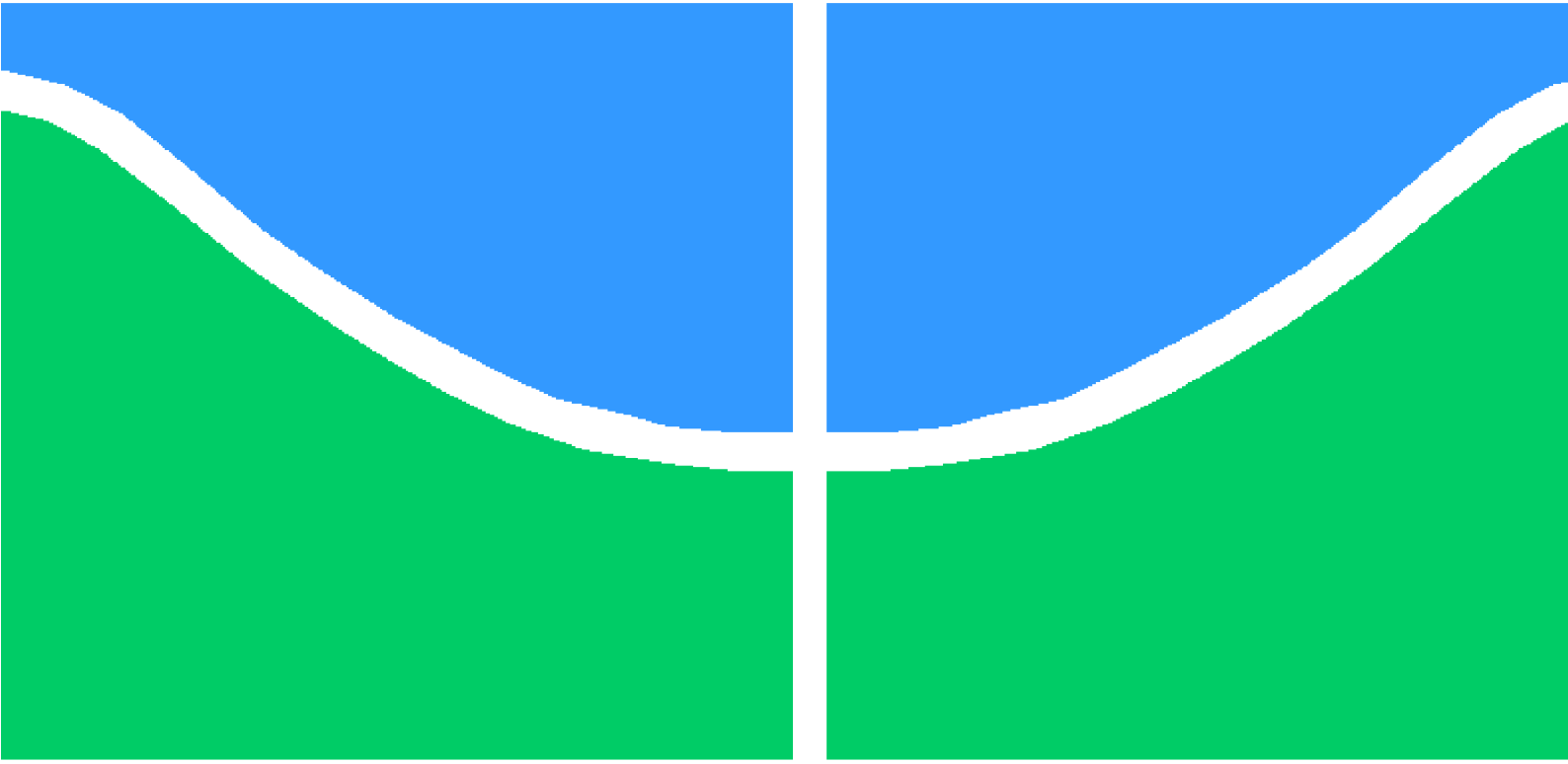


PROJETO DE GRADUAÇÃO

Implementação e Análise do Elemento F-bar em subrotina UEL

Por,

Lucas Lopes Martins

Brasília, 6 de dezembro de 2017

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA MECÂNICA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Departamento de Engenharia Mecânica

PROJETO DE GRADUAÇÃO

Implementação e Análise do Elemento F-bar em subrotina UEL

Por,

Lucas Lopes Martins

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro Mecânico

Banca Examinadora

Prof. Thiago Doca _____

Prof. Lucival Malcher _____

Prof. Fábio Comes de Castro _____

Brasília 6 de dezembro de 2017

RESUMO

Neste trabalho foi apresentado o processo de implementação do elemento F-bar utilizando uma rotina UEL e o programa Abaqus. A formulação do elemento F-bar foi detalhada, bem como a estrutura da UEL. Para a validação e verificação do desempenho do F-BAR, quatro análises bidimensionais foram realizadas: (i) compressão pura, (ii) tração pura, (iii) cisalhamento e (iv) contato. Em todas as simulações, as soluções obtidas com o elemento F-bar foram comparadas com os resultados obtidos por outros dois elementos do próprio programa: elemento quadrilateral de quatro nós de integração plena -CP4E- e o elemento quadrilateral de quatro nós híbrido -CPE4H. Foi observado boa concordância entre os resultados obtidos pelo elemento híbrido e F-BAR nas quatro análises, apesar do elemento híbrido ter apresentado instabilidade e não convergência no problema de contato, quando malhas ligeiramente maiores e menores (em relação à malha que produziu o resultado) foram utilizadas. Em geral, nos problemas bidimensionais abordados neste relatório, o elemento F-bar não apresentou grandes vantagens que justifiquem sua utilização. É esperado que ele se mostre mais vantajoso quando empregado em problemas tridimensionais. Foi verificado também a impossibilidade de visualização de todas as variáveis de saída (e.g., tensão e área de contato) calculadas nas simulações com o elemento F-bar através da ferramenta de pós-processamento do Abaqus, necessitando então a utilização de uma ferramenta externa para leitura e visualização dessas informações.

Palavras-chaves: UEL, F-BAR, Abaqus.

ABSTRACT

This report follows the work of implementing the F-bar element through an UEL routine for its usage in Abaqus. The formulation of the F-bar element has been detailed as well as the UEL structure. For its validation of this new element three bidimensional analysis were performed (i) pure compression, (ii) pure traction, (iii) shear test and (iv) contact analysis. All four analysis results were compared to results obtained using Abaqus standard elements (4-node quadrilateral with full integration -CP4E and 4-node quadrilateral hybrid -CPE4H). Even though the hybrid element reported instability and non-convergence analyzing the contact problem, when using different size's mesh, it was noted great similarity between the four analysis results from the hybrid and the F-bar element. In general, at the bidimensional problems developed by this study the usage of the F-bar did not showed great advantages that would justify its usage. It is believed; however, that the element would be more effective when dealing with tridimensional problems. It was also verified the impossibility of visualizing all the output variables (e.g., stress or contact area) calculated at the simulations with the F-bar element through Abaqus post-processing tool, thus an external tool is required to further visualize this information.

Key-words: UEL, F-BAR, Abaqus.

SUMÁRIO

1 INTRODUÇÃO 1

1.1 Introdução 1

1.2 Objetivos 2

1.3 Organização do Relatório 3

2 FUNDAMENTAÇÃO TEÓRICA 4

2.1 Método dos Elementos Finitos 4

2.1.1 Elemento Finito Quadrilateral 4

2.1.2 Funções de Forma e Famílias de Elementos 10

2.2 Mecanismo de Deformação 13

2.2.1 Gradiente de Deformação F 16

2.2.2 Mudanças no volume e o determinante de F 17

2.2.3 Partições Isocórica e Volumétrica do Gradiente de Deformação F 18

2.3 Elemento F-bar 19

2.3.1 O vetor de força interna 21

2.3.2 Linearização consistente e a matriz de rigidez tangente `K` 22

2.3.3 Adequação do elemento para o estado plano de deformação 23

2.4 Modelos de Hiperelasticidade 24

2.4.1 Conceitos básicos de hiperelasticidade 24

2.4.2 Modelos de Mooney-Rivlin e Neo-Hookean 25

2.5 Modelo Elastoplástico 26

2.5.1 Encruamento Isotrópico Linear 27

3 MODELAGEM NUMÉRICA 28

3.1 Tecnologia de Elementos do Abaqus 28

3.2 Rotina UEL 32

3.2.1 Funcionamento da UEL 32

3.2.1.1 Subrotinas UEL e modelo material constitutivo 35

3.2.2 Implementação da UEL 37

4 ANÁLISE NUMÉRICA 40

4.1 Análise I - Compressão Pura 40
4.2 Análise II - Tração Pura 43
4.3 Análise III - Cisalhamento 45
4.4 Análise IV - Compressão com Contato 46

5 CONCLUSÕES 51

REFERÊNCIAS 53

APÊNDICE A MÓDULO TANGENTE ESPACIAL 56

APÊNDICE B OBTENÇÃO DO PRIMEIRO TENSOR DE PIOLA-KIRCHHOFF 57

APÊNDICE C SUBROTINA UEL 59

APÊNDICE D EXEMPLOS DE ARQUIVOS *INPUT FILES* 81

LISTA DE FIGURAS

Figura 1 – Elemento finito quadrilateral de quatro nós com geometria arbitrária . 5
Figura 2 – Sistema de coordenadas locais 6
Figura 3 – Localização dos pontos de Gauss no sistema de coordenadas locais . . . 10
Figura 4 – Exemplo de elemento finito de 5 nós arbitrariamente distribuídos (Adaptado) 11
Figura 5 – Exemplos de elementos bidimensionais da família Lagrangeana (Adaptado) 12
Figura 6 – Critério de seleção dos termos do triângulo de Pascal para elementos bidimensionais da família Lagrangeana (Adaptado) 12
Figura 7 – Exemplos de elementos bidimensionais da família *Serendipity* (Adaptado) 13
Figura 8 – Critério de seleção dos termos do triângulo de Pascal para elementos bidimensionais da família *Serendipity* (Adaptado) 13
Figura 9 – Deformação do corpo β 14
Figura 10 – Deformação rígida (Adaptado) 15
Figura 11 – Movimento de β 16
Figura 12 – Gradiente de Deformação 16
Figura 13 – Determinante de Gradiente de Deformação 17
Figura 14 – Elemento F-bar quadrilateral de quatro nós e hexaédrico de oito nós . . 20
Figura 15 – Encruamento isotrópico 27
Figura 16 – Famílias de elementos comumente utilizadas (Adaptado) 29
Figura 17 – Efeito de *Hourglassing* em um elemento de integração reduzida 30
Figura 18 – Mudança de forma de um corpo sob flexão pura em uma situação ideal 31
Figura 19 – Efeitos de *Hourglassing* e *Shear Locking* em elementos quadrilaterais . 31
Figura 20 – Esquema de funcioamento da UEL 34
Figura 21 – Modificação na linha de comando de definição do tipo de elemento no *input file* 38
Figura 22 – Exemplo de criação de uma malha *dummy* no *input file* 38
Figura 23 – Chamada da UEL pelo prompt de comando do Abaqus 39
Figura 24 – Condições de contorno para problema de compressão 40
Figura 25 – Deslocamento do elemento finito híbrido na direção x (Compressão) . . 42
Figura 26 – Deslocamento do elemento finito híbrido na direção y (Compressão) . . 42

Figura 27 – Deslocamento do elemento finito híbrido nas direções x e y (Compressão 100%) 42
Figura 28 – Deslocamento do elemento F-bar nas direções x e y (Compressão 70%) 43
Figura 29 – Deslocamento do elemento F-bar nas direções x e y (Compressão 100%) 43
Figura 30 – Deslocamento do elemento finito híbrido na direção x (Tração) 44
Figura 31 – Deslocamento do elemento finito híbrido na direção y (Tração) 44
Figura 32 – Condições de contorno da análise de cisalhamento 45
Figura 33 – Elemento F-bar na análise de cisalhamento 46
Figura 34 – Modelagem do problema de contato 46
Figura 35 – Deformação plástica para o elemento finito híbrido (Malha 15x15 elementos) 48
Figura 36 – Deformação plástica no exemplo do Manual Elfen 49
Figura 37 – Logaritmo da deformação para o elemento F-bar 49
Figura 38 – Logaritmo da deformação para o elemento híbrido 50

LISTA DE TABELAS

Tabela 1 – Valores de compressão das simulações para cada elemento 41
Tabela 2 – Valores de tração aplicados para cada elemento 44
Tabela 3 – Propriedades do Aço SAE 1020 47
Tabela 4 – Curva de Encruamento 47
Tabela 5 – Propriedades da Modelagem do Contato 47

Lista de símbolos

a	Vetor de deslocamentos nodais
f	Vetor de forças nodais
h	Espessura do elemento finito
x	Coordenadas globais
p	Número de graus de liberdade do elemento
m	Número de direções consideradas
n	Número de nós do elemento
s	Coordenadas locais
N	Função de forma (Função interpoladora)
J	Matriz jacobiana
u	Campo de deslocamentos
F	Campo de deformações
$\bar{F}$	Campo de deformações do elemento F-bar
L	Operador diferencial
B	Matriz de deformação
σ	Tensor de tensões de Cauchy
D	Matriz de elasticidade
κ	Matriz de rigidez
k	Número de nós em um lado do elemento finito
β	Corpo de deformação
Ω	Região ocupada pelo corpo β
φ	Função de mapeamento das partículas do corpo β

$\mathtt{F}^{EXT}$	Vetor de força externa
$\mathtt{F}^{INT}$	Vetor de força interna do elemento
R	Resíduo (Forças fora de equilíbrio)
U	Vetor de deslocamento global
$\mathtt{K}$	Matriz de rigidez tangente
G	Operador gradiente espacial
$[a]$	Módulo tangente espacial
$[q]$	Matriz de forma do tensor de quarta ordem
A	Módulo tangente material
ψ	Função de energia de deformação
ρ	Massa específica
τ_{kh}	Tensor de tensão de Kirchhoff
$\mathtt{B}$	Tensor de deformação esquerdo de Cauchy-Green
I_1^*, I_2^*	Invariantes principais do tensor isocórico
K	Módulo de incompressibilidade
μ	Módulo de cisalhamento
E	Módulo de elasticidade
σ_y	Tensão de escoamento
ε	Deformação
Φ	Função de escoamento
H	Módulo de encruamento isotrópico linear
$\dot{\gamma}$	Multiplicador plástico
$\mathtt{E}$	Regime Elástico
$\mathtt{P}$	Regime Plástico
$\mathtt{D}$	Elongamento plástico

1 INTRODUÇÃO

1.1 INTRODUÇÃO

Os problemas de engenharia podem ser estudados analiticamente, numericamente ou experimentalmente. Entretanto, nem sempre é possível fazer-se uso de soluções analíticas, devido à complexidade matemática envolvida na sua modelagem. Por outro lado, a utilização do método experimental mostra-se bastante interessante no sentido de validar na prática resultados obtidos em análises numéricas, porém despendem custos elevados para sua implementação. Tendo isso em vista, o emprego de métodos numéricos torna-se relevante, justificado também pelo crescimento exponencial da capacidade de processamento dos computadores, possibilitando a resolução de problemas de grande complexidade à baixo custo e com rapidez.

Dentro os métodos numéricos utilizados para análise de problemas mecânicos, o Método dos Elementos Finitos (MEF) se destaca. Esse método, baseado no Princípio dos Trabalhos Virtuais e em uma metodologia de escolha conveniente de funções interpoladoras, é comumente aplicado nos diversos campos de engenharia e consiste na fundamentação teórica de diversos *softwares*, por exemplo, Abaqus, ANSYS e NASTRAN (FILHO, 2014). Neste trabalho o *software* Abaqus será utilizado para a análise numérica.

No passado, grande parte dos utilizadores do MEF estavam envolvidos no processo de programação do *software*, entretanto verifica-se atualmente que a maioria dos usuários atenta-se apenas com a utilização do programa e interpretação do resultado obtido (AZEVEDO, 2003). Perante um problema e dispondo de uma interface intuitiva, o usuário obtém uma resposta, apesar do desconhecimento do código computacional e características dos modelos empregados.

Uma vez que o usuário não possui esses conhecimentos, nem base para sua compreensão, costuma-se aceitar os resultados adquiridos, quaisquer que sejam, considerando a qualidade do programa utilizado. Esse procedimento pode levar a escolhas inadequadas nas condições de contorno introduzidas e a não correspondência entre modelo e estrutura analisada. Na ausência de uma comparação dos resultados provenientes do MEF com os oriundos de outros modelos, existe o sério risco de a segurança de uma estrutura ser justificada com base em cálculos inadequados (AZEVEDO, 2003). Desta forma, o estudo dos fundamentos do MEF para uma utilização mais segura desses tipos de programas é justificada.

É um fato conhecido que o desempenho de análises a partir de elementos finitos de baixa ordem mostra-se bastante ineficiente à medida que se aproxima do limite de incompressibilidade. Alguns problemas de engenharia em que a incompressibilidade desempenha papel fundamental incluem análise de materiais elastoméricos, comumente modelados como materiais hiperelásticos incompressíveis, bem como simulações elastoplásticas, onde as deformações plásticas são dominantes e supõe-se preservação de volume, observadas em modelos de plasticidade em metais. Para essas situações, o travamento de volumétrico residual, em outras palavras, soluções com rigidez excessiva, são regularmente obtidas como consequência da incapacidade de interpolações polinomiais de baixa ordem representarem campos de deslocamentos que mantêm o volume (NETO; PERIĆ; OWEN, 2008).

Essa deficiência pode ser remediada utilizando-se elementos de ordens superiores, porém, devido ao seu menor custo computacional, elementos de baixa ordem são preferidos em cálculos de larga escala. Além disso, em problemas mais realísticos, que envolvem fenômenos mais complexos, como contato com atrito e deformações finitas, esse tipo de elemento é preferido, por ser mais simples e robusto.

Para problemas que envolvem grandes deformações, como em análises de materiais de borracha e processos de conformação de metais, é comum obter-se boas soluções apenas com a implementação de rotinas que utilizem mecanismos de refinamento de malha adaptativo (NETO; PERIĆ; OWEN, 2008). Entretanto, uma única formulação não costuma ser suficientemente robusta para produzir resultados ótimos para uma grande diversidade de aplicações.

Neste sentido, para tentar resolver esses problemas diversas abordagens foram propostas. Dentre elas, alguns exemplos presentes na literatura são (SIMO; TAYLOR; PISTER, 1985) com o *mixed variational method*, (SUSSMAN; BATHE, 1987) através da formulação *mixed u/p* (deslocamento-pressão), (HUGHES; TAYLOR; SACKMAN, 1975) e (MORAN; ORTIZ; SHIH, 1990) com a metodologia *B-bar* e (NETO et al., 1996) por meio da metodologia F-bar, que será a metodologia abordada neste trabalho.

1.2 OBJETIVOS

Tendo em vista as limitações no uso de elementos finitos convencionais em análises que envolvem, por exemplo, deformações finitas e/ou contato com atrito, a metodologia F-bar foi escolhida para contornar essas deficiências. Conforme mostrado em (NETO; PERIĆ; OWEN, 2008), essa metodologia mostrou-se capaz de evitar o travamento volumétrico e bastante adequada para a captação de fenômenos de deformação. Além disso, sua implementação pode ser realizada para qualquer modelo material constitutivo e também pode ser usada em conjunto com mecanismos de malha adaptativa.

