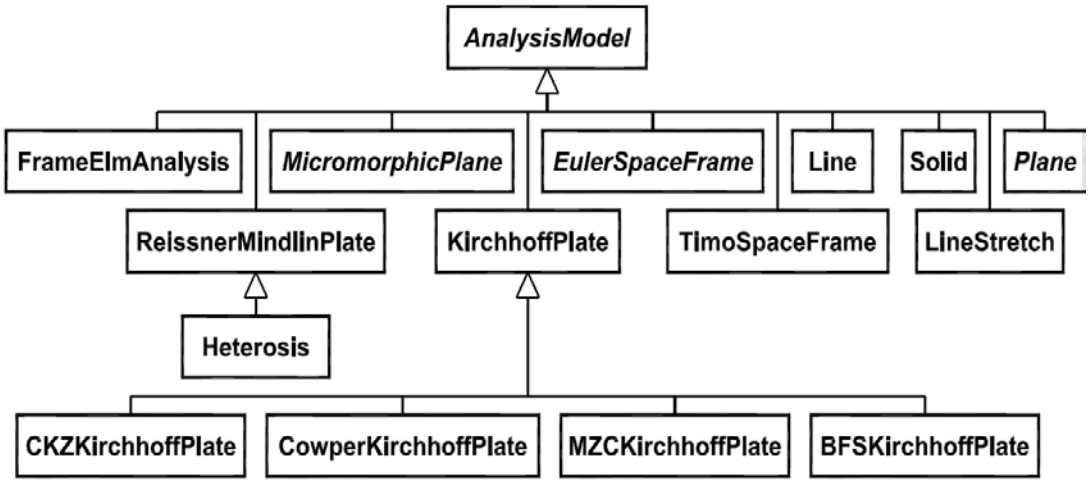


Figura 12 – Especialização da classe AnalysisModel.

O projeto orientado a objetos acima descrito disponibilizou os elementos finitos ilustrados nas Figuras 2 e 4, além das técnicas anti-bloqueio de integração reduzida e seletiva e de deformação de cisalhamento imposta. A seguir, algumas simulações numéricas ilustram o uso destes recursos.

5 SIMULAÇÕES COMPUTACIONAIS

Para ilustrar e validar a implementação proposta neste trabalho, apresentam-se três simulações numéricas: um estudo de convergência, uma comparação com solução analítica e uma comparação com resultados do software ABAQUS.

5.1 Placa fina quadrada

Com o objetivo de mostrar a convergência dos elementos implementados, analisa-se uma placa quadrada simplesmente apoiada, sob a ação de uma carga concentrada no centro (Figura 13). A placa possui lados de comprimento igual a *10 uc* (*uc* = unidade de comprimento) e espessura igual a *0,1 uc*, a carga aplicada é igual a *1 uf* (*uf* = unidade de força) e as propriedades do material são: módulo de elasticidade longitudinal igual a *1092000 uf/uc*^{2} e coeficiente de Poisson igual a *0,3*. Devido à dupla simetria do problema somente ¼ da placa foi discretizado, conforme as malhas apresentadas na Figura 13. Por se tratar de um problema de placa fina e por objetivar a validação da implementação, realizou-se a análise utilizando as três técnicas anti-bloqueio para elementos de Reissner-Mindlin. A solução analítica para o deslocamento vertical no centro desta placa, segundo TIMOSHENKO e WOINOWSKY-KRIEGER (1959) é $w_{centro} = -0,0116\ uc$. A Figura 14 apresenta os resultados obtidos na análise da placa para todos os elementos implementados e para as várias malhas mostradas na Figura 13.

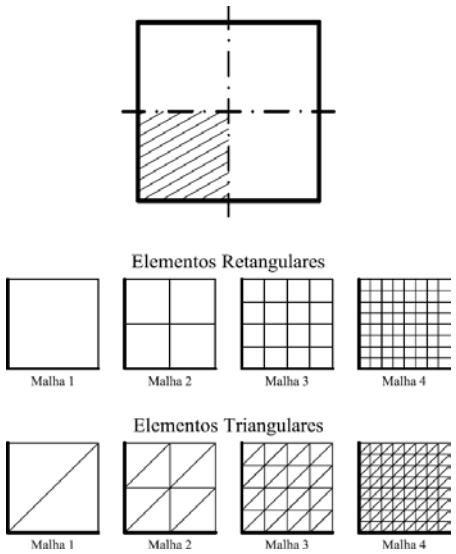
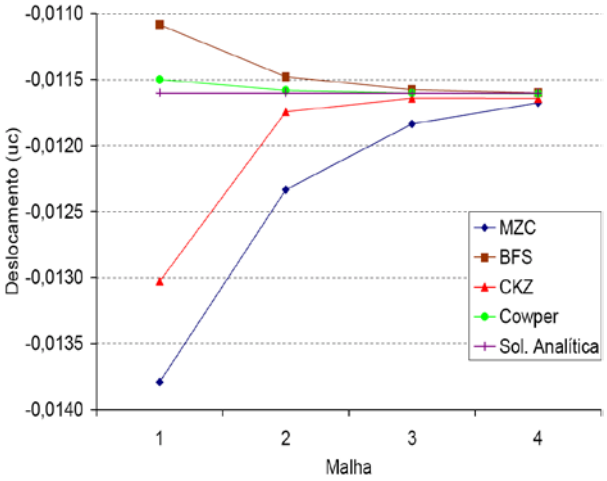
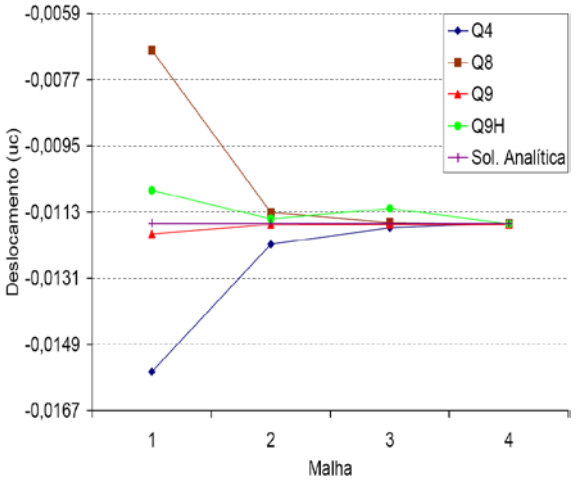


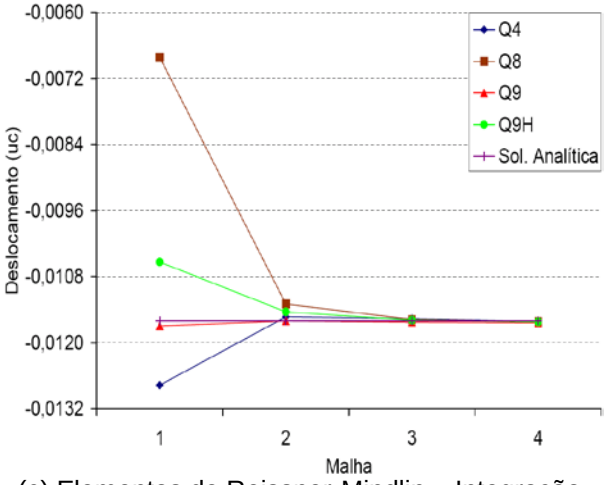
Figura 13 – Malhas utilizadas na análise da placa fina quadrada simplesmente apoiada.



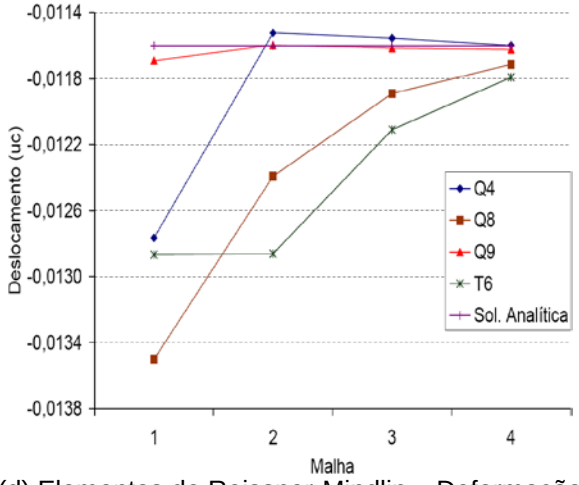
(a) Elementos de Kirchhoff



(b) Elementos de Reissner-Mindlin – Integração Reduzida



(c) Elementos de Reissner-Mindlin – Integração Seletiva



(d) Elementos de Reissner-Mindlin – Deformação de Cisalhamento Imposta

Figura 14 – Deslocamento vertical no centro da placa quadrada.