Atentando-se aos benefícios que o elemento F-bar possui e levando-se em conta que o programa utilizado (Abaqus) não possui esse elemento, o presente trabalho tem os

seguintes objetivos:

- Implementação do elemento F-bar no Abaqus através de uma rotina UEL;
- Validação e comparação do desempenho do elemento F-bar com outros elementos disponíveis no Abaqus em análises numéricas.

1.3 ORGANIZAÇÃO DO RELATÓRIO

Buscando alcançar os objetivos propostos, o trabalho foi dividido em cinco capítulos. No primeiro capítulo faz-se a introdução e descrição dos objetivos do trabalho.

O segundo capítulo contém a fundamentação teórica acerca do MEF, mecanismos de deformação, descrição da metodologia F-bar e dos modelos constitutivos materiais utilizados neste trabalho. No terceiro capítulo está presente uma breve introdução sobre o pacote comercial de simulação Abaqus e, de forma bem detalhada, o funcionamento e implementação da rotina UEL.

No capítulo quatro é descrito as quatro análises realizadas para validação e comparação do elemento F-bar com os outros elementos do Abaqus. Já no quinto e último capítulo, encontram-se as conclusões do trabalho e sugestões para estudos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 MÉTODO DOS ELEMENTOS FINITOS

O Método dos Elementos Finitos tem como objetivo a determinação do estado de tensão e de deformação de um sólido com geometria arbitrária que está sujeito à ações externas. Antes do surgimento do MEF, a análise dos meios contínuos era efetuada através da resolução de sistemas de equações de derivadas parciais que descrevem o fenômeno, levando-se em conta as condições de contorno. Devido à sua complexidade, esses procedimentos eram aplicados somente a meios contínuos e homogêneos e de geometria simples.

Para tentar superar essas limitações, alguns métodos foram aplicados, como o método das diferenças finitas, substituindo derivadas exatas por derivadas aproximadas, e métodos de relaxação, baseados na diminuição sucessiva de um conjunto de resíduos (TIMOSHENKO; GOODIER, 1988). Devido à morosidade associada à aplicação de qualquer um desses métodos, era interessante a substituição do problema real por um análogo, permitindo recorrer-se à resultados publicados em tabelas ou ábacos (COOK et al., 2002).

O MEF foi desenvolvido a partir da década de 1960, o método que será apresentado a seguir baseia-se no método dos deslocamentos e na discretização de uma estrutura em sub-estruturas. Cada sub-estrutura é designada como um elemento finito e possui comportamento conhecido, de forma que o comportamento geral é definido como a soma das partes. Cada elementos finito possui n nós, considerando explicitamente apenas os deslocamentos generalizados nesses nós. Os deslocamentos nos restantes dos pontos do elemento finito são obtidos por interpolação dos deslocamentos dos nós.

2.1.1 ELEMENTO FINITO QUADRILATERAL

Por conveniência e concordância com o tipo de elemento que será utilizado neste trabalho, será desenvolvida a formulação do **elemento finito bidimensional quadrilateral de quatro nós**. Para isso, serão utilizados e abordados alguns recursos como: processos de substituição de variáveis, quadratura de Gauss, funções de forma e o Princípio dos Trabalhos Virtuais. A Fig. 1 apresenta esse tipo de elemento com geometria arbitrária.

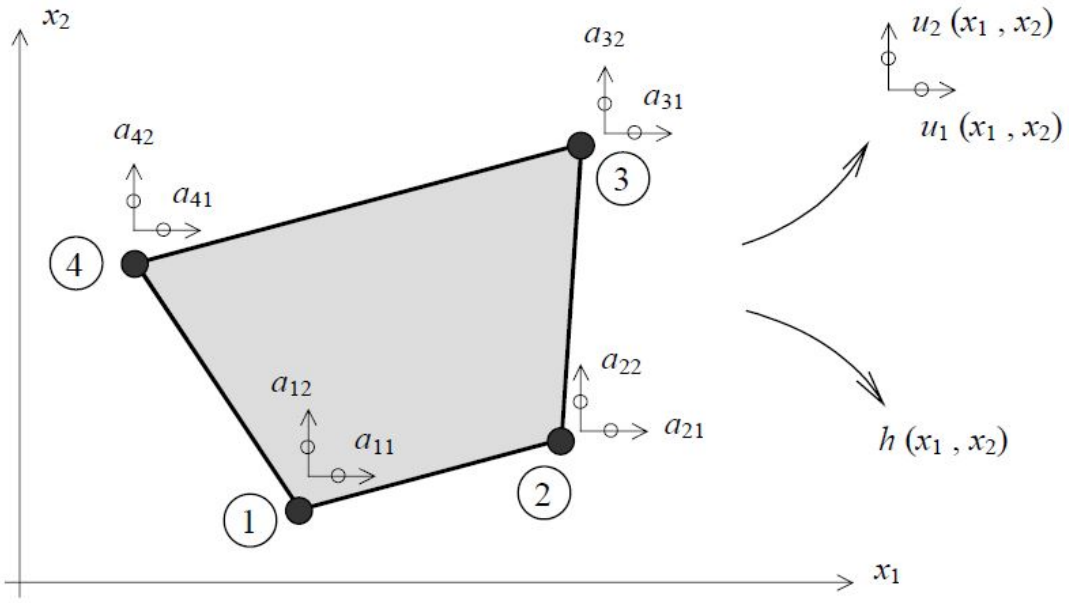


Figura 1 – Elemento finito quadrilateral de quatro nós com geometria arbitrária[1]

Define-se $\underline{a}$ como o vetor que contém os deslocamentos nodais e $\underline{f}$ o vetor das forças nodais correspondentes às ações externas.

$$\underset{(p\times 1)}{\underline{a}} = \begin{bmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{41} \\ x_{42} \end{bmatrix} \qquad \underset{(p\times 1)}{\underline{f}} = \begin{bmatrix} f_{11} \\ f_{12} \\ \vdots \\ f_{41} \\ f_{42} \end{bmatrix} \tag{2.1}$$

a espessura h do elemento finito laminar representado na Fig. 1 também pode ser uma função de x_1 e x_2.

O número de graus de liberdade de elemento é definido como $p = n \times m$, em que m é o número de direção consideradas (neste caso, $m = 2$). No referencial global, a matriz das coordenadas dos nós do elemento é:

$$\underset{(n\times m)}{\underline{\bar{x}}} = \begin{bmatrix} \bar{x}_{11} & \bar{x}_{12} \\ \bar{x}_{21} & \bar{x}_{22} \\ \vdots & \vdots \\ \bar{x}_{41} & \bar{x}_{42} \end{bmatrix} \tag{2.2}$$

Com o intuito de facilitar a programação em computador e possibilitar a utilização do Quadratura de Gauss para integração, faz-se uma substituição de variáveis de forma

[1] (AZEVEDO, 2003)

a alterar o domínio de integração para o intervalo [-1,1] conveniente. Desta forma, temos as novas coordenadas (Eq. (2.3)) :

$$\begin{cases} x_1 \rightarrow x_1(s_1, s_2) \\ x_2 \rightarrow x_2(s_1, s_2) \end{cases} \tag{2.3}$$

A passagem do sistema de coordenadas s para o sistema x é efetuada por interpolação através de funções de forma que serão melhores explicadas na Seção 2.1.2. Cada uma das componentes de x são interpoladas separadamente com base nessas funções de forma e no deslocamentos dos nós. A Eq. (2.4) representa uma interpolação para uma das componentes de x:

$$x_1(s_1, s_2) = N_1(s_1, s_2)\bar{x}_{11} + N_2(s_1, s_2)\bar{x}_{21} + \ldots + N_4(s_1, s_2)\bar{x}_{41} \tag{2.4}$$

Na Fig. 2 apresenta-se o novo sistema de coordenadas locais (em s) e o novo domínio de integração correspondente.

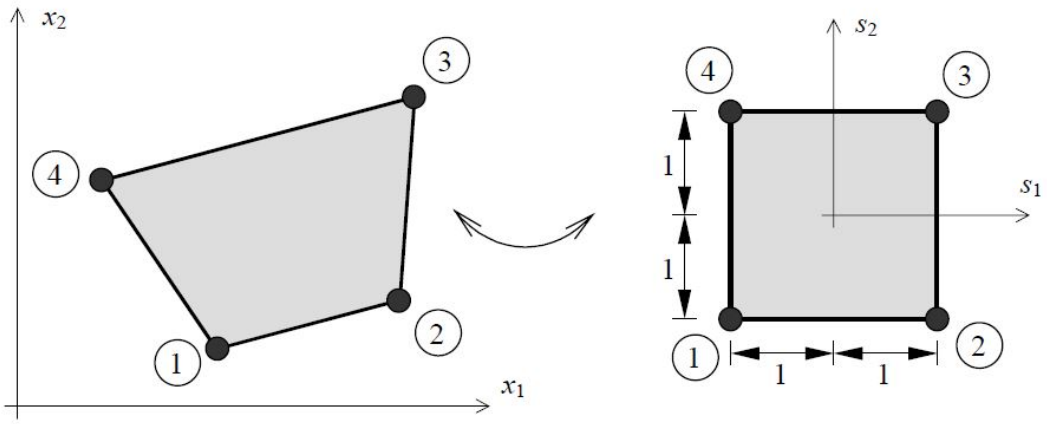


Figura 2 – Sistema de coordenadas locais[2]

Desta forma, os novos valores nodais das coordenadas s_1, s_2 são:

$$\underset{(n\times m)}{\bar{\underline{s}}} = \begin{bmatrix} \bar{s}_{11} & \bar{s}_{12} \\ \bar{s}_{21} & \bar{s}_{22} \\ \bar{s}_{31} & \bar{s}_{32} \\ \bar{s}_{41} & \bar{s}_{42} \end{bmatrix} = \begin{bmatrix} -1 & -1 \\ +1 & -1 \\ +1 & +1 \\ -1 & +1 \end{bmatrix} \tag{2.5}$$

Supõe-se que este elemento pertence à família Lagrangeana, desta forma as funções de forma estão de acordo com a Eq. (2.6).

[2](AZEVEDO, 2003)

$$\begin{cases} N_1(s_1, s_2) = (1 - s_1)(1 - s_2)/4 \\ N_2(s_1, s_2) = (1 + s_1)(1 - s_2)/4 \\ N_3(s_1, s_2) = (1 + s_1)(1 + s_2)/4 \\ N_4(s_1, s_2) = (1 - s_1)(1 + s_2)/4 \end{cases} \tag{2.6}$$

A interpolação das coordenadas cartesianas é realizada segundo a Eq. (2.7):

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \bar{x}_{11} & \bar{x}_{21} & \bar{x}_{31} & \bar{x}_{41} \\ \bar{x}_{12} & \bar{x}_{22} & \bar{x}_{32} & \bar{x}_{42} \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{bmatrix} \tag{2.7}$$

correspondente a:

$$\underset{(m\times 1)}{\underline{x}} = \underset{(m\times n)}{\bar{\underline{x}}^{\mathrm{T}}} \underset{(n\times 1)}{\underline{N}_{\mathrm{V}}}$$

A matriz jacobiana J correspondente à transformação descrita pela Eq. (2.3) é definida como:

$$J = \begin{bmatrix} \frac{\partial x_1}{\partial s_1} & \frac{\partial x_1}{\partial s_2} \\ \frac{\partial x_2}{\partial s_1} & \frac{\partial x_2}{\partial s_2} \end{bmatrix} \tag{2.8}$$

Define-se agora o campo de deslocamentos u como o produto entre as funções de forma N e os deslocamentos nodais a (Eq. (2.9)):

$$\underset{(m\times 1)}{\underline{u}} = \underset{(m\times p)}{\underline{N}} \underset{(p\times 1)}{\underline{a}} \tag{2.9}$$

A interpolação do campo de deslocamentos da Eq. (2.9) é efetuada de forma coincidente à interpolação realizada para as coordenadas cartesianas na Eq. (2.4). Como a interpolação dos deslocamentos nodais e das coordenadas cartesianas coincidem, a formulação deste elemento é dita isoparamétrica.

O campo de deformações F é definido como

$$F = \frac{du}{dx} \tag{2.10}$$

como os deslocamentos nodais a não são funções de x, a partir da Eq. (2.9) é possível reescrever Eq. (2.10):

$$F = \frac{dN}{dx}\, a = \underline{L}\, \underline{N}\, \underline{a} = \underline{B}\, \underline{a} \tag{2.11}$$

Na Eq. (2.11), defini-se L como o operador diferencial e a matriz de deformação B como o produto LN, da seguinte forma:

$$LN = \begin{bmatrix} \frac{\partial}{\partial x_1} & 0 \\ 0 & \frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} \end{bmatrix} \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{bmatrix} \tag{2.12}$$

$$B = \begin{bmatrix} \frac{\partial N_1}{\partial x_1} & 0 & \frac{\partial N_2}{\partial x_1} & 0 & \frac{\partial N_3}{\partial x_1} & 0 & \frac{\partial N_4}{\partial x_1} & 0 \\ 0 & \frac{\partial N_1}{\partial x_2} & 0 & \frac{\partial N_2}{\partial x_2} & 0 & \frac{\partial N_3}{\partial x_2} & 0 & \frac{\partial N_4}{\partial x_2} \\ \frac{\partial N_1}{\partial x_2} & \frac{\partial N_1}{\partial x_1} & \frac{\partial N_2}{\partial x_2} & \frac{\partial N_2}{\partial x_1} & \frac{\partial N_3}{\partial x_2} & \frac{\partial N_3}{\partial x_1} & \frac{\partial N_4}{\partial x_2} & \frac{\partial N_4}{\partial x_1} \end{bmatrix} \tag{2.13}$$

A matriz B depende das derivadas das funções de forma $\partial N_i/\partial x_j$, porém é necessário que seus elementos sejam funções apenas de s_1 e s_2, portanto, $\partial N_i/\partial x_j$ devem ser funções de s_1 e s_2. Considerando uma função de forma qualquer:

$$N_i = N_i(x_1(s_1, s_2), x_2(s_1, s_2))$$

pela regra da cadeia, temos que:

$$\begin{cases} \frac{\partial N_i}{\partial s_1} = \frac{\partial N_i}{\partial x_1}\frac{\partial x_1}{\partial s_1} + \frac{\partial N_i}{\partial x_2}\frac{\partial x_2}{\partial s_1} \\ \frac{\partial N_i}{\partial s_2} = \frac{\partial N_i}{\partial x_2}\frac{\partial x_1}{\partial s_2} + \frac{\partial N_i}{\partial x_2}\frac{\partial x_2}{\partial s_2} \end{cases}$$

essas expressões podem ser resumidas na forma matricial:

$$\frac{\partial \underline{N}}{\partial \underline{s}} = \frac{\partial \underline{N}}{\partial \underline{x}}\,\underline{J} \tag{2.14}$$

Multiplicando-se ambos os membros, à direita, da Eq. (2.14) por J^{-1} obtêm-se:

$$\frac{\partial \underline{N}}{\partial \underline{x}} = \frac{\partial \underline{N}}{\partial \underline{s}}\,\underline{J}^{-1} \tag{2.15}$$

$\partial \underline{N}/\partial \underline{s}$ é facilmente obtida pela derivação das funções de forma:

$$\frac{\partial \underline{N}}{\partial \underline{s}} = \begin{bmatrix} \frac{\partial N_1}{\partial s_1} & \frac{\partial N_1}{\partial s_2} \\ \frac{\partial N_2}{\partial s_1} & \frac{\partial N_2}{\partial s_2} \\ \frac{\partial N_3}{\partial s_1} & \frac{\partial N_3}{\partial s_2} \\ \frac{\partial N_4}{\partial s_1} & \frac{\partial N_4}{\partial s_2} \end{bmatrix} = \begin{bmatrix} (-1+s_2)/4 & (-1+s_1)/4 \\ (+1-s_2)/4 & (-1-s_1)/4 \\ (+1+s_2)/4 & (+1+s_1)/4 \\ (-1-s_2)/4 & (+1-s_1)/4 \end{bmatrix} \tag{2.16}$$

desta forma, os elementos da matriz $\partial \underline{N}/\partial \underline{x}$ dependem somente de s_1 e s_2:

$$\frac{\partial \underline{N}}{\partial \underline{x}} = \begin{bmatrix} \frac{\partial N_1}{\partial x_1} & \frac{\partial N_1}{\partial x_2} \\ \frac{\partial N_2}{\partial x_1} & \frac{\partial N_2}{\partial x_2} \\ \frac{\partial N_3}{\partial x_1} & \frac{\partial N_3}{\partial x_2} \\ \frac{\partial N_4}{\partial x_1} & \frac{\partial N_4}{\partial x_2} \end{bmatrix} \tag{2.17}$$

Considera-se agora um estado plano de tensão constituído pelo elemento finito formulado anteriormente e que existam apenas ações distribuídas por unidade de comprimento na periferia do elemento. Do Princípio dos Trabalhos Virtuais, estabelece-se que o trabalho realizado pelas tensões internas na deformação virtual do corpo é igual ao trabalho realizado pelas forças exteriores nos deslocamentos virtuais dos seus pontos de aplicação (COOK et al., 2002; ZIENKIEWICZ; TAYLOR, 1988). Em outras palavras, pode-se dizer que o trabalho interno de deformação é igual ao trabalho externo das forças aplicadas (Eq. (2.18)).

$$Trabalho\ Interno = Trabalho\ Externo$$

$$\int_V \delta \underline{F}^T \, \underline{\sigma} \, dV \; = \; \int_L \delta \underline{u}^T \, \underline{p} \, dL \tag{2.18}$$

Na Eq. (2.18), o prefixo δ designa que os deslocamentos ou deformações são virtuais e p são as ações exteriores distribuídas por unidade de comprimento e o tensor de tensões σ é:

$$\underline{\sigma} \; = \; \underline{D} \, \underline{F}$$

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{12} \end{bmatrix} = \underline{D} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \gamma_{12} \end{bmatrix} \tag{2.19}$$

ε corresponde à extensão e γ à distorção e D é a matriz de elasticidade.

Utilizando-se o Princípio dos Trabalhos Virtuais, é possível deduzir a matriz de rigidez κ e o vetor das forças nodais f, assim:

$$\kappa = \int_V \underline{F}^T \, \underline{D} \, \underline{B} \, dV \tag{2.20}$$

$$f = \int_L \underline{N}^T \, \underline{p} \, dV \tag{2.21}$$

como todos os elementos da matriz rigidez κ são funções de s_1 e s_2 é possível calcular cada um deles através da Quadratura de Gauss.

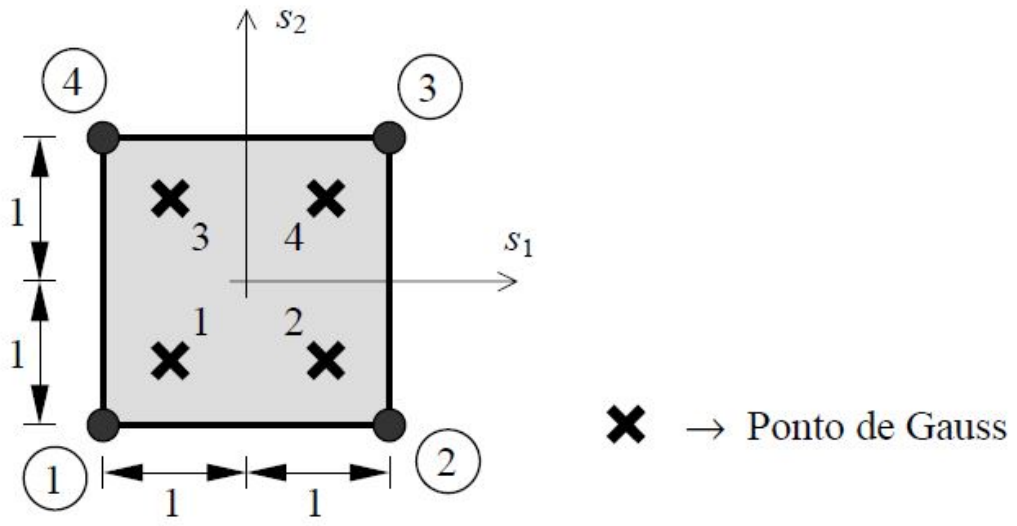


Figura 3 – Localização dos pontos de Gauss no sistema de coordenadas locais[3]

A partir da resolução das integrais κ e f é possível calcular a tensão e deformação de qualquer ponto do elemento finito, entretanto, apesar da formulação permitir esse cálculo em qualquer ponto, existe uma maior precisão se os pontos selecionados coincidirem com os pontos de Gauss (ZIENKIEWICZ; TAYLOR, 1988), conforme a Fig. 3. O procedimento geral está descrito abaixo:

(i) Selecionar o elemento finito e o ponto de Gauss em que se deseja conhecer o estado de tensão;

(ii) Calcular a matriz B no ponto selecionado;

(iii) Extrair para um vetor os deslocamentos dos nós do elemento finito estudado ($\underline{a}$);

(iv) Calcular o campo de deformações $\underline{F}$ através da Eq. (2.11);

(v) Calcular a matriz elasticidade $\underline{D}$;

(vi) Calcular o tensor de tensões σ.

Obtendo σ e F nos pontos de Gauss, é possível interpolar ou extrapolar para os nós do elemento.

2.1.2 FUNÇÕES DE FORMA E FAMÍLIAS DE ELEMENTOS

Nesta seção será discutida brevemente as características essenciais de uma função de forma (ou função interpoladora) arbitrária N_i e as duas principais famílias de elementos.

[3](AZEVEDO, 2003)

Seja $N = N(x)$, $\bar{x}_i$ a coordenada do nó i e i o número do nó. As características essenciais de uma função de forma N_i são:

- Valor unitário para $x = \bar{x}_i$;
- Valor nulo para $x \neq \bar{x}_i$;
- $\sum_{n=1}^{i} N_i = 1$.

Para funções de forma polinomiais é desejável manter o grau do polinômio o mais baixo possível (COOK et al., 2002).

Em um processo genérico para a determinação de uma função interpoladora **polinomial** de um elemento finito de n nós, arbitrariamente distribuídos (Fig. 4), é necessário selecionar os termos no triângulo de Pascal (ou na pirâmide de Pascal, para um caso tridimensional) de forma a se obter o menor grau possível do polinômio. Além disso, deve se dar preferência a termos de grau mais elevado na direção que apresenta o maior número de nós.

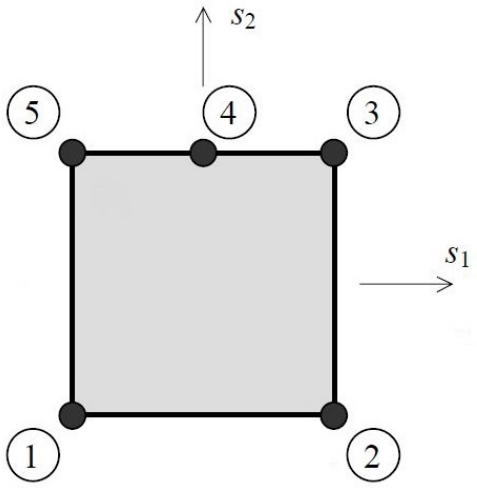


Figura 4 – Exemplo de elemento finito de 5 nós arbitrariamente distribuídos (Adaptado)[4]

Em uma distribuição arbitrária de nós, faz-se necessário selecionador adequadamente os termos do triângulo de Pascal para cada caso. Quando não existe critério óbvio para a escolha, deve-se analisar alternativas, já que para cada conjunto de termos selecionados surge uma formulação distinta para o elemento finito, desta forma, é necessário averiguar qual delas possui melhor precisão.

Segundo (ZIENKIEWICZ; TAYLOR, 1988), para situações comuns já existem formulações que conduzem bons resultados, conhecidas como família de elementos Lagrangeana e *Serendipity*.

[4](AZEVEDO, 2003)

Elementos bidimensionais da família Lagrangeana são quadriláteros de k^2 nós, em que k é o número de nós de um lado do elemento finito (Fig. 5).

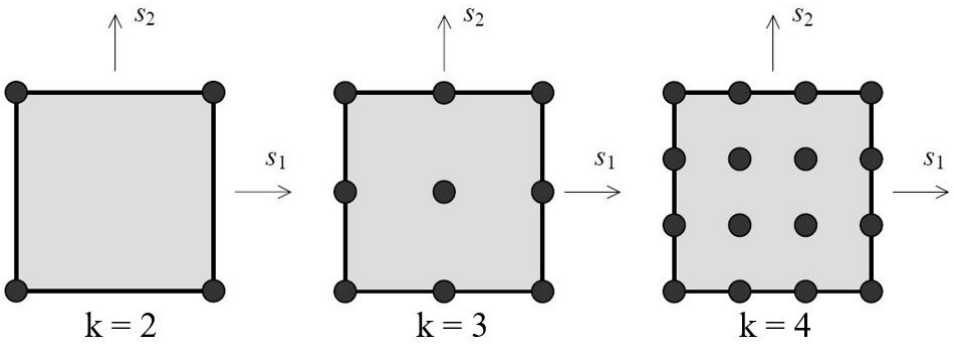


Figura 5 – Exemplos de elementos bidimensionais da família Lagrangeana (Adaptado)[5]

Para elementos dessa família a seleção dos termos do triângulo de Pascal deve seguir o critério da Fig. 6.

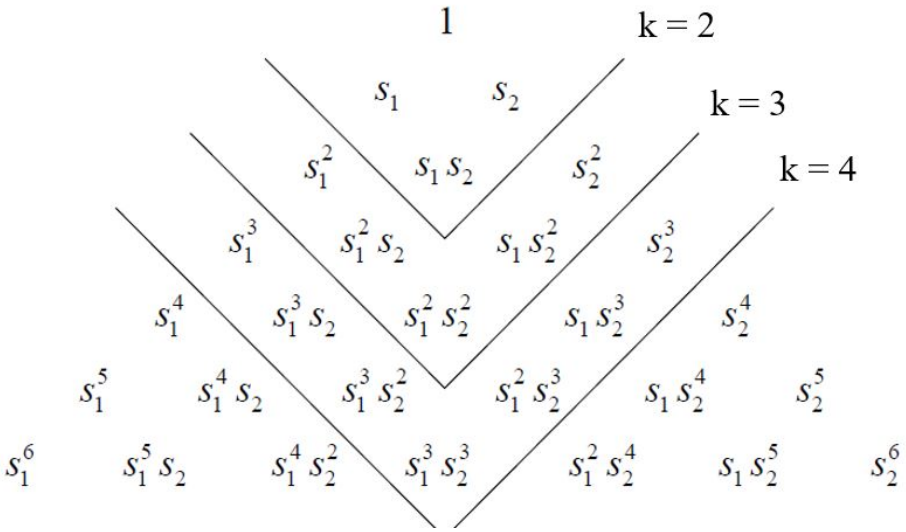


Figura 6 – Critério de seleção dos termos do triângulo de Pascal para elementos bidimensionais da família Lagrangeana (Adaptado)[6]

Para os elementos da família *Serendipity* temos os seguintes exemplos bidimensionais na Fig. 7. Neste caso, o número de nós é $4(k-1)$.

[5](AZEVEDO, 2003)
[6](AZEVEDO, 2003)

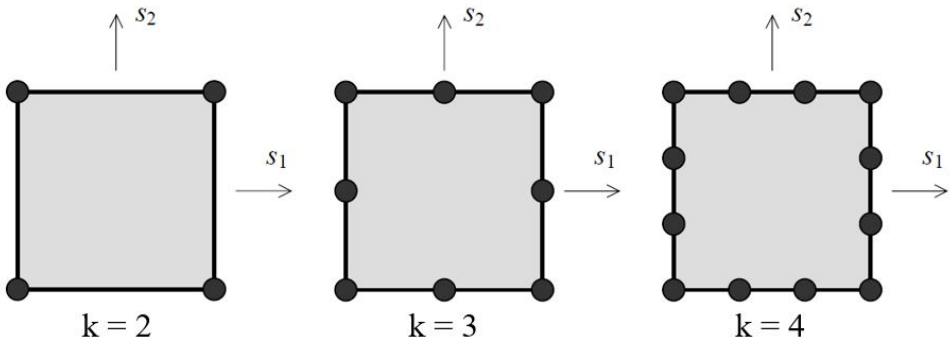


Figura 7 – Exemplos de elementos bidimensionais da família *Serendipity* (Adaptado)[7]

O critério de seleção dos termos de Pascal se dá segundo a Fig. 8.

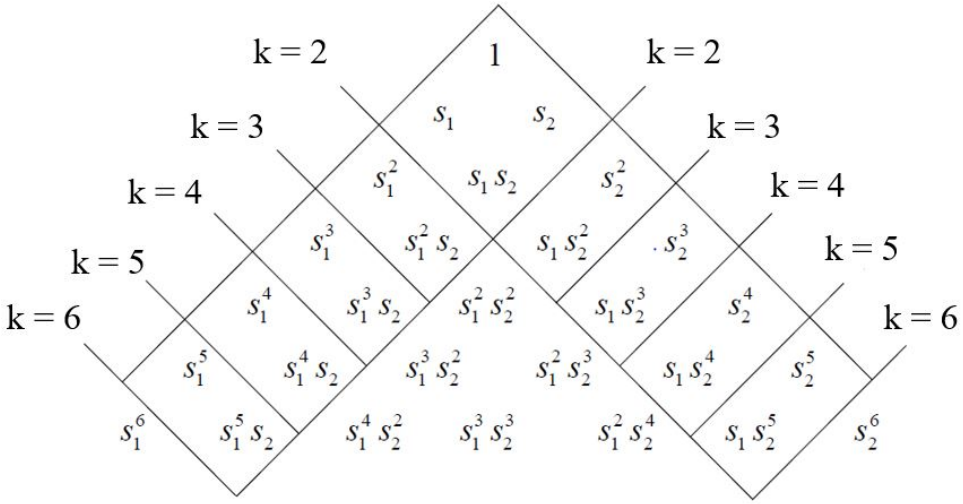


Figura 8 – Critério de seleção dos termos do triângulo de Pascal para elementos bidimensionais da família *Serendipity* (Adaptado)[8]

Para ambas as famílias, foi apresentado apenas o critério de seleção somente para um elemento bidimensional. Entretanto, essa metodologia é facilmente adaptada ao caso tridimensional, em que no lugar do triângulo de Pascal, utiliza-se a pirâmide de Pascal, contendo termos nas três direções (s_1, s_2, s_3) cujo vértice é unitário.

2.2 MECANISMO DE DEFORMAÇÃO

Nesta seção serão revisados alguns conceitos básicos sobre deformação de um corpo contínuo que servirão de base teórica para compreensão do funcionamento do elemento F-bar desenvolvido por (NETO et al., 1996). Para esse elemento, é interessante estudar o comportamento do gradiente de deformação F, bem como deformações isocóricas e volumétricas.

[7] (AZEVEDO, 2003)
[8] (AZEVEDO, 2003)

Deformação pode ser entendida como qualquer mudança geométrica na configuração de um corpo que produza variação em suas dimensões ou forma a partir de uma solicitação externa.

Inicia-se a descrição do mecanismo de deformação definindo um corpo β que ocupa uma região Ω. A deformação de β é definida pela função φ que mapeia cada ponto material p do corpo β em um local

$$x = \varphi(p) \tag{2.22}$$

onde ele é posicionado na configuração deformada de β. A região ocupada por esse ponto material na configuração deformada é definida como $\varphi(\Omega)$ (Fig. 9).

O vetor $u(p)$ definido como

$$u(p) = \varphi(p) - p \tag{2.23}$$

é o deslocamento de p, assim, a partir da Eq. (2.22), temos:

$$x = p + u(p) \tag{2.24}$$

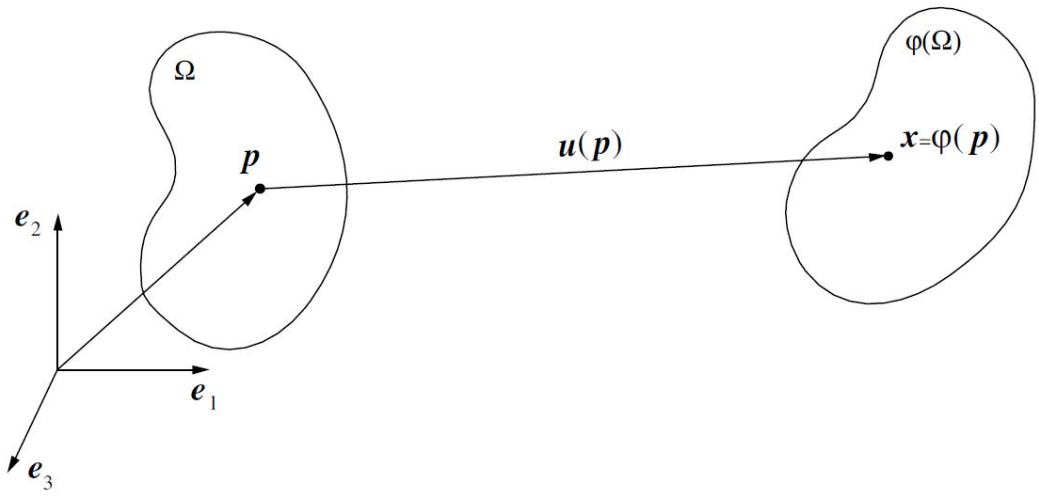


Figura 9 – Deformação do corpo β[9]

A deformação rígida de um corpo é a deformação que mantém todas as distâncias entre os pontos materiais do corpo. Ela pode ser caracterizada por uma translação, rotação ou pela combinação dessas duas últimas (Fig. 10).

[9](NETO; PERIĆ; OWEN, 2008)

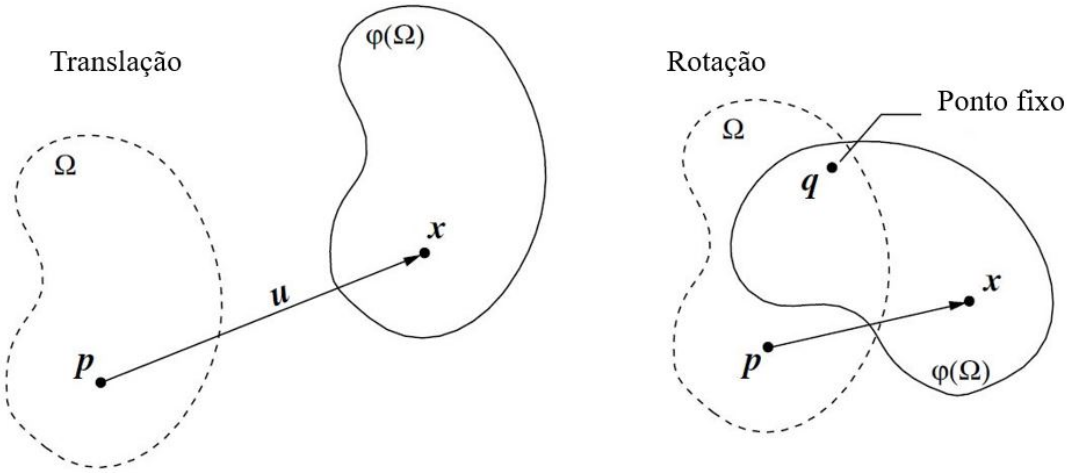


Figura 10 – Deformação rígida (Adaptado)[10]

Uma translação rígida é a deformação em que o vetor deslocamento é constante, portanto u é independente de p, sendo expressa por:

$$\varphi(p) = p + u \tag{2.25}$$

Já a rotação rígida é expressa como:

$$\varphi(p) = q + R(p - q) \tag{2.26}$$

em que R é a rotação e q o ponto de rotação. Desta forma, combinando as Eqs. (2.25) e (2.26), obtêm-se a expressão para a deformação rígida:

$$\varphi(p) = \varphi(q) + R(p - q) \tag{2.27}$$

A função temporal da deformação do corpo β é definida como **movimento** (Fig. 11) de β. Para cada instante de tempo t, $\varphi(\cdot, t)$ é a deformação dele. Durante o movimento φ, a posição x de uma partícula p no tempo t é:

$$x = \varphi(p, t) \tag{2.28}$$

[10](NETO; PERIĆ; OWEN, 2008)

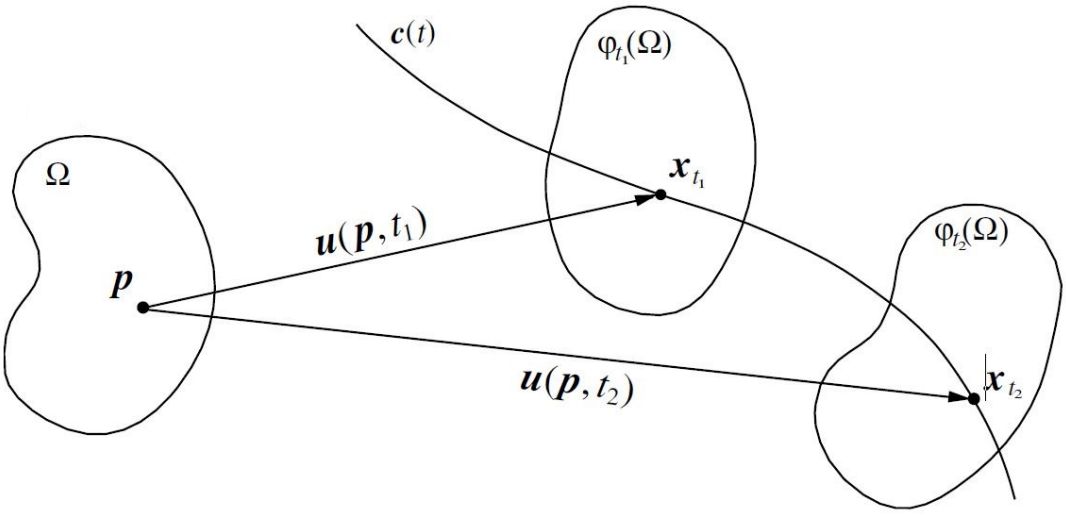


Figura 11 – Movimento de β[11]

A curva paramétrica $c(t) = \varphi(p,t)$ descreve a trajetória de p durante o movimento de β.