5.2 Placa elíptica

Neste exemplo, analisa-se uma placa elíptica com semi-eixo menor de *5,0 uc*, semi-eixo maior de *10 uc*, *0,1 uc* de espessura, engastada e submetida um carregamento distribuído (*q=-1 uf/uc²*) sobre toda sua superfície. Apenas ¼ da placa foi discretizada, devido à dupla simetria, num total de 192 elementos Q4, conforme apresentado na Figura 15. Por se tratar de elementos de Reissner-Mindlin para placa fina, aplicou-se a técnica anti-bloqueio de deformação de cisalhamento imposta. A placa é constituída de material com módulo de elasticidade longitudinal igual a *1092000 uf/uc²* e coeficiente de Poisson igual a *0,3*. A Figura 16 apresenta os resultados para os deslocamentos verticais ao longo do semi-eixo maior da placa, obtidos com a solução analítica (TIMOSHENKO e WOINOWSKY-KRIEGER, 1959) e com a discretização adotada.

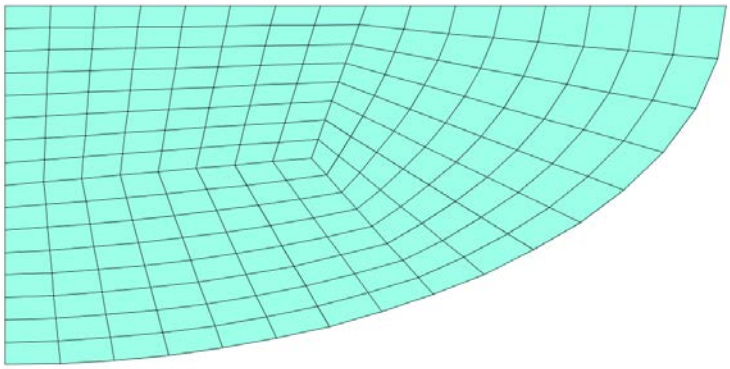


Figura 15 – Malha utilizada na análise da placa elíptica engastada.

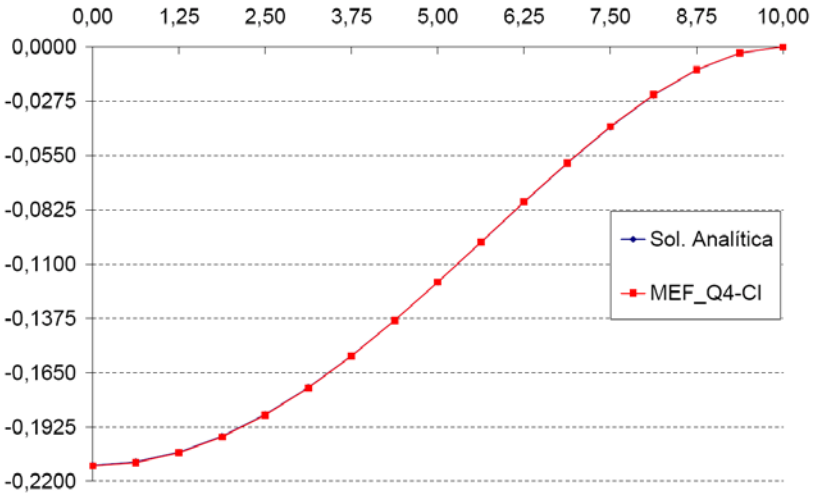


Figura 16 – Deslocamentos verticais ao longo do semi-eixo maior da placa elíptica.

5.3 Placa em "U" moderadamente espessa

A placa com configuração geométrica e condições de contorno mostradas na Figura 17, submetida a carregamento uniformemente distribuído igual a *-10 uf/uc²*, foi discretizada com 500 elementos quadrilaterais de oito nós (Q8), conforme a malha da Figura 18. Esta mesma placa foi analisada com o software ABAQUS 6.7, utilizando uma malha equivalente. O material da placa possui módulo de elasticidade longitudinal igual a *20.000.000 uf/uc²* e coeficiente de Poisson igual a *0,25*. A Figura 19 mostra as configurações deformadas da placa, obtidas com os dois softwares.