2.2.1 GRADIENTE DE DEFORMAÇÃO *F*

A partir do movimento φ defini-se o gradiente de deformação F como o tensor de segunda ordem dessa função (Eq. (2.29)), ilustrado pela Fig. 12.

$$F(p,t) = \nabla_p \varphi(p,t)$$

$$F = I + \nabla_p u \tag{2.29}$$

As componentes cartesianas de F são:

$$F_{ij} = \frac{\partial x_i}{\partial p_j} = \delta_{ij} + \frac{\partial u_i}{\partial p_j}$$

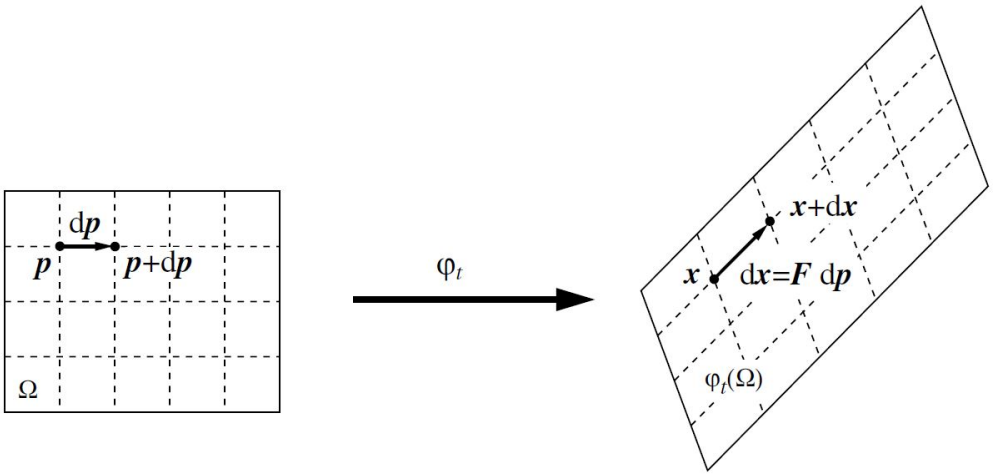


Figura 12 – Gradiente de Deformação[12]

[11] (NETO; PERIĆ; OWEN, 2008)

Tendo em vista a Fig. 12, vamos considerar um pedaço infinitesimal de material dp que interliga partículas vizinhas p e $p + \mathrm{d}p$ de um corpo em deformação. Sob a deformação φ_t, essas partículas são mapeadas em x e $x + \mathrm{d}x$. O gradiente de deformação é, portanto, o operador linear que relacionada o pedaço infinitesimal dp com sua parte deformada correspondente dx:

$$dx \;=\; F\,dp \tag{2.30}$$

Quando a deformação do corpo β produz um gradiente de deformação F uniforme (F é independente de p), tem-se uma deformação homogênea. A deformação é homogênea se e somente se for possível representá-la como

$$\varphi(p) \;=\; \varphi(q) \;+\; F(q - p)$$

para todos os pontos $p, q \in \beta$, com F positivo. Portanto, translações e rotações rígidas são deformações homogêneas.

2.2.2 MUDANÇAS NO VOLUME E O DETERMINANTE DE *F*

De maneira análoga à análise realizada na seção anterior, considera-se agora um volume infinitesimal dv_0 definido pelos vetores infinitesimais da, db e dc com origem na partícula p (Fig. 13).

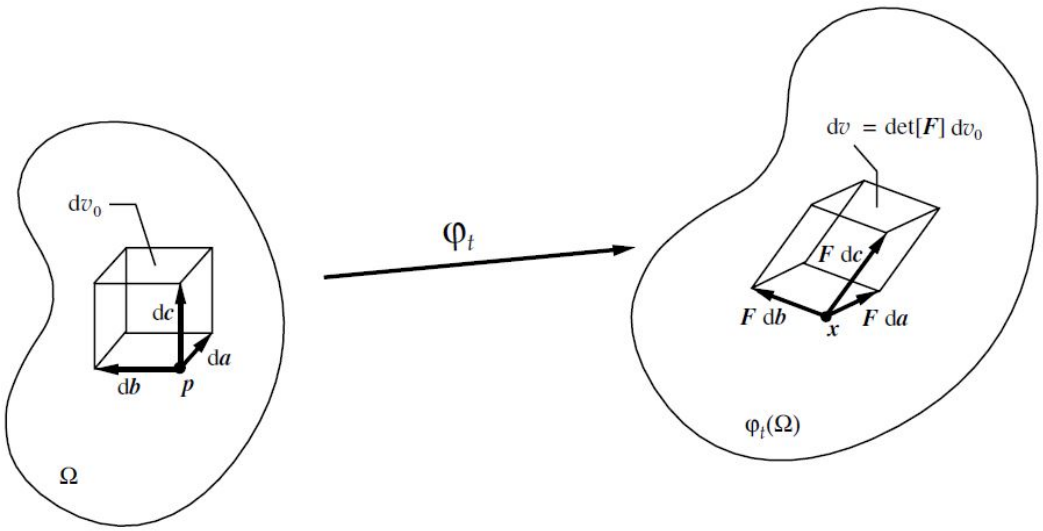


Figura 13 – Determinante do Gradiente de Deformação[13]

[12](NETO; PERIĆ; OWEN, 2008)
[13](NETO; PERIĆ; OWEN, 2008)

A deformação φ_t mapeia os vetores infinitesimais anteriores em $F\,\mathrm{d}a$, $F\,\mathrm{d}b$ e $F\,\mathrm{d}c$. Desta forma, os volumes infinitesimais na configuração inicial e deformada são:

$$\begin{aligned} dv_0 &= (da \times db) \cdot dc \\ dv &= (F\,da \times F\,db) \cdot F\,dc \end{aligned} \tag{2.31}$$

Da Eq. (2.31), utilizando a identidade matemática

$$\det T = \frac{(T\,u \times T\,v) \cdot T\,w}{(u \times v) \cdot w}$$

válida para qualquer tensor T com o conjunto $\{u, v, w\}$ de vetores linearmente independentes, temos que o $\det F$ é:

$$\det \boldsymbol{F} = \frac{dv}{dv_0} \tag{2.32}$$

O determinante do gradiente de deformação representa, localmente, o volume após a deformação por unidade de volume inicial (taxa de variação de volume específica). Da Eq. (2.32), tem-se as seguintes conclusões:

- Se $\det \boldsymbol{F} = 0$, o volume infinitesimal foi colapsado em uma partícula. Como o corpo não pode penetrar em si mesmo, essa situação representa uma condição fisicamente inadmissível;
- Na configuração inicial, $\boldsymbol{F} = \boldsymbol{I}$, então $\det \boldsymbol{F} = 1$. Portanto, uma situação em que $\det \boldsymbol{F} < 0$ só poderá existir se em algum momento o $\det \boldsymbol{F}$ for nulo. Desta forma, para qualquer deformação de um corpo, é necessário satisfazer a condição $\det \boldsymbol{F} > 0$.

2.2.3 PARTIÇÕES ISOCÓRICA E VOLUMÉTRICA DO GRADIENTE DE DEFORMAÇÃO $\boldsymbol{F}$

Qualquer deformação pode ser localmente decomposta em uma contribuição puramente volumétrica e uma puramente isocórica, conforme explorado por (SIMO; TAYLOR; PISTER, 1985), assim F pode ser reescrito como:

$$\boldsymbol{F} = \boldsymbol{F}_{iso}\,\boldsymbol{F}_V = \boldsymbol{F}_V\,\boldsymbol{F}_{iso} \tag{2.33}$$

As deformações isocóricas são deformações que não produzem alteração no volume. Localmente, elas são caracterizadas por

$$\det \boldsymbol{F} = 1$$

Já as deformações volumétricas são deformações produzidas puramente a partir de uma dilatação (ou contração) uniforme em todas as direções. O gradiente de deformação de qualquer deformação volumétrica é um tensor esférico

$$\boldsymbol{F} = \alpha\, I$$

em que α é o escalar correspondente à taxa dilatação (ou contração) específica. Na Eq. (2.33), as componentes isocóricas e volumétricas são definidas

$$\begin{cases} \boldsymbol{F}_V \equiv (\det \boldsymbol{F})^{\frac{1}{3}} I \\ \boldsymbol{F}_{iso} \equiv (\det \boldsymbol{F})^{-\frac{1}{3}} \boldsymbol{F} \end{cases} \tag{2.34}$$

É possível verificar que de fato $\det \boldsymbol{F}_V$ corresponde a uma deformação puramente volumétrica, já que

$$\det \boldsymbol{F}_V = [(\det \boldsymbol{F})^{\frac{1}{3}}]^3 \det I = \det \boldsymbol{F}$$

portanto $\boldsymbol{F}_V$ produz a mesma mudança no volume que $\boldsymbol{F}$.

De maneira análoga, verifica-se que de fato $\boldsymbol{F}_{iso}$ representa uma deformação com conservação do volume, pois

$$\det \boldsymbol{F}_{iso} = [(\det \boldsymbol{F})^{-\frac{1}{3}}]^3 \det \boldsymbol{F} = 1$$

2.3 ELEMENTO F-BAR

Com o intuito de tornar possível a resolução de problemas que envolvem grandes deformações, será utilizado o elemento finito F-bar desenvolvido por (NETO et al., 1996). Esse elemento será introduzido no *software* Abaqus através de uma rotina UEL e funcionará como um novo elemento finito do programa. O código da rotina foi baseado no trabalho de (HENANN; CHESTER; BERTOLDI, 2013) e adaptado à necessidade deste estudo. Sua implementação será melhor explicada na Seção 3.2.

O elemento F-bar é desenvolvido a partir do elemento finito padrão inserindo um gradiente de deformação modificado no cálculo do tensor de tensões. Para evitar o efeito de travamento volumétrico, esse gradiente de deformação modificado é construído de tal forma que as restrições de incompressibilidade possam ser aplicadas em um sentido médio em todo o elemento, e não pontualmente.

Segundo (NETO; PERIĆ; OWEN, 2008), diversas análises numéricas mostraram que elementos F-bar produzem boas soluções com malhas de tamanho razoável para uma grande diversidade de problemas industrialmente relevantes. Nessas aplicações, a metodologia evitou o efeito de travamento volumétrico e, além disso, mostrou-se bastante adequada na captura dos fenômenos de deformação.

Outra característica importante desse elemento, consiste na possibilidade de sua implementação em conjunto com ferramentas de malha adaptativa, devido à ausência de parâmetros de elementos internos não físicos (NETO et al., 1996). A formulação deste elemento será explicada a seguir.

Considere um quadrilátero de 4 nós ou um hexaedro de oito nós com coordenadas locais definidas como ξ (Fig. 14). A integração numérica do elemento requer o cálculo do tensor de tensões. Tome como um intervalo de tempo $[t_n, t_{n+1}]$ e seja σ_{n+1} o tensor de tensões para o ponto de integração i no instante t_{n+1}.

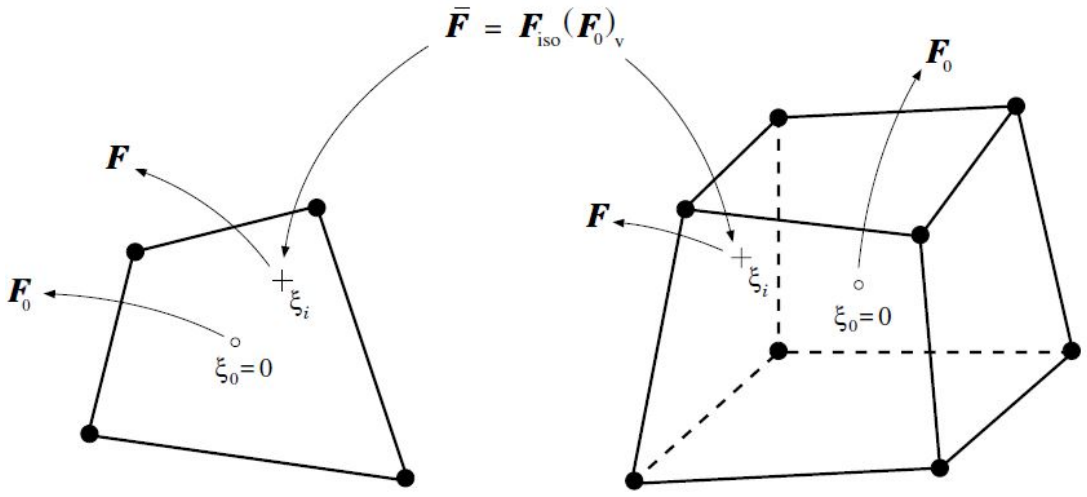


Figura 14 – Elemento F-bar quadrilateral de quatro nós e hexaédrico de oito nós[14]

Para problemas de deformações finitas, o tensor σ é o obtido a partir do gradiente de deformação ao final do intervalo de tempo através de funções constitutivas

$$\sigma_{n+1} = \hat{\sigma}(\alpha_n, F) \tag{2.35}$$

onde α_n é o o conjunto de variáveis internas do modelo no instante t_n. Para elementos finitos convencionais, o gradiente de deformação F é calculado diretamente da interpolação do campo de deslocamentos em um ponto de integração genérico i.

A ideia central por trás desta formulação se dá a partir do conceito da separação do gradiente de deformação F descrito na Seção 2.2.3. O método F-bar consiste então na substituição do gradiente convencional F pelo gradiente modificado equivalente, definido como gradiente de deformação F-bar ($\bar{F}$), com o intuito de contornar o travamento volumétrico residual produzido pelos elementos padrões de baixa ordem próximos ao limite de incompressibilidade.

Para a construção desse novo gradiente $\bar{F}$, aplica-se a ideia de separação das contribuições para o gradiente F no ponto de Gauss de interesse, bem como para F_0, que é resultado da interpolação convencional dos deslocamentos no centroide do elemento ξ_0

[14](NETO; PERIĆ; OWEN, 2008)

(Fig. 14). Logo:

$$\begin{cases} \boldsymbol{F} = \boldsymbol{F}_{iso}\, \boldsymbol{F}_V \\ \boldsymbol{F}_0 = (\boldsymbol{F}_0)_{iso}\, (\boldsymbol{F}_0)_V \end{cases} \tag{2.36}$$

O gradiente de deformação F-bar é definido como o produto entre a componente isocórica de F pela componente volumétrica de F_0, ou seja:

$$\bar{\boldsymbol{F}} = \boldsymbol{F}_{iso}\, (\boldsymbol{F}_0)_V = \left(\frac{\det \boldsymbol{F}_0}{\det \boldsymbol{F}} \right)^{\frac{1}{3}} \mathbf{F} \tag{2.37}$$

Tendo o gradiente $\bar{F}$ definido, o elemento quadrilateral de quatro nós F-bar é obtido substituindo-se F por $\bar{F}$ na Eq. (2.35). Desta forma, para este elemento, o tensor de tensões é dado por:

$$\sigma_{n+1} = \hat{\sigma}(\alpha_n, \bar{\boldsymbol{F}}) \tag{2.38}$$

Para a construção do gradiente modificado $\bar{F}$, a separação de suas contribuições é resumida em:

$$\begin{cases} \bar{\boldsymbol{F}}_{iso} = (\det \boldsymbol{F})^{-\frac{1}{3}}\, \boldsymbol{F} = \boldsymbol{F}_{iso} \\ \bar{\boldsymbol{F}}_V = (\det \boldsymbol{F}_0)^{\frac{1}{3}}\, I = (\boldsymbol{F}_0)_V \end{cases} \tag{2.39}$$

Assim, a partir do conjunto de relações da Eq. (2.39), observa-se que a componente isocórica de $\bar{F}$ coincide com a componente semelhante do gradiente F (obtido pelas funções de interpolação convencionais), enquanto que a parte volumétrica corresponde a dilatação no centroide do elemento. Essa formulação implica, para materiais em que as contribuições isocórica e volumétrica são desacopladas, uma pressão constante ao longo de todo o elemento.

2.3.1 O VETOR DE FORÇA INTERNA

Vamos recorrer novamente ao Princípio dos Trabalhos Virtuais para um elemento finito em equilíbrio:

$$\mathtt{F}^{INT} - \mathtt{F}^{EXT} = 0 \tag{2.40}$$

em que $\mathtt{F}^{EXT}$ é o vetor de força externa, e $\mathtt{F}^{INT}$ é o obtido através da assemblagem dos vetores de força interna do elemento expressos, na configuração espacial, por:

$$\mathtt{F}_e^{INT} = \int_{\varphi(\Omega^e)} B^T \sigma(\bar{\boldsymbol{F}}) dv \tag{2.41}$$

Nota-se da Eq. (2.41) que a única diferença no cálculo do vetor de força interna do elemento F-bar e do elemento convencional é a avaliação do tensor de Cauchy σ em

$\bar{F}$. Desta forma, além da avaliação de F e σ em cada ponto de Gauss, procedimento que é realizado pelo elemento padrão, o elemento F-bar proposto requer apenas o cálculo do $\det F_0$ (no centróide) e de $\bar{F}$, que pela definição em Eq. (2.37), este último é obtido simplesmente pela multiplicação do gradiente F por um escalar.

Além disso, diferentemente de outras metologias propostas para análise de elementos finitos que envolvem grandes deformações, como os procedimentos conhecidos como métodos B-bar, desenvolvidos por (SIMO; TAYLOR; PISTER, 1985) e (MORAN; ORTIZ; SHIH, 1990), o elemento F-bar não altera a matriz de deformação B. Isso deve-se ao fato de que o gradiente de deformação utilizado foi introduzido na equação constitutiva da tensão e não na de energia de deformação (NETO et al., 1996).

2.3.2 LINEARIZAÇÃO CONSISTENTE E A MATRIZ DE RIGIDEZ TANGENTE $\mathtt{K}$

Para a computação de elementos finitos de forma implícita, a linearização da Eq. (2.40) é base do método de Newton-Rapshon para a obtenção da solução do problema de valor de contorno discretizado.

Define-se como R uma força fora de equilíbrio (ou resíduo) e U como vetor de deslocamento global do elemento finito, tal que $R \to R(U)$, é possível reescrever a equação de equilíbrio Eq. (2.40) da seguinte forma:

$$R(U) := \mathtt{F}^{INT}(U) - \mathtt{F}^{EXT} = 0 \tag{2.42}$$

O método de Newton-Raphson permite, portanto, a resolução de um problema de convergência (ou, no caso, minimização de resíduos), de tal forma que a cada iteração k e para cada incremento $\Delta U^{(k)}$, o sistema abaixo seja resolvido:

$$\mathtt{K}(U^{(k)})[\Delta U^{(k)}] = -R(U) \tag{2.43}$$

assim, ao final do *loop* de iterações busca-se um valor próximo a zero para o resíduo $R(U)$.

Na Eq. (2.43), $\mathtt{K}$ é a matriz de rigidez tangente. Ela é obtida de forma análoga ao vetor de força interna $\mathtt{F}^{INT}$, através da assemblagem das matrizes de rigidez do elemento:

$$\mathtt{K}_e = \underbrace{\int_{\varphi(\Omega^e)} G^T [a]_{\boldsymbol{F}=\bar{\boldsymbol{F}}} G dv}_{\text{rigidez do elemento convencional para } F=\bar{F}} + \underbrace{\int_{\varphi(\Omega^e)} G^T [q](G_0 - G) dv}_{\text{rigidez adicional}} \tag{2.44}$$

em que G é o operador gradiente espacial e G_0 o operador gradiente espacial computado

no centróide do elemento. A matriz G para um elemento bidimensional de quatro nós é:

$$G = \begin{bmatrix} \frac{\partial N_1}{\partial x_1} & 0 & \frac{\partial N_2}{\partial x_1} & 0 & \frac{\partial N_3}{\partial x_1} & 0 & \frac{\partial N_4}{\partial x_1} & 0 \\ 0 & \frac{\partial N_1}{\partial x_1} & 0 & \frac{\partial N_2}{\partial x_1} & 0 & \frac{\partial N_3}{\partial x_1} & 0 & \frac{\partial N_4}{\partial x_1} \\ \frac{\partial N_1}{\partial x_2} & 0 & \frac{\partial N_2}{\partial x_2} & 0 & \frac{\partial N_3}{\partial x_2} & 0 & \frac{\partial N_4}{\partial x_2} & 0 \\ 0 & \frac{\partial N_1}{\partial x_2} & 0 & \frac{\partial N_2}{\partial x_2} & 0 & \frac{\partial N_3}{\partial x_2} & 0 & \frac{\partial N_4}{\partial x_2} \end{bmatrix} \tag{2.45}$$

$[a]_{F=\bar{F}}$ é o módulo tangente espacial, mais detalhado no Apêndice A, expresso por:

$$a_{ijkl} = \frac{1}{\det \boldsymbol{F}} \boldsymbol{F}_{jp} \boldsymbol{F}_{tq} A_{ipkq} \tag{2.46}$$

e $[q]$ é a matriz de forma do tensor de quarta ordem expressa por:

$$q := \frac{1}{3} a : (I \otimes I) - \frac{2}{3} (\sigma \otimes I) \tag{2.47}$$

Na Eq. (2.46), A_{ipkq} denota o módulo tangente material (ver Apêndice A), definido como:

$$A_{ipkq} = \frac{\partial P_{ip}}{\partial \boldsymbol{F}_{kq}} \tag{2.48}$$

onde $P \to P(F)$ é o primeiro tensor de tensão de Piola-Kirchhoff, que é função do gradiente de deformação F.

Na expressão Eq. (2.44), verifica-se que o primeiro termo é idêntico à rigidez tangente de um elemento convencional calculados em $F = \bar{F}$. A adição do segundo termo da equação não é tão complicada, tendo em vista ser necessário computar-se somente G no centróide, G_0, e a matriz $[q]$. O custo computacional requerido para o cálculo da rigidez tangente proposta é, portanto, ligeiramente maior em relação ao cálculo da rigidez tangente convencional (NETO et al., 1996).

2.3.3 ADEQUAÇÃO DO ELEMENTO PARA O ESTADO PLANO DE DEFORMAÇÃO

Tendo em vista que o elemento utilizado será bidimensional, é interessante ressaltar as alterações do elemento F-bar para um estado de plano de deformação. Para isso, define-se $\bar{F}_p$ como o gradiente modificado da componente plana F_p do gradiente de deformação F, definido como:

$$\bar{\boldsymbol{F}}_p = \left(\frac{\det \boldsymbol{F}_{p0}}{\det \boldsymbol{F}_p} \right)^{\frac{1}{2}} \boldsymbol{F}_p \tag{2.49}$$

Nota-se que para esta definição de plano de deformação, temos que a segunda expressão da Eq. (2.39) para a componente volumétrica se mantém válida, entretanto na

primeira expressão, a componente isocórica modificada não coincide com a componente isocórica convencional.

Além disso, a expressão da matriz $[q]$ (Eq. (2.47)) também deve ser reescrita:

$$q := \frac{1}{2}a : (I \otimes I) - \frac{1}{2}(\sigma \otimes I) \tag{2.50}$$

2.4 MODELOS DE HIPERELASTICIDADE

No código da UEL, além da subrotina de cálculo do novo gradiente de deformação $\bar{F}$ está também acoplado ao código o modelo constitutivo material, isto está melhor discutido na Seção 3.2. Em três das quatro análises realizadas, o modelo material utilizado no código é o modelo neo-Hookean para materiais hiperelásticos. Desta forma, faz-se uma breve revisão teórica sobre esse modelo constitutivo.

2.4.1 CONCEITOS BÁSICOS DE HIPERELASTICIDADE

Modelos constitutivos hiperelásticos são adequados para descrever o comportamento de diversos materiais de engenharia. No contexto de deformações finitas, essas teorias são particularmente adequadas para análises de materiais elastoméricos e em muitos casos mostram excelente concordância com experimentos que envolvem deformações acima de 700% (OGDEN, 1986).

Os modelos hiperelásticos são não-dissipativos e por essa razão não levam em consideração variáveis internas. Um modelo hiperelátisco finito genérico é caracterizado pela existência de uma função de energia livre específica (*free-energy function*) na forma:

$$\psi = \psi(\boldsymbol{F}) \tag{2.51}$$

Funções desse tipo - que não dependem de variáveis internas - são casos particulares do potencial geral $\psi = \psi(\boldsymbol{F}, \theta, \alpha)$. A partir da Eq. (2.51), define-se o primeiro tensor de tensão de Piola-Kirchhoff:

$$P = \bar{\rho}\,\frac{\partial \psi(\boldsymbol{F})}{\partial \boldsymbol{F}} \tag{2.52}$$

onde $\bar{\rho}$ é a massa específica na configuração de referência. Neste caso, a tensão só depende da configuração nova/imediata - ao final do incremento - e não é afetada pelo histórico de deformação (NETO; PERIĆ; OWEN, 2008). A obtenção do tensor P para o modelo de neo-Hookean está melhor descrita no Apêndice B.

A equação constitutiva para o tensor de tensão de Kirchhoff τ_{kh} é:

$$\tau_{kh} = P\ \boldsymbol{F}^T \tag{2.53}$$

para um material hiperelástico esse tensor é dado por:

$$\tau_{kh}(\boldsymbol{F}) = \bar{\rho}\,\frac{\partial\psi(\boldsymbol{F})}{\partial\boldsymbol{F}}\,\boldsymbol{F}^T \tag{2.54}$$

e o tensor de tensões $\sigma = \tau_{kh}/\det\boldsymbol{F}$:

$$\sigma(\boldsymbol{F}) = \frac{\bar{\rho}}{\det\boldsymbol{F}}\,\frac{\partial\psi(\boldsymbol{F})}{\partial\boldsymbol{F}}\,\boldsymbol{F}^T \tag{2.55}$$

2.4.2 MODELOS DE MOONEY-RIVLIN E NEO-HOOKEAN

Dois modelos constitutivos importantes para materiais elastoméricos serão descritos nesta seção: os modelos de Mooney-Rivlin e neo-Hookean. Devido à sua simplicidade matemática e boa precisão quando grandes deformações são envolvidas, esses modelos são bastante utilizados na descrição do comportamento de materiais semelhantes à borracha (NETO; PERIĆ; OWEN, 2008).

Na versão original desses modelos, a configuração de incompressibilidade perfeita é assumida. Para o modelo Mooney-Rivlin, a energia de deformação é expressa por:

$$\bar{\rho}\,\bar{\psi}(I_1, I_2) = C_1(I_1 - 3) + C_2(I_2 - 3) \tag{2.56}$$

em que C_1 e C_2 são constantes do material.

O modelo de neo-Hookean é o modelo mais simplificado para materiais semelhantes à borracha. Trata-se de um caso particular do modelo de Mooney-Rivlin quando $C_2 = 0$, assim:

$$\bar{\rho}\,\bar{\psi}(I_1, I_2) = C_1(I_1 - 3) \tag{2.57}$$

Para obter-se a versão compressível desses dois modelos, é necessário recorrer às equações (2.33) e (2.34) referentes à partição do gradiente de deformação F. A partir dessas equações define-se o tensor de deformação esquerdo isocórico de Cauchy-Green:

$$\mathtt{B}_{iso} = \boldsymbol{F}_{iso}\,\boldsymbol{F}_{iso}{}^T = (\det\boldsymbol{F})^{-\frac{2}{3}}\,\boldsymbol{F}\,\boldsymbol{F}^T \tag{2.58}$$

e as invariantes principais:

$$I_1^* = \mathrm{tr}\mathtt{B}_{iso} \qquad\qquad I_2^* = \frac{1}{2}\{(I_1^*)^2 \;-\; \mathrm{tr}[\mathtt{B}_{iso}^2]\} \tag{2.59}$$

Desta forma, a versão compressível do modelo Mooney-Rivlin é:

$$\bar{\rho}\,\bar{\psi}^*(I_1^*, I_2^*, \det\boldsymbol{F}) = C_1(I_1^* - 3) + C_2(I_2^* - 3) + \frac{1}{2}K(\ln\det\boldsymbol{F})^2 \tag{2.60}$$

já a versão compressível correspondente ao modelo de neo-Hookean:

$$\bar{\rho}\,\bar{\psi}^*(I_1^*, \det \boldsymbol{F}) = C_1(I_1^* - 3) + \frac{1}{2}K(\ln \det \boldsymbol{F})^2 \tag{2.61}$$

em que K é o módulo de incompressibilidade e $\mu = 2C_1$, onde μ é o módulo de cisalhamento.

2.5 MODELO ELASTOPLÁSTICO

Além do modelo hiperlástico Neo-Hookean, também será utilizado um modelo elastoplástico na análise de um problema de contato (Seção 4.4). Nessa análise a tensão de contato é dada por um modelo de endurecimento isotrópico linear. Portanto, faz-se a seguir uma breve descrição desse modelo utilizado.

Um modelo elastoplástico genérico deve possuir os seguintes componentes:

- Decomposição da deformação elastoplástica;
- Equação de hiperelasticidade;
- Critério de escoamento;
- Fluxo de plasticidade;
- Equação de encruamento.

Uma hipótese da teoria de plasticidade no regime de grandes deformações consiste na decomposição do gradiente de deformações F em um produto entre uma componente plástica F^p e elástica F^e.

$$\boldsymbol{F} = \boldsymbol{F}^e\,\boldsymbol{F}^p \tag{2.62}$$

A relação referente ao regime de hiperelasticidade é derivada da equação de energia ψ expressa como uma função do logaritmo da deformação elástica ε^e, $\psi = \psi(\varepsilon^e, \boldsymbol{\alpha})$

$$\boldsymbol{\tau} = \bar{\rho}\,\frac{\partial \psi}{\partial \varepsilon^e} \tag{2.63}$$

em que A é definido como

$$\boldsymbol{A} = \bar{\rho}\,\frac{\partial \psi}{\partial \boldsymbol{\alpha}} \tag{2.64}$$

O critério de escoamento $\Phi(\boldsymbol{\tau}, \boldsymbol{A})$ é utilizado para definir o início do regime plástico. Desta forma, o regime elástico (reversível) é estabelecido como

$$\mathtt{E} = \{\boldsymbol{\tau} | \Phi(\boldsymbol{\tau}, \boldsymbol{A}) < 0\} \tag{2.65}$$

O regime plástico se dá, portanto, quando $\boldsymbol{\tau}$ se encontra no limite do regime elástico, ou seja, na superfície de escoamento

$$\mathtt{P} = \{\boldsymbol{\tau} | \Phi(\boldsymbol{\tau}, \boldsymbol{A}) = 0\} \tag{2.66}$$

Já o fluxo de plasticidade $\Psi(\boldsymbol{\tau}, \boldsymbol{A})$ é determinado a partir da equação constitutiva do elongamento plástico

$$\mathtt{D}^p = \dot{\gamma}\,\frac{\partial \Psi}{\partial \boldsymbol{\tau}} \tag{2.67}$$

A variação no conjunto de variáveis internas $\boldsymbol{\alpha}$ é

$$\dot{\boldsymbol{\alpha}} = \dot{\gamma}\,\boldsymbol{H}(\boldsymbol{\tau}, \boldsymbol{A}) \tag{2.68}$$

em que $\boldsymbol{H}$ é o módulo de encruamento dado por

$$\boldsymbol{H}(\boldsymbol{\tau}, \boldsymbol{A}) = -\frac{\partial \Psi}{\partial \boldsymbol{A}} \tag{2.69}$$

e $\dot{\gamma}$ é o multiplicador plástico, um escalar não negativo, $\dot{\gamma} \geq 0$, que satisfaz

$$\Phi\dot{\gamma} = 0 \Rightarrow \Phi \leq 0$$

2.5.1 ENCRUAMENTO ISOTRÓPICO LINEAR

O encruamento é dito isotrópico se o crescimento da superfície de escoamento, para qualquer nível de encruamento, corresponde a uma expansão uniforme da superfície inicial, ou seja, mantém-se a forma inicial sem sofrer translação (Fig. 15).

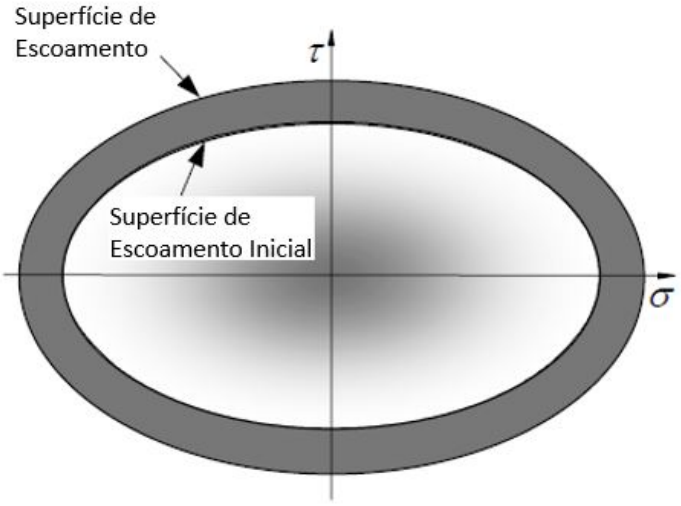


Figura 15 – Encruamento isotrópico[15]

Para um modelo isotrópico linear, a função de encruamento é linear e expressa por

$$\boldsymbol{\tau}_y(\bar{\varepsilon^p}) = \boldsymbol{\tau}_{y0} + \boldsymbol{H}\,\bar{\varepsilon^p} \tag{2.70}$$

em que $\boldsymbol{\tau}_{y0}$ é o escoamento inicial, $\boldsymbol{H}$ é o módulo de encruamento isotrópico linear e ε^p é o logaritmo da deformação plástica acumulada.

[15](JORGE; DINIS, 2004)

3 MODELAGEM NUMÉRICA

Nesta seção será realizada uma breve introdução acerca do *software* Abaqus que será utilizado para as simulações numéricas, detalhando alguns de seus elementos disponíveis, bem como os elementos finitos que serão utilizados nas simulações. Além disso, será explicado também a lógica de implementação e funcionamento da rotina UEL.

3.1 TECNOLOGIA DE ELEMENTOS DO ABAQUS

O pacote comercial de simulação Abaqus fornece uma ampla variedade de tipos de elementos de discretização para diversas geometrias e tipos de análises (Fig. 16). Um elemento pode ser caracterizado da seguinte maneira:

- Família: contínuo, casca, membrana, rígido, viga, treliça, etc;
- Número de nós: dependendo da forma do elemento e ordem de interpolação;
- Graus de liberdade por nó: conforme o método de solução (deslocamento, rotação, temperatura, etc) e a dimensão do modelo (1D, 2D ou 3D);
- Formulação (pequenas ou grandes deformações);
- Tipo de integração: reduzida ou plena.

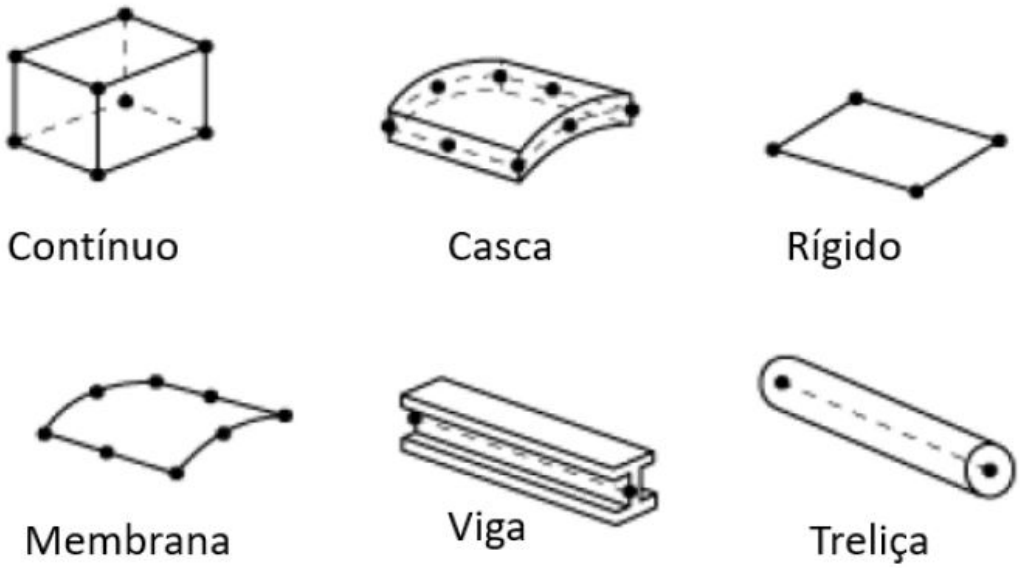


Figura 16 – Famílias de elementos comumente utilizadas (Adaptado)[1]

A escolha do elemento a ser utilizado requer conhecimento da teoria e aplicação da análise de elementos finitos. Neste trabalho será utilizado o elemento sólido (ou contínuo), esse elemento é usado em análises lineares e em análises não lineares complexas que envolvem contato, plasticidade e grandes deformações. Ele permite interpolações de primeira ordem (linear) e segunda ordem (quadráticas), entretanto para o presente estudo foca-se no elemento de primeira ordem, tendo em vista que o elemento F-bar implementado já possibilita a resolução de problemas que envolvem grandes deformações, deixando de lado, portanto, a necessidade de interpolações de ordem superior para esse tipo de problema.

Há também opções para a forma do elemento, para os bidimensionais temos os quadrilaterais e triangulares, já para o caso tridimensional, elementos hexaédricos e tetraédricos. A utilização de elementos quadrilaterais e hexaédricos é mais frequente em relação aos outros elementos, entretanto, eles são mais difíceis de utilizar em geometrias complexas (DOCUMENTATION, 2014).

Na análise de elementos finitos a rigidez e a massa dos elementos são calculadas através de integrações numéricas; o Abaqus utiliza o método da quadratura Gaussiana para a maior parte dos elementos. A integração reduzida das matrizes dos elementos faz uso de integrações de baixa ordem, em, geralmente, só um ponto de Gauss, isso permite cálculos mais rápidos e menos custosos computacionalmente, mas pode trazer consequências significativas na precisão da solução para um determinado problema, além de sofrer com problemas de flexibilidade excessiva, conhecido como *Hourglassing*.

O efeito de Hourglassing (Fig. 17) afeta elementos de primeira ordem com integração reduzida em análises de tensões/deformações. Como há somente um ponto de

[1] (DOCUMENTATION, 2014)

integração, é possível que o elemento distorça de maneira que as deformações calculadas nesse ponto de integração sejam nulas, gerando distorções não controladas na malha. Uma maneira de controlar esse efeito é adicionando um controle de *Hourglass*, isso significa acrescentar uma rigidez virtual ao elemento. Outra forma de minimizar esse problema é distribuindo cargas pontuais e condições de contorno em diversos nós adjacentes (DOCUMENTATION, 2014).