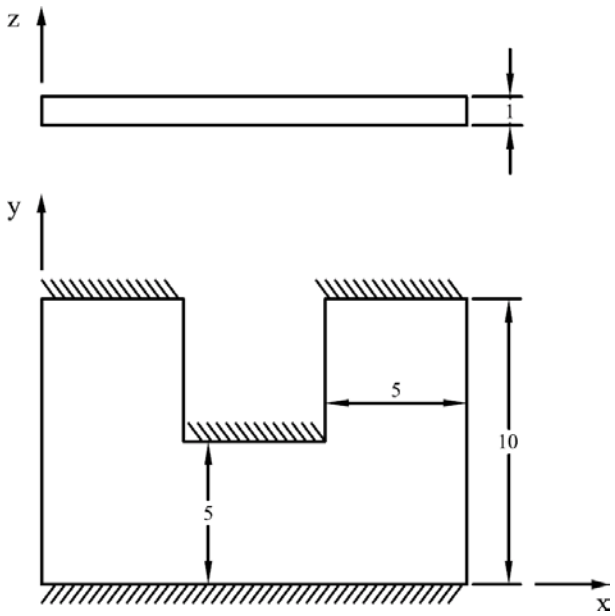


Figura 17 – Placa em “U” moderadamente espessa.

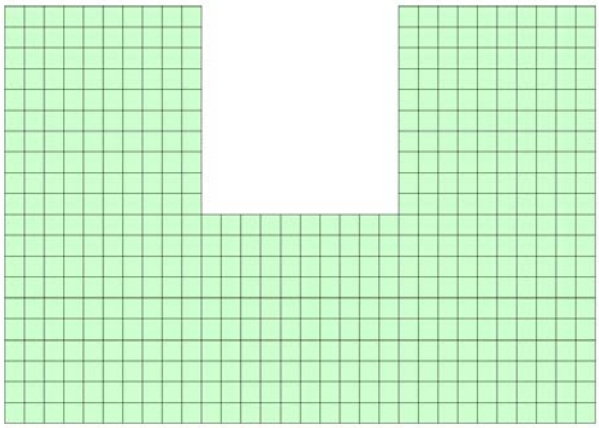
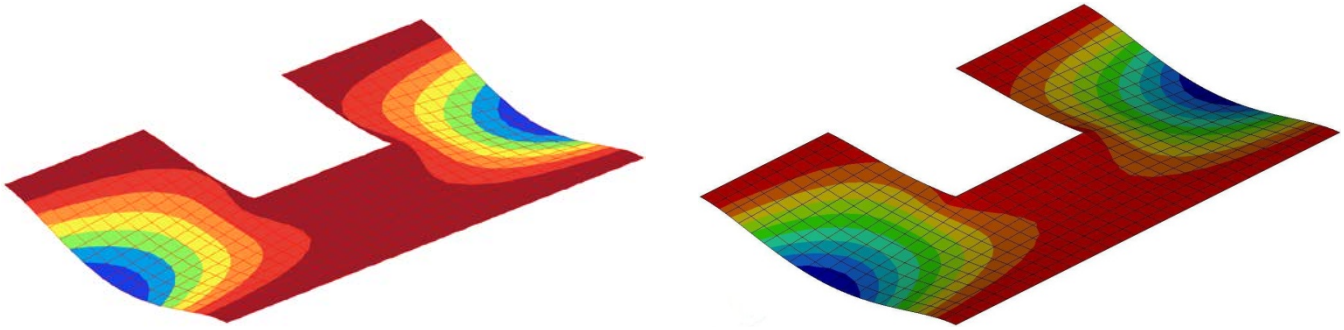


Figura 18 – Malha utilizada para análise da placa em “U”.



(a) INSANE – flecha máxima = -1,526 x 10^{-4} (b) ABAQUS – flecha máxima = -1,529 x 10^{-4}

Figura 19 – Configuração deformada da placa.