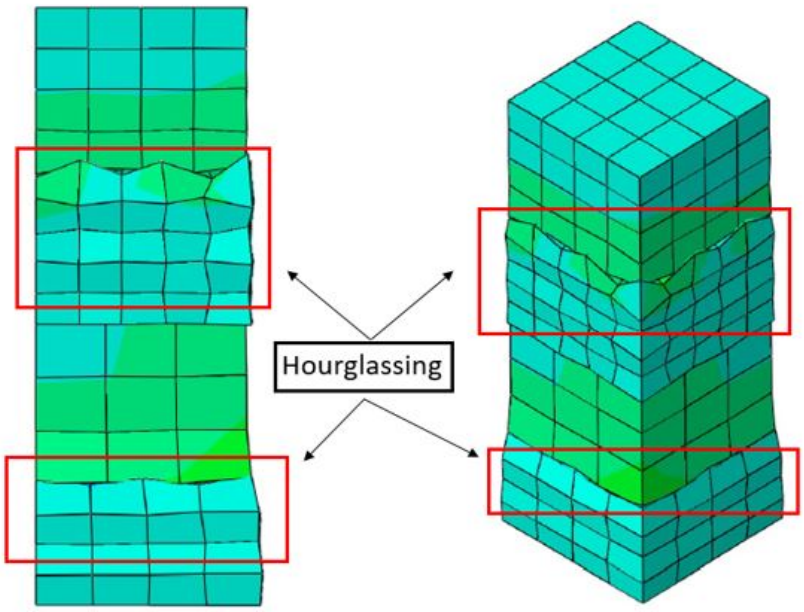


Figura 17 – Efeito de *Hourglassing* em um elemento de integração reduzida

Em contrapartida, integrações plenas sofrem com efeitos de travamento volumétrico e cisalhante (*Volumetric and Shear locking*). O travamento cisalhante ocorre em elementos de primeira ordem de integração plena que estão sujeitos à flexão, nesse caso a formulação numérica dá origem a deslocamentos errados, tensões falsas e cisalhamentos que não existem, conhecidos como modos espúrios. Em uma situação ideal, um corpo sob flexão pura deveria se comportar conforme a Fig. 18.

Um modelo ideal deveria ter a capacidade de assumir a curvatura, porém os vértices de um elemento de primeira ordem de integração pura não são capazes de se curvarem. Em vez disso, eles apresentam o comportamento da Fig. 19, as linhas tracejadas permanecem retas, porém ângulo entre elas não continua sendo 90°. Assim, a superfície superior experimenta uma tensão de tração, enquanto a inferior tensão de compressão.

Para causar a mudança no ângulo, uma tensão cisalhante "artificial" teve que ser introduzida. Isso significa também que a energia de deformação do elemento está gerando deformação cisalhante e não flexão (*bending deformation*).

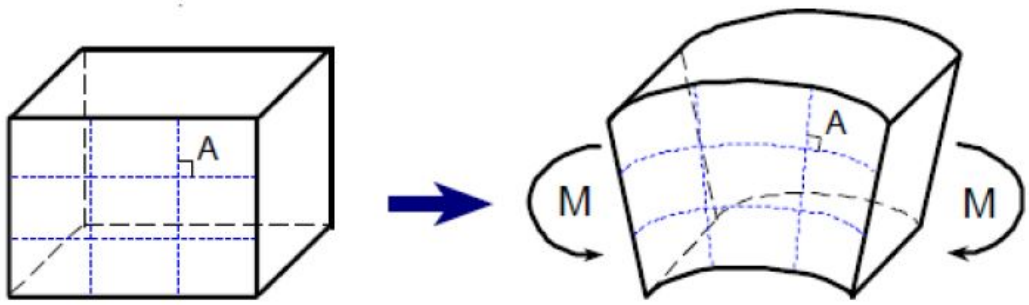


Figura 18 – Mudança de forma de um corpo sob flexão pura em uma situação ideal

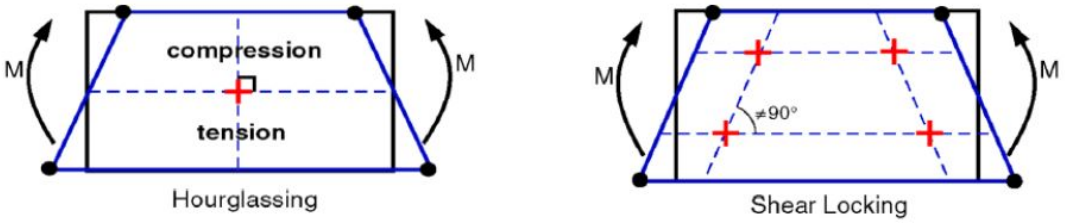


Figura 19 – Efeitos de *Hourglassing* e *Shear Locking* em elementos quadrilaterais[2]

Já o problema de travamento volumétrico (*Volume locking*) ocorre quando o comportamento do material é quase incompressível. Tensões espúrias são geradas nos pontos de integração, causando um comportamento extremamente rígido do elemento para deformações que não devem provocar mudanças no volume.

Além dos elementos de integração plena e reduzida, o Abaqus ainda possui elementos Híbridos. Esses elementos são destinados para materiais que possuem comportamento incompressível ou quase incompressível. Uma situação quase incompressível se dá quando o módulo de incompressibilidade é muito maior que o módulo de cisalhamento e tem-se o comportamento que se aproxima do limite de incompressibilidade: um pequeno deslocamento produz grandes mudanças na pressão. Esse comportamento é removido tratando a tensão devido à pressão como uma variável de solução básica interpolada independentemente, acoplada à solução de deslocamento através da teoria constitutiva e da condição de compatibilidade. Essa interpolação não-dependente da tensão devido à pressão é a base dos elementos híbridos (DOCUMENTATION, 2014).

Tendo em vista que o elemento F-bar implementado é bidimensional, os elementos do Abaqus que serão utilizados também serão bidimensionais, todos eles são contínuos e de interpolação de primeira ordem, além disso a suposição do estado plano de deformação foi assumida para todos os elementos. Assim, sumariza-se os elementos finitos usados:

- CP4E: elemento quadrilateral contínuo de 4 nós de primeira ordem e integração plena;

[2] (DOCUMENTATION, 2014)

- CPE4H: elemento quadrilateral contínuo de 4 nós híbrido;
- F-bar: elemento quadrilateral contínuo de 4 nós com gradiente de deformação F modificado ($\bar{F}$).

3.2 ROTINA UEL

O programa Abaqus permite que o usuário crie programações externas ao ambiente do programa, que são implementadas através de subrotinas. Essa programação se dá por meio de códigos em Fortran que podem ser aplicados para definir, por exemplo, um comportamento de um material, um novo elemento finito, entre outras aplicações. Neste trabalho utiliza-se a rotina UEL, que tem como finalidade introduzir um novo elemento no Abaqus.

Uma rotina UEL consiste em um código que contém a metodologia de cálculo para um determinado elemento não disponível no Abaqus, neste caso o elemento F-bar. Esse código funciona, então, como um novo elemento que substitui o utilizado pelo programa. De maneira sucinta, a diferença entre o elemento finito disponibilizado pelo Abaqus e o elemento F-bar está na computação do gradiente de deformações F, conforme explicado na Seção 2.3. Assim, esse código contém os diversos processos de cálculo necessários e específicos desse novo elemento para a obtenção desse novo gradiente modificado.

Em geral, somente as subrotinas de cálculo do novo gradiente $\bar{F}$ deveriam ser suficientes para a implementação deste elemento. Entretanto, na resolução do problema de minimização de resíduos descrito pela Eq. (2.43), a assemblagem da matriz de rigidez tangente `K` requer também a computação de dois parâmetros que são funções do modelo constitutivo material, são eles o módulo tangente espacial $[a]$ e a matriz de forma do tensor de quarta ordem $[q]$ (ver Seção 2.3.2).

Por esse motivo, acoplado à UEL, também deve estar definido o modelo constitutivo material. A UEL aqui utilizada foi desenvolvida e adaptada a partir do trabalho de (HENANN; CHESTER; BERTOLDI, 2013), ela está disponível no Apêndice C. O código utilizado por (HENANN; CHESTER; BERTOLDI, 2013) envolvia o comportamento de materiais elastoméricos sujeitos a um campo elétrico. Assim, para adaptar a UEL à necessidade do trabalho, todo o efeito elétrico foi removido e o modelo constitutivo que anteriormente era o modelo de Gent (GENT, 1996; HORGAN, 2014) foi substituído pelo modelo neo-Hookean, já que esse último é disponibilizado pelo Abaqus, o que permite a comparação entre o elemento F-bar e os elementos do *software*.

3.2.1 FUNCIONAMENTO DA UEL

O funcionamento da UEL se dá a partir de subrotinas de cálculo, elas podem ser separadas em quatro grupos:

- Subrotina de cálculo UEL;
- Subrotina de cálculo do modelo constitutivo material;
- Subotinas do elemento;
- Subrotinas de serviço.

As duas primeiras subrotinas consistem na parte principal do código e por isso serão melhor detalhadas na Seção 3.2.1.1. Já as duas últimas consistem, essencialmente, em funções auxiliares para obtenção de pontos e pesos de Gauss, mapeamento e derivação das funções de forma e funções que realizam operações matemáticas como cálculo de matriz inversa e determinante de uma matriz.

O esquema de funcionamento da UEL pode ser resumido em: obtenção dos deslocamentos nodais, modificação do gradiente de deformação F, assemblagem das matrizes rigidez tangente `K` e residual R - levando-se em conta o modelo constitutivo implementado - e construção do problema de minimização de resíduos, através da alocação dos elementos de `K` e R em A_{matrx} (vetor resíduo) e RHS (matriz rigidez do problema). A partir desse ponto, o *solver* do próprio programa inicia o processo de iteração até que o resíduo seja minimizado. A Fig. 20 representa esse esquema de funcionamento.

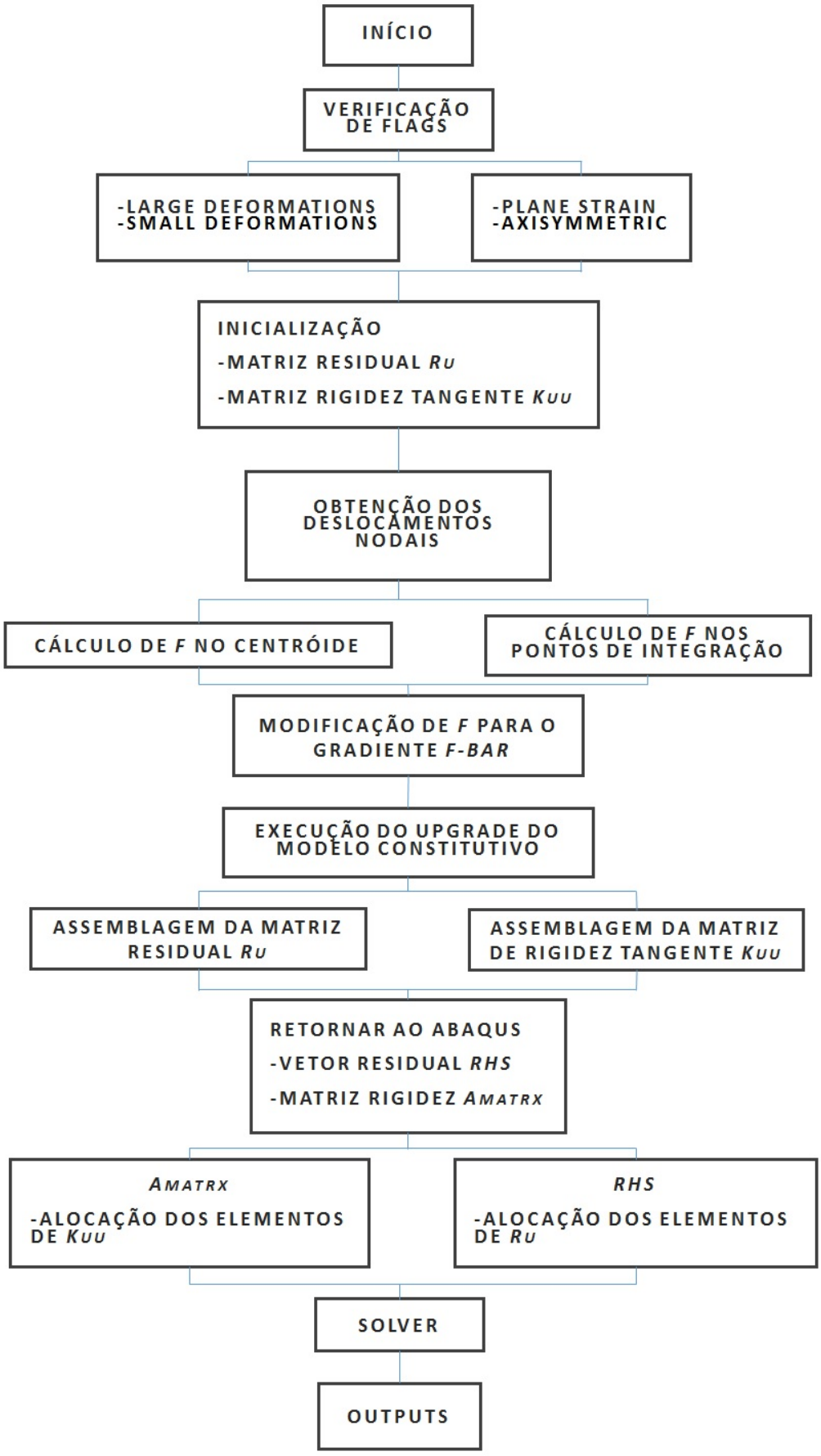


Figura 20 – Esquema de funcionamento da UEL

Na Fig. 20 estão descritos também alguns passos intermediários. No início do

código é "sinalizado" ao Abaqus através de *flags* o tipo de simulação que será realizada. É preciso garantir que o problema envolve grandes deformações e verificar se está definido no estado plano de deformação ou axissimétrico. Em seguida, as matrizes residual e de rigidez tangente, identificadas no código como R_u e K_{uu}, são inicializadas com valores nulos. Logo depois, inicia-se o processo mencionado no parágrafo anterior até que se atinja o valor mínimo residual e, por fim, obtêm-se os parâmetros de saída (deslocamentos, tensões, etc).

3.2.1.1 SUBROTINAS UEL E MODELO MATERIAL CONSTITUTIVO

Após a inicialização das matrizes rigidez tangente K_{uu} e residual R_u e a obtenção dos deslocamentos nodais, o primeiro passo para a modificação do gradiente F é iniciado com o cálculo do gradiente de deformação no centróide do elemento, conforme o algoritmo:

- Cálculos no centróide do elemento;
  - Obtenha as funções de forma e suas derivadas no centróide;
    - As funções de forma e suas derivadas são definidas pelas Eqs. (2.6) e (2.16) para as coordenadas $s_1 = s_2 = 0$.
  - Mapeie as funções de forma de coordenadas locais para globais;
    - Calcule a matriz jacobiana de mapeamento J (Eq. (2.8)), sua inversa e seu determinante;
    - Através da relação $\boxed{\frac{\partial N}{\partial \underline{x}} = \frac{\partial N}{\partial \underline{s}}\,\underline{J}^{-1}}$ (Eq. (2.15)), têm-se as derivadas das funções de forma de s_1 e s_2 (coordenadas locais) para x_1 e x_2 (coordenadas globais).
  - Mapeie as funções de forma de coordenadas locais para as coordenadas dos pontos na configuração de referência (ao final do incremento).
    - Processo análogo ao mapeamento anterior, porém as coordenadas dos pontos correspondem à posição ao final do incremento.
- Calcule o gradiente de deformação no centróide F_0.
  - Com a Eq. (2.11), $\boxed{F = \frac{\partial N}{\partial x}\,a}$, determina-se F para o ponto de coordenada no centróide do elemento.

Posteriormente, de forma semelhante à computação de F no centroide, determina-se F nos quatro pontos de integração e modifica-se o gradiente para o $\bar{F}$:

- Cálculos nos pontos de integração do elemento;
  - Obtenha as coordenadas dos pontos de integração e os pesos de Gauss;

  - Obtenha as funções de forma e suas derivadas nesses pontos de integração;
  - Mapeie as funções de forma de coordenadas locais para globais;
  - Mapeia as funções de forma de coordenadas locais para as coordenadas dos pontos na configuração de referência (ao final do incremento).
- Calcule o gradiente de deformação F para esses pontos de integração;
- Modifique o gradiente de deformação F para o gradiente F-bar $\bar{F}$.
  - Com a Eq. (2.37), $\boxed{\bar{\boldsymbol{F}} = \left(\frac{\det \boldsymbol{F}_0}{\det \boldsymbol{F}}\right)^{\frac{1}{3}} \mathbf{F}}$, obtém-se $\bar{F}$.

Finalizado a modificação de F, o processo de assemblagem K_{uu} e R_u pode ser iniciado. Entretanto, para a computação dessas matrizes é necessário realizar um passo intermediário e executar a subrotina do modelo material constitutivo.

Isso deve-se ao fato de que a matriz rigidez tangente (Eq. (2.44)) é função de dois parâmetros que estão vinculados ao modelo material, são eles: o módulo tangente material $A_{ipkq} = \frac{\partial P_{ip}}{\partial \boldsymbol{F}_{kq}}$, sendo $P = \bar{\rho}\, \frac{\partial \psi(\boldsymbol{F})}{\partial \boldsymbol{F}}$, onde ψ tem relação direta com o modelo constitutivo, e o módulo tangente espacial $[a]$, definido pela Eq. (2.46), que leva em conta a avaliação de A.

Já a matriz residual (Eq. (2.41)) requer o processamento do tensor de Cauchy $\sigma(\boldsymbol{F}) = \frac{P}{\det \boldsymbol{F}}\, \boldsymbol{F}^T$, sendo P calculado nesta etapa intermediária.

Com a atualização promovida pelo modelo constitutivo, monta-se a matriz R_u. Tendo em vista que ela é definida a partir de uma integral (Eq. (2.41)), é necessário discretizá-la e calcula-la através do método da quadratura de Gauss. O algoritmo para esse passo é:

- Compute a matriz residual;
  - Aloque as componentes do tensor de Cauchy σ em um vetor S;
  - Monte a matriz de deformação B a partir das derivadas das funções de forma;
  - Retorne a matriz R_u através de $\boxed{R_u = -B^T\, S\, (\det J) w h}$.

Já o algoritmo que traduz o processamento da matriz rigidez tangente K_{uu} é:

- Compute a matriz de rigidez tangente;
  - Monte a matriz G com as derivadas das funções de forma nos pontos de integração;
  - Monte a matriz G_0 com as derivadas das funções de forma calculadas no

centróide do elemento;

- Assemble o módulo tangente espacial $\boxed{a_{ijkl} = \frac{1}{\det \boldsymbol{F}} \boldsymbol{F}_{jp} \boldsymbol{F}_{tq} A_{ipkq}}$;
  - Obtenha o tensor P;
  - Obtenha o módulo tangente material $\boxed{A = \frac{\partial P}{\partial \boldsymbol{F}}}$;
  - Retorne $[a]$.
- Obtenha a matriz de forma do tensor de quarta ordem $\boxed{[q] := \frac{1}{3} a : (I \otimes I) - \frac{2}{3} (\sigma \otimes I)}$;
- Retorne a matriz $\boxed{K_{uu} = G^T [a] G (\det J) wh + G^T [q] (G_0 - G) (\det J) wh}$.

Para finalizar é necessário montar o problema de minimização de resíduos segundo a Eq. (2.43) e retornar ao Abaqus o vetor resíduo RHS e a matriz rigidez A_{matrx}, esses dois correspondem, respectivamente, ao resíduo R e a matriz `K` descritas na equação referenciada. Para realizar a assemblagem das matrizes RHS e A_{matrx}, é necessário alocar nelas os resíduos R_u calculados globalmente e a matriz tangente K_{uu} para todos os elementos da malha.