5.4 Discussão dos resultados

Uma análise dos resultados apresentados fornece uma boa compreensão do comportamento dos elementos implementados no INSANE. Como destaque pode-se citar os elementos BFS e Cowper da teoria de Kirchhoff e o elemento Q9 (teoria de Reissner-Mindlin) que possui uma rápida aproximação da solução analítica, independentemente da técnica aplicada (integração reduzida, integração seletiva ou deformação de cortante imposta).

Um aspecto importante a ser observado é a monotonicidade dos resultados. Pode-se destacar os elementos baseados na teoria de Kirchhoff que apresentam monotonicidade nos quatro casos estudados e os elementos Q4, Q8, Q9 e T6, baseados na teoria de Reissner-Mindlin, que também apresentaram monotonicidade em todos os casos estudados em todas as técnicas anti-bloqueio que lhes são aplicáveis. Os elementos MZC, CKZ e T6, quando apresentam monotonicidade, têm convergência por cima (os valores de deslocamentos obtidos são superiores ao valor analítico), por outro lado, os elementos Q4 e Q8 na maioria das vezes possuem convergência monotônica por baixo (os valores de deslocamentos obtidos são inferiores ao valor analítico).

Para análise de placas finas os elementos baseados na teoria de Kirchhoff (elementos específicos para placas finas) são em geral melhores do que os elementos baseados na teoria de Reissner-Mindlin.

Comparando-se as técnicas anti-bloqueio, a utilização da técnica de deformação de cortante imposta pode ser dita superior, por não apresentar modos espúrios de energia nula, tornando sua utilização mais confiável.

6 CONCLUSÕES

As simulações apresentadas validam as implementações realizadas e ilustram alguns dos diversos recursos disponibilizados. Com os elementos de Reissner-Mindlin e as técnicas anti-bloqueio, é possível analisar placas de diversas formas, espessuras e sob variadas condições de contorno e carregamento, tornando esta possibilidade mais eficaz do ponto de vista da utilização em aplicações práticas. Do ponto de vista de estudo do tema, entretanto, os elementos de Kirchhoff, apesar das limitações de utilização, guardam em si toda a história de desenvolvimento do MEF aplicado a placas, que pode e deve ser revisitada em favor do processo de ensino-aprendizagem da matéria.

Dentre os recursos, já disponíveis no sistema, não ilustrados aqui, merecem destaque: (1) a análise de placas compostas, como as de concreto armado, exemplificada em SALIBA (2007); (2) a análise dinâmica linear de placas, implementada por Fonseca (2008) e (3) a combinação dos elementos finitos de placas aqui apresentados com elementos de estado plano de tensões, gerando elementos de casca, realizada por AJEJE (2009). Também merecem destaque três outros recursos em fase de desenvolvimento: (1) a adaptação dos modelos fisicamente não lineares de plasticidade e fissuração distribuída – implementados por FUINA (2009) –, para a análise de placas; (2) a incorporação do modelo de descontinuidade forte embutida – implementado por WOLFF (2010) –, para estudo de propagação de fissuras em placas e (3) a implementação de modelos geometricamente não lineares para placas.

Todos esses recursos têm sido facilmente incorporados ao sistema demonstrando a eficiência do mesmo quanto a inclusão de novas implementações e atualizações para novas tecnologias. Esta eficiência que pode ser creditada às metodologias de desenvolvimento de software empregadas e, também, à linguagem de programação escolhida. Java é, hoje, a linguagem de programação mais usada no mundo, conforme TIOBE (2011).

Mais importante que a quantidade de recursos é a forma de distribuição do código fonte dos mesmos como software livre. Seguindo as premissas ideológicas preconizadas por Richard Stallman, criador do projeto GNU (DIBONA *et al*., 2009), o *INSANE* pode ser usado para qualquer propósito,

modificado para adaptá-lo a necessidades específicas, redistribuído de forma gratuita ou não e distribuído com modificações, beneficiando a comunidade com as melhorias.

7 AGRADECIMENTOS

Os autores agradecem à FAPEMIG (Fundação de Amparo à Pesquisa do Estado de Minas Gerais) pelo apoio financeiro.

8 REFERÊNCIAS

AJEJE, F. H. **Abordagem Orientada a Objetos para Implementação Computacional de Elementos Finitos de Cascas Planos**. 2009, 140 p. Dissertação (Mestrado em Engenharia de Estruturas) – Universidade Federal de Minas Gerais, Belo Horizonte.

BOGNER, F. K.; FOX, R. L.; SCHMIT, L. A.The generation of interelement compatible stiffness and mass matrices by the use of interpolation formulae. **Proc. Conf. Matrix Methods in Struct. Mech**. 1965.

CHEUNG, Y. K.; KING, I. P.; ZIENKIEWICZ, O. C. Slab bridges with arbitrary shape and support condition: a general method of analysis based on finite elements. **Proc. Inst. Civil Engng.**, Vol. 40, pp. 9-36, 1968.

COWPER, G. R., KOSKO, E., LINDBERG, G. M. e Olson, M. D. **A high precision triangular plate-bending element.** Report LR-514, National Aeronautical Establishment, National Research Council of Canadian, Ottawa, 1968.

DIBONA, C.; OCKMAN, S.; STONE, M. **Open Sources: Voices from the Open Source Revolution.** Sebastopol: Editora O'Reilly, 1999.

FONSECA, F. T. **Sistema Computacional para Análise Dinâmica Geometricamente Não-Linear através do Método dos Elementos Finitos**. 2008, 223p. Dissertação (Mestrado em Engenharia de Estruturas) – Universidade Federal de Minas Gerais, Belo Horizonte.

FUINA, J. S. **Formulações de Modelos Constitutivos de Microplanos para Contínuos Generalizados**. 2009, 283 p. Tese (Doutorado em Engenharia de Estruturas) – Universidade Federal de Minas Gerais, Belo Horizonte.

GAMMA, E.; HELM, R.; JONHSON, R.; VLISSIDES, J. **Design Patterns – elements of reusable object-oriented software**, Nova Jersey: Editora Addison Wesley, 1995. 395 p.

HUGHES, T. J. R. e COHEN, M. The "Heterosis" finite element for plate bending. **Computers and Structures**, Vol. 9, pp. 445-450, 1978.

LARMAN, C. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos**, Porto Alegre: Editora Bookman. 2000. 492 p.

MELOSH, R. J. A stiffness matrix for the analysis of thin plates in bending. **Journal of Aerospace Science**, Vol. 28, pp. 34-42, 1961.

OÑATE, E. **Cálculo de Estructuras por el Método de Elementos Finitos - Análisis Estático Lineal.** Barcelona: Editora Centro Internacional de Métodos Numéricos en Ingeniería, 1995. 838 p.

PITANGUEIRA, R. L. S. *et al*. Insane - versão 2.0, Universidade Federal de Alagoas, 2008, **XXVII Latin American Congress on Computational Methods in Engineering**, Maceió, 2008, p. 1-20.

RAY, E. T. **Aprendendo XML.** Rio de Janeiro: Editora Campus, 2000. 372 p.

SALIBA, S. S. **Implementação Computacional e Análise Crítica de Elementos Finitos de Placas**. 2007, 217 p. Dissertação (Mestrado em Engenharia de Estruturas) – Universidade Federal de Minas Gerais, Belo Horizonte.

SORIANO, H. L. **Método de Elementos Finitos em Análise de Estruturas**. São Paulo: EDUSP, 2003. 589 p.

TIMOSHENKO, S.;WOINOWSKY-KRIEGER, S. **Theory of Plates and Shells**. Nova York: Editora McGraw-Hill, 1959. 580 p.

TIOBE. <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>. Acesso em: 26 abril 2011.

WEAVER, Jr., W.; JOHNSTON, P. R. **Finite Elements for Structural Analysis.** Nova Jersey: Editora Prentice-Hall, 1984. 403 p.

ZIENKIEWICZ, O. C. e CHEUNG,Y. K. The finite element method for analysis of elastic isotropic and isotropic slabs. **Proc. Inst. Civ. Engng.**, Vol. 28, pp. 471-488, 1964.

WOLFF, K. P. **Implementação Computacional de um Modelo de Fissuração para o Concreto baseado no Método dos Elementos Finitos Estendido – XFEM.** 2010, 134 p. Dissertação (Mestrado em Engenharia de Estruturas) – Universidade Federal de Minas Gerais, Belo Horizonte.