- Retorne ao Abaqus o resíduo RHS;
  - Aloque os elementos $R_u \rightarrow RHS$.
- Retorne ao Abaqus a matriz rigidez A_{matrx};
  - Aloque os elementos $K_{uu} \rightarrow A_{matrx}$.
- Fim da rotina UEL.

Com a montagem do problema de resíduos, o *solver* do próprio programa inicia o processo de iteração até que a solução seja obtida.

3.2.2 IMPLEMENTAÇÃO DA UEL

Na seção anterior foi discutido o funcionamento da rotina UEL, já esta seção abordará o procedimento para a implementação dela.

A partir do código desenvolvido é necessário indicar ao programa Abaqus para que ele utilize o novo elemento F-bar e não o disponível por ele. Para isso, é necessário algumas alterar linhas de comando do *"input file"* do programa. O *input file* consiste no arquivo que contém as informações sobre a modelagem do problema, nele estão definidos parâmetros como:

- Definição do modelo (nós e suas coordenadas);
- Definição do tipo de elemento;

- Características do material;
- Determinação das condições de contorno e carregamentos;
- Descrição dos *Steps*;
- Indicação dos *Output Requests* (parâmetros de saída).

Então, para a implementação, deve-se realizar algumas alterações neste arquivo de entrada de um elemento padrão do Abaqus (CP4E ou CPE4H, por exemplo). Primeiro é preciso substituir a linha de comando que define o tipo de elemento pela linha que está indicada na UEL (ver Apêndice C, *"usage"*), conforme a Fig. 21 abaixo.

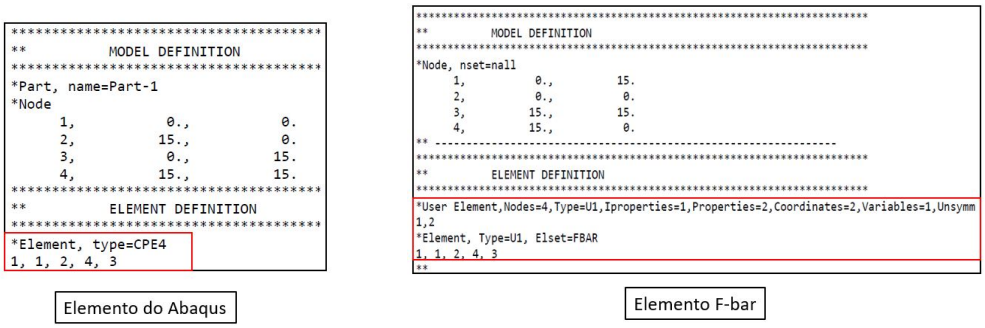


Figura 21 – Modificação da linha de comando no *input file*

Posteriormente, tem que se construir uma malha auxiliar *dummy*. Essa malha *dummy* compartilha os mesmos nós da malha do elemento introduzido (malha "real") e permite a visualização dos resultados no Abaqus. Desta forma, deve-se apenas copiar as mesmas linhas que definem os nós da malha real, alterando apenas o número de cada nó para não gerar conflitos durante a compilação do arquivo. A Fig. 22 mostra um exemplo de como esse procedimento deve ser realizado.

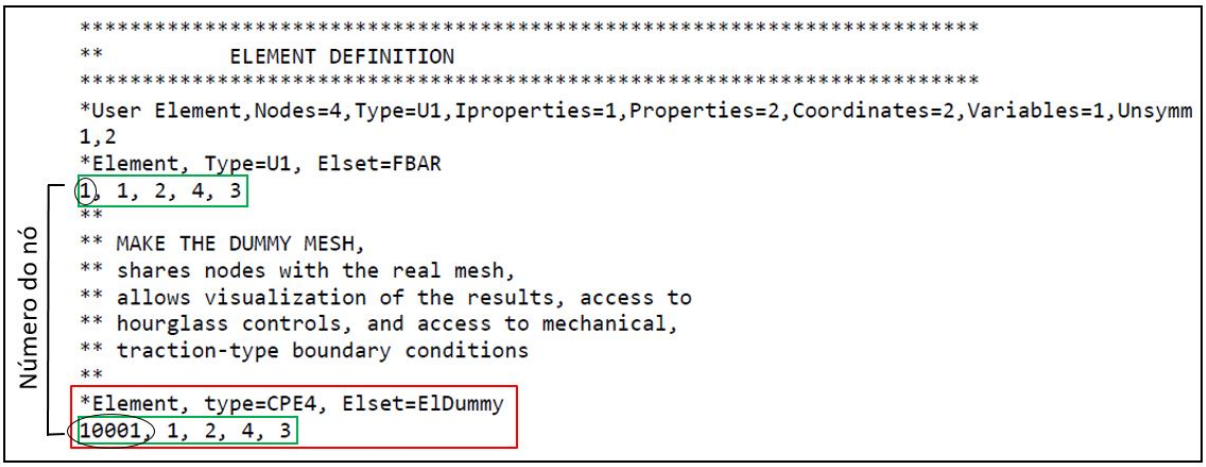


Figura 22 – Exemplo de criação de uma malha *dummy* no *input file*

Para finalizar, é necessário também definir as propriedades do elemento F-bar que, neste caso, devido ao modelo constitutivo utilizado, define-se apenas o módulo de

cisalhamento μ e de incompressibilidade K, além das propriedades do elemento *dummy*. A única propriedade definida para esse último elemento é seu módulo de elasticidade, que deve ser o menor possível, já que essa malha não deve oferecer resistência a nenhum carregamento e só é utilizada para visualização dos resultados da malha real.

Com todas as alterações realizadas no *input file* o elemento está implementado e assim pode-se resolver um problema de interesse modelado. Devido à substituição de um elemento do Abaqus para a utilização da UEL, não é possível utilizar a interface do programa para iniciar o processo de resolução e, por isso, é necessário utilizar a prompt de comando do programa.

No prompt de comando, deve-se inserir a seguinte linha:

abaqus job="**nome do job**" user="**nome da UEL.for**" interactive

conforme o exemplo da Fig. 23.

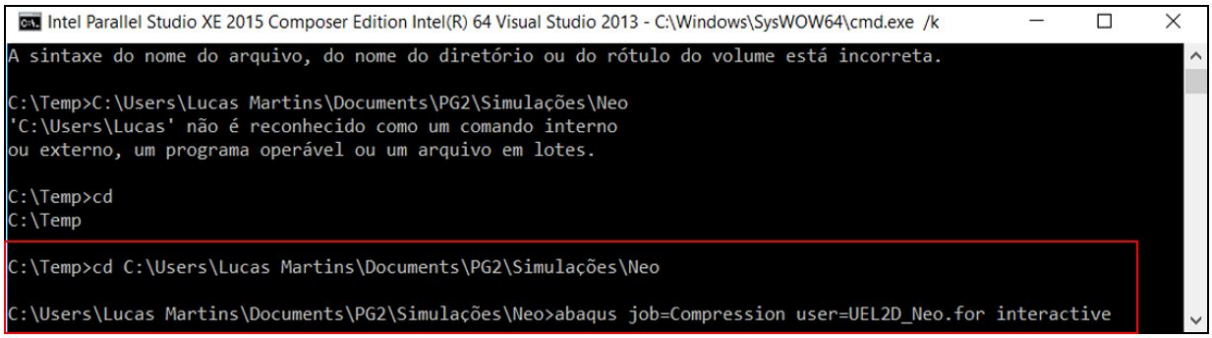


Figura 23 – Chamada da UEL pelo prompt de comando do Abaqus

O termo *interactive* permite o monitoramento da solução pelo prompt de comando. Ao final da simulação, serão criados arquivos com extensão *.odb* e *.dat* que contém os resultados da simulação. Para a visualização no Abaqus basta abrir o arquivo *.odb*.

É importante verificar também se o diretório que contem a rotina UEL e o *input file* são os mesmos. Ambos devem estar na mesma pasta e antes de inserir a linha de chamada no prompt, é preciso alterar o diretório para a pasta que os contém, para isso, utiliza-se o comando ***cd*** seguido do local do diretório.

No Apêndice D há dois arquivos *input files* como exemplos, cada um representa a modelagem de um mesmo problema genérico, o primeiro não possui alteração que permita a inserção do novo elemento F-bar, ou seja, utiliza o elemento do próprio Abaqus e o segundo possui as modificações necessárias para a implementação.

4 ANÁLISE NUMÉRICA

Neste capítulo são apresentados e analisados os resultados referentes à quatro tipos de simulações, uma análise de compressão pura, outra de tração pura, um teste de cisalhamento e uma análise envolvendo contato. Os resultados obtidos em cada análise são comparados entre os três tipos elementos utilizados e com resultados disponíveis na literatura.

4.1 ANÁLISE I - COMPRESSÃO PURA

Esta primeira análise consiste na compressão de um único elemento finito quadrilateral de 4 nós, de aresta 15 mm. As condições de contorno estão ilustradas na Fig. 24. Os nós da aresta inferior têm restrição de movimento vertical e rotação, por sua vez, os nós da aresta lateral esquerda, restrição de movimento horizontal e rotação, por fim, nos nós superiores é aplicado um deslocamento vertical para baixo.

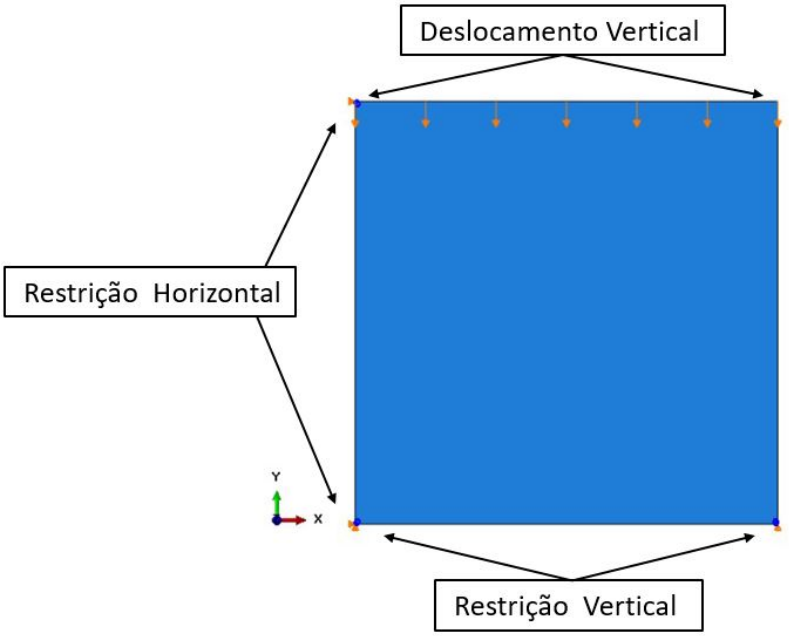


Figura 24 – Condições de contorno para problema de compressão

Devido ao modelo hiperelástico implementado, as propriedades materiais foram definidas para um material elastomérico genérico, desta forma, os módulo de incompres-

sibilidade K e cisalhamento μ são, respectivamente, 1 GPa e 0,3 MPa.

Com o intuito de verificar as limitações de cada elemento e validar a implementação do elemento F-bar, cada um dos elementos foi submetido a diversos valores de compressão, segundo a Tab. 1

Tabela 1 – Valores de compressão das simulações para cada elemento

Tipo de Elemento	Compressão (%)
CP4E	100
CPE4H	70
	90
	95
	100
F-bar	70
	100

O primeiro elemento testado nesta análise foi o elemento de integração plena CP4E. Para esse elemento não foi possível realizar nenhuma simulação de compressão e antes de iniciar a resolução, a operação era abortada. Isso deve-se ao fato de que para os valores de K e μ adotados, obtém-se um coeficiente de Poisson superior ao valor máximo definido pelo programa ($\nu_{max} = 0,495$) para o uso desse tipo de elemento. Levando-se em conta a relação que define o coeficiente de Poisson, temos que para o material definido ν é

$$\nu = \frac{3K - 2\mu}{2(3K - \mu)} = 0,499$$

de fato, maior que o limite estabelecido pelo programa.

Um valor de ν muito próximo a 0,5 caracteriza um material praticamente incompressível e, conforme explicado na Seção 3.1 e reportado pelo próprio *software* ao final da tentativa de simulação, é necessário utilizar o elemento híbrido CPE4H para este tipo de simulação.

A utilização do elemento híbrido CPE4H de fato possibilitou a solução do problema de maneira correta para quase todos os valores de compressão, exceto para 100%. As Figs. 25 e 26 apresentam os resultados obtidos para o deslocamento do elemento nas direções x e y.

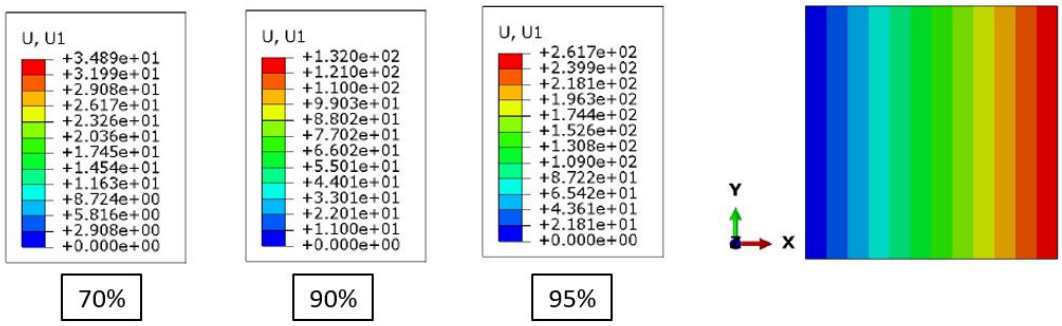


Figura 25 – Deslocamento do **elemento finito híbrido** na direção x (Compressão)

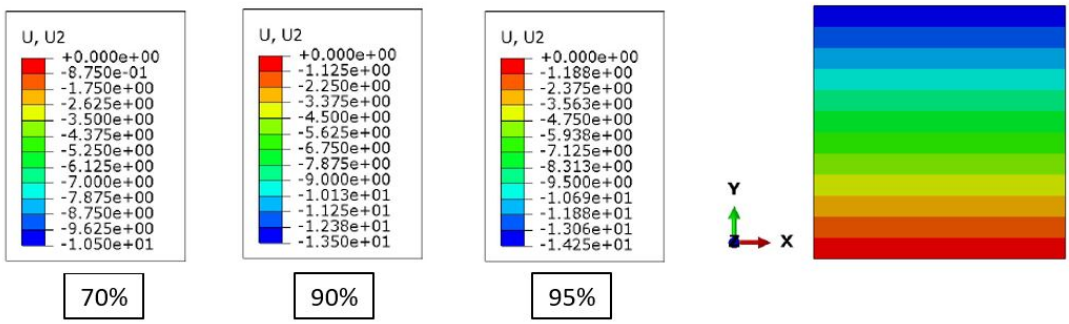
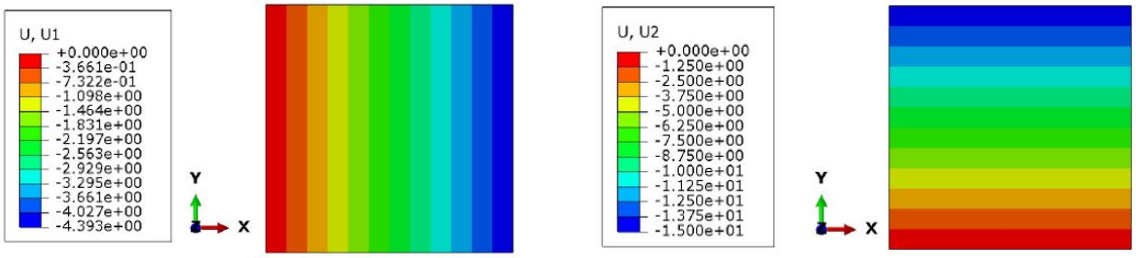


Figura 26 – Deslocamento do **elemento finito híbrido** na direção y (Compressão)

Para 100% de compressão, o resultado encontrado para o deslocamento em ambas as direções se encontra na Fig. 27 abaixo.



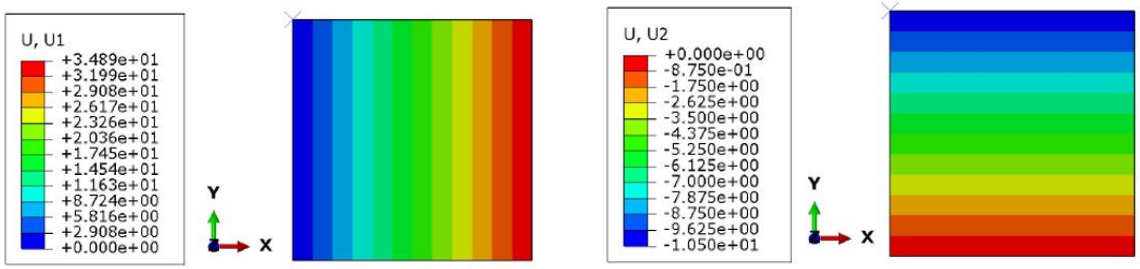
(a) Direção x (b) Direção y

Figura 27 – Deslocamento do **elemento finito híbrido** nas direções x e y (Compressão 100%)

Nas Figs. 25 e 26 percebe-se o correto comportamento do elemento quando ele é comprimido. Como sua face lateral esquerda não pode se deslocar horizontalmente, somente a parte oposta movimenta-se no sentido positivo de x e sua aresta superior para baixo. Nota-se também que quanto maior a compressão, maior é o deslocamento na direção x, como esperado. Além disso, nas três simulações referenciadas nessas Figs. 25 e 26, o elemento foi capaz de se deformar na direção vertical até o valor correspondente à cada compressão (10,5; 13,5 e 14,25 mm).

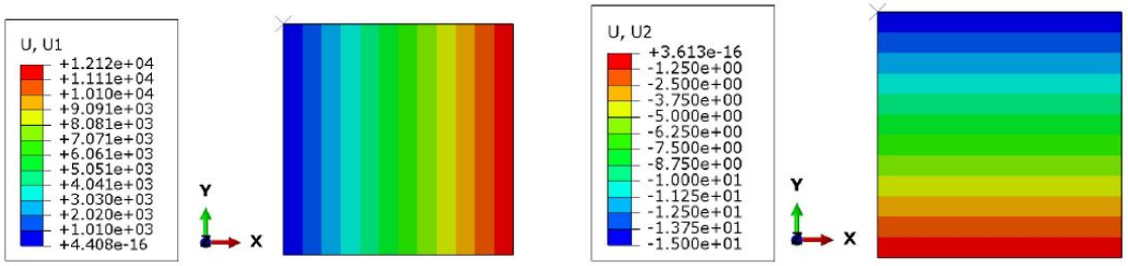
Tendo em vista a Fig. 27, percebe-se que o resultado desta simulação não é correto. Para essa compressão de 100%, o deslocamento apresentado pelo elemento não faz sentido. A solução mostra um deslocamento na direção negativa de x e y, sugerindo que o efeito de Poisson não foi computado. Além disso, o elemento não mostra resistência à mudança de volume, reduzindo-o, ainda que seu material seja praticamente incompressível.

Nas simulações de compressão com o elemento F-bar, ambas apresentaram soluções corretas, inclusive para a compressão de 100% (Figs. 28 e 29).



(a) Direção x (b) Direção y

Figura 28 – Deslocamento do **elemento F-bar** nas direções x e y (Compressão 70%)



(a) Direção x (b) Direção y

Figura 29 – Deslocamento do **elemento F-bar** nas direções x e y (Compressão 100%)

Da Fig. 29 percebe-se que o elemento F-bar desloca-se para baixo e para a direita, descrevendo a deformação esperada. Além disso, para um valor de compressão inferior a 100%, por exemplo, 70%, tanto o elemento F-bar quanto o híbrido apresentaram exatamente o mesmo comportamento.

4.2 ANÁLISE II - TRAÇÃO PURA

Esta segunda análise é semelhante a anterior, entretanto o elemento finito quadrilateral de 4 nós e aresta 15 mm está sob tração. As condições de contorno estabelecidas são as mesmas que as apresentadas na Fig. 24, com a diferença que o deslocamento vertical imposto é para cima. Além disso, as propriedades materiais foram mantidas, ou seja, K equivale a 1 GPa e μ, 0,3 MPa.

Para a verificação das limitações de cada elemento e validação do elemento F-bar, cada um dos elementos foi submetido aos valores de tração diversos apresentados na Tab. 2.

Tabela 2 – Valores de tração aplicados para cada elemento

Tipo de Elemento	Tração (%)
CP4E	100
CPE4H	100
	200
	400
	600
F-bar	100
	600

Novamente o elemento de integração plena CP4E não pode ser utilizado pelo mesmo motivo discutido na Análise I.

Semelhantemente ao ocorrido na análise de compressão, o elemento híbrido CPE4H mostrou-se bastante eficiente quando submetido à tração. A deformação apresentada por esse elemento apresentou comportamento esperado, ou seja, o deslocamento positivo em y promoveu um movimento na direção negativa de x, caracterizando o efeito de Poisson, para todos os valores de tração aplicados. Além disso, nas quatro simulações referenciadas nas Figs. 30 e 31, o elemento foi capaz de se deformar na direção vertical até o valor correspondente à cada tração (15, 30, 60 e 90 mm).

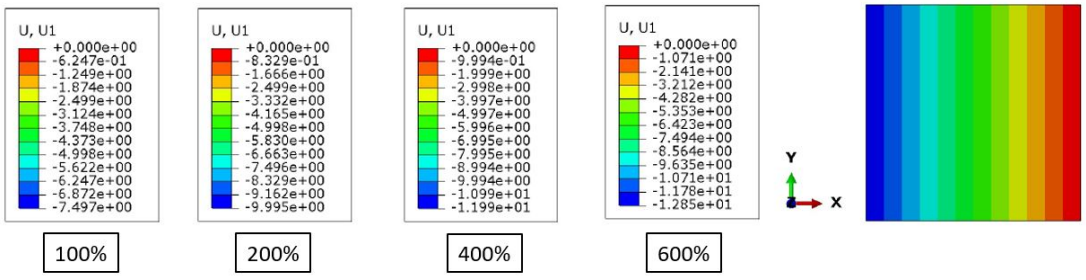


Figura 30 – Deslocamento do **elemento finito híbrido** na direção x (Tração)

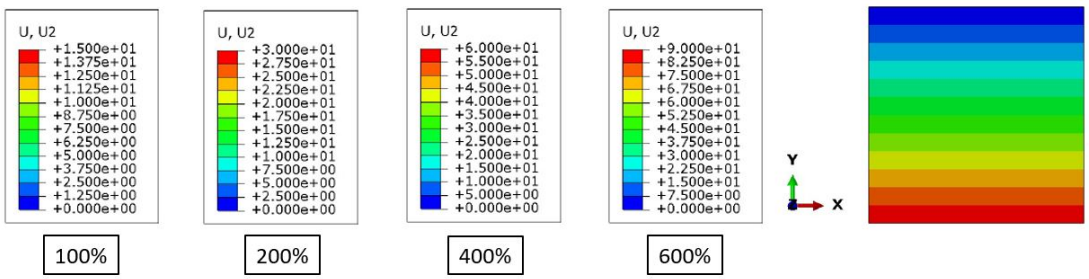


Figura 31 – Deslocamento do **elemento finito híbrido** na direção y (Tração)

Nas simulações utilizando o elemento F-bar, foram encontrados exatamente os mesmos resultados ilustrados nas Figs. 30 e 31 e por isso não serão apresentados novamente. Essa concordância de suas soluções com as obtidas a partir do elemento híbrido afirma também a validade da implementação do elemento F-bar.

4.3 ANÁLISE III - CISALHAMENTO

Definindo ainda o material através do modelo hiperelástico já citado e mantendo as mesmas propriedades (K e μ), uma análise de cisalhamento foi realizada a partir da aplicação de um deslocamento horizontal nos nós da face superior de um elemento finito (Fig. 32). Os deslocamentos aplicados foram de 15, 30 e 90 mm e as simulações foram realizadas apenas com o elemento híbrido e o elemento F-bar, visto que a utilização do elemento de integração plena não convém para essa definição de material, conforme explicado anteriormente.

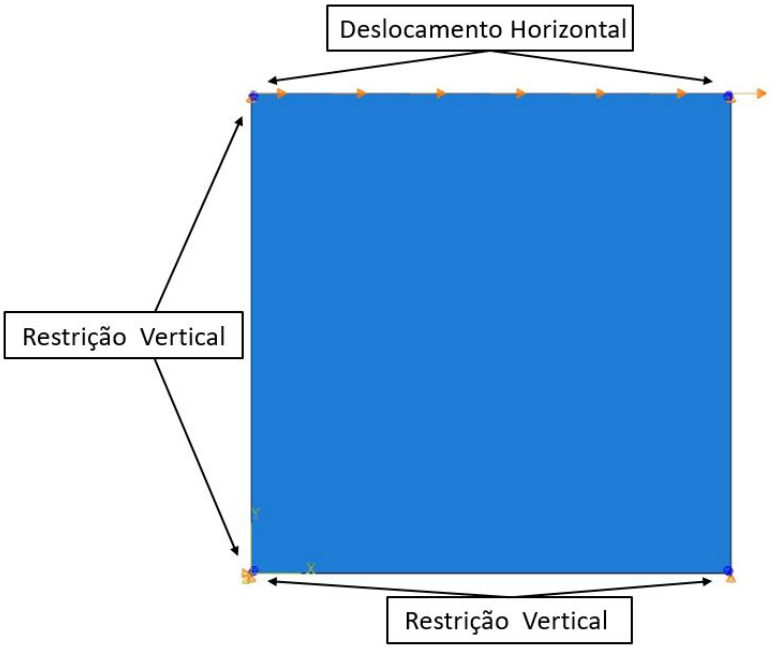
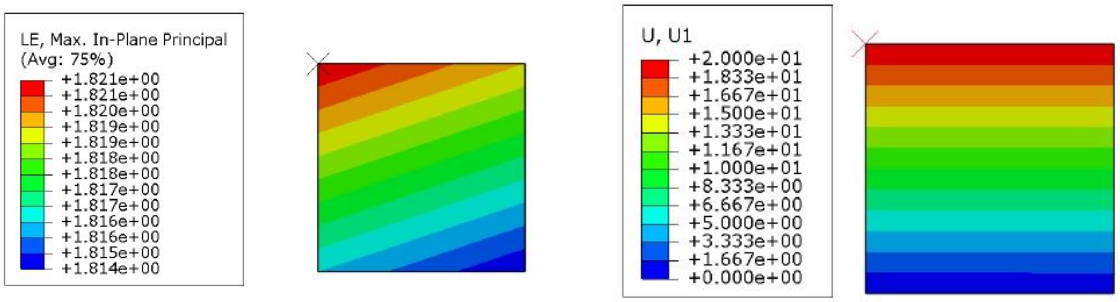


Figura 32 – Condições de contorno da análise de cisalhamento

Mais uma vez, a solução obtida com o uso desses dois elementos foi a mesma. Para os deslocamentos horizontais impostos, ambos computaram rigorosamente a mesma deformação do elemento e não apresentaram problemas de convergência, até mesmo na simulação que envolvia um deslocamento de 90 mm (Fig. 33).



(a) Logaritmo da deformação

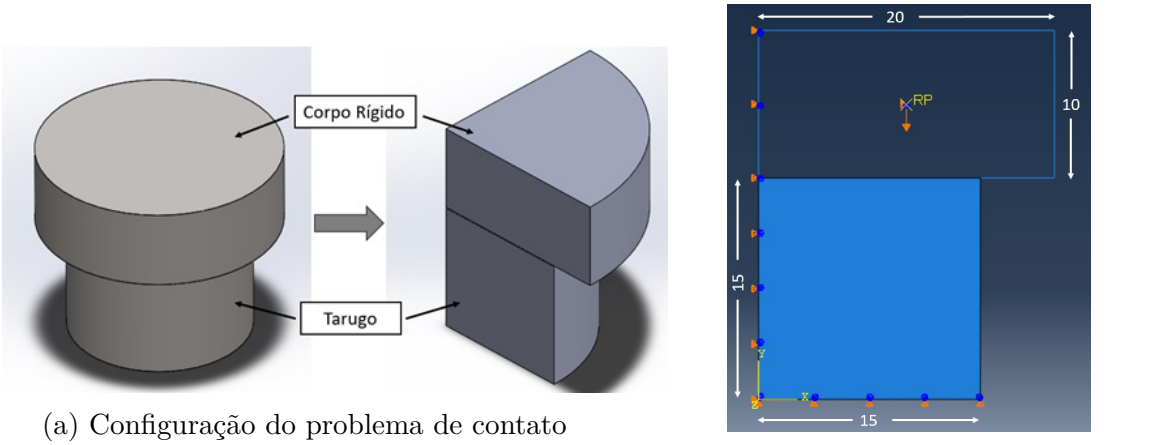
(b) Deslocamento em x

Figura 33 – Elemento **F-bar** na análise de cisalhamento

4.4 ANÁLISE IV - COMPRESSÃO COM CONTATO

O problema analisado nesta seção consiste na compressão de um tarugo cilíndrico por um corpo rígido também cilíndrico. Essa modelagem é semelhante ao problema de *Billet Upsetting* abordado em (NETO; PERIĆ; OWEN, 2008) e (ELFEN, 2004).

A configuração do problema consiste em dois cilindros de alturas e diâmetros diferentes em contato, em que um deslocamento vertical de 9 mm é aplicado no cilindro superior, que é um corpo rígido (Fig. 34(a)). Devido à simetria do problema e tendo em vista que a análise será bidimensional, o problema foi reduzido à um quarto de seu volume, suas dimensões se encontram na Fig. 34(b).



(a) Configuração do problema de contato

(b) Dimensões do problema de contato

Figura 34 – Modelagem do problema de contato

O material do tarugo é definido como aço SAE 1020 e suas propriedades estão sintetizadas na Tab. 3.

Tabela 3 – Propriedades do Aço SAE 1020

Propriedade	Medida
Módulo de Elasticidade $[GPa]$	200
Coeficiente de Poisson	0,3
Massa Específica $[kg/m^3]$	7800
Tensão de Escoamento $[MPa]$	275

Conforme realizado em (ELFEN, 2004), também foi introduzido uma curva de encruamento (Tab. 4) para garantir que um regime elastoplástico (ver Seção 2.5).

Tabela 4 – Curva de Encruamento

Deformação Plástica	Tensão $[MPa]$
0,01	300,2
0,05	366,4
0,1	418,8
0,2	488,5
0,3	537,9
0,4	577,0
0,5	609,9
0,6	638,4
0,7	663,7
0,8	686,5

Para o corpo rígido, a única propriedade atribuída a ele foi o módulo de elasticidade e a fim de aproximá-lo a uma estrutura rígida, foi adotado um valor arbitrário e bem elevado (10^{10} MPa).

Na modelagem do contato, o coeficiente de atrito tem valor de $0,25$. A formulação de deslizamento foi definida para deslocamentos finitos (*Finite Sliding*) e o método de discretização na superfície de contato é *Surface-to-surface*. Para o contato normal utilizou-se o tipo *Hard Contact* com a formulação de Lagrange Aumentado. Já no contato tangencial, o método dos multiplicadores de Lagrange foi utilizado. A Tab. 5 resume os parâmetros da modelagem do contato.

Tabela 5 – Propriedades da Modelagem do Contato

Formulação de Deslizamento	*Finite Sliding*
Método de Discretização	*Surface-to-surface*
Formulação do Contato Normal	Lagrange Aumentado
Formulação do Contato Tangencial	Multiplicadores de Lagrange

Para finalizar a modelagem, foi imposto como condições de contorno: simetria da aresta lateral esquerda em relação ao eixo x e simetria da aresta inferior do tarugo em relação ao eixo y.

O primeiro tipo de elemento utilizado foi o elemento de integração plena CP4E. Diferentes tamanhos de malhas para a discretização do tarugo foram utilizadas, malhas com 15x15, 10x10 e 20x20 elementos e em nenhuma simulação foi obtido convergência. Todas as tentativas de resolução apresentaram distorção excessiva em alguns elementos, principalmente, nos que se encontravam na interface de contato e assim não foi possível obter resultados nessas simulações.

Já com o segundo tipo de elemento, o elemento híbrido CPE4H, foram utilizados cinco tamanhos diferentes para a malha que discretiza o tarugo: 10x10, 12x12, 15x15, 17x17 e 20x20 elementos. Novamente, todas as simulações apresentaram distorção excessiva em alguns elementos, principalmente, nos que se encontravam na interface de contato, e não convergiram, exceto na simulação com a malha de 15x15 elementos (Fig. 35).

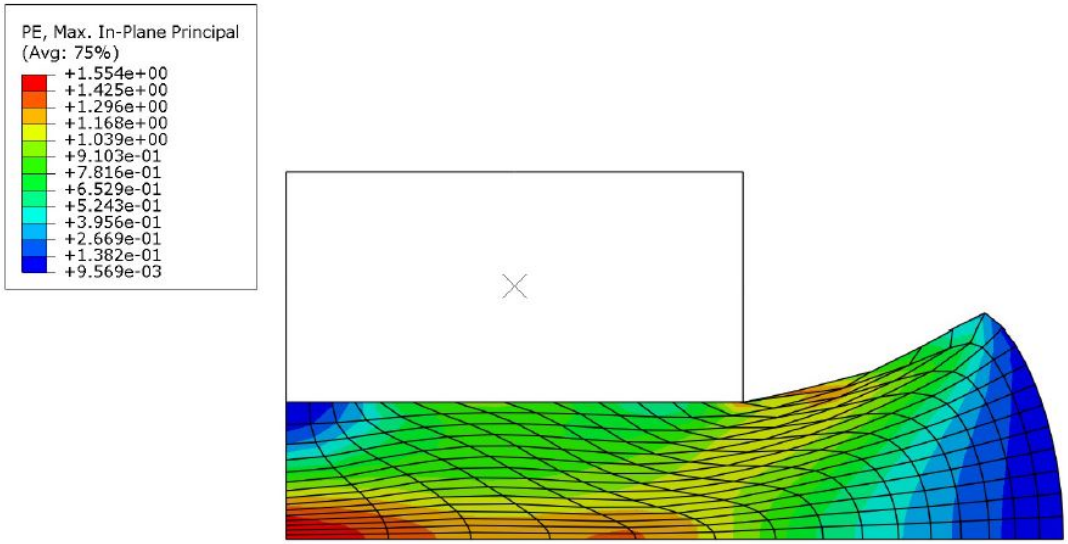


Figura 35 – Deformação plástica para o **elemento finito híbrido** (Malha 15x15 elementos)

Tendo em vista a Fig. 35, foi verificado que os deslocamentos da malha estão de acordo com o resultado obtido em (ELFEN, 2004) mostrado na Fig. 36. Já com relação à deformação plástica, apesar do elemento híbrido apresentar resultados próximos ao esperado da Fig. 36 em quase toda a malha, ele não foi capaz de capturar com grande exatidão a região de maior deformação plástica que se encontra em uma extremidade da interface de contato. Justamente nesse local, esse elemento apresentou elevada distorção e perda de forma.

Além disso, as disparidades entre os dois resultados podem estar relacionadas ao fato de que na simulação realizada em (ELFEN, 2004), diferente da malha utilizada na simulação com o elemento híbrido, utiliza uma malha não estruturada e um mecanismo

de malha adaptativa.

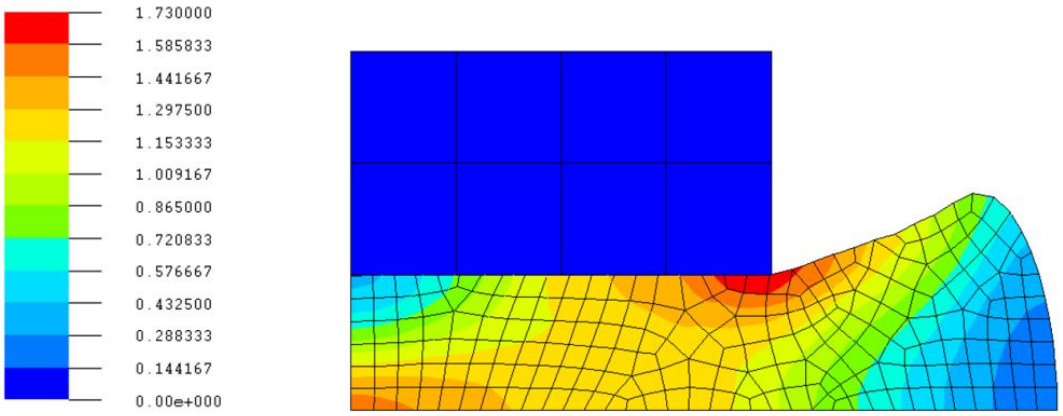
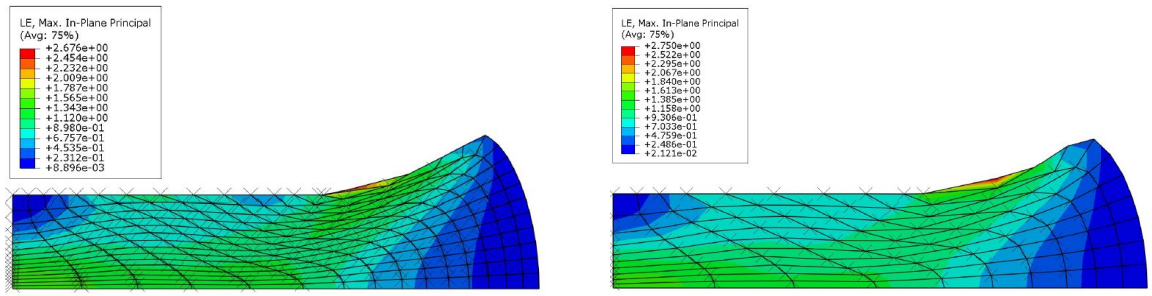


Figura 36 – Deformação plástica no exemplo do Manual Elfen[1]

A simulação com o elemento F-bar foi realizada com malhas de 10x10 e 15x15 elementos e não houve nenhum problema de convergência. Ao contrário do elemento híbrido, o F-bar foi capaz de gerar resultados até mesmo com a malha de menos elementos (10x10). Devido ao uso de um elemento não disponível no Abaqus e a uma restrição do próprio programa, não é possível utilizar sua ferramenta de visualização para exibir todos os dados de saída calculados, por exemplo, tensões equivalentes e nas direções principais. Os únicos parâmetros de saída que podem ser exibidos através da ferramenta de visualização são: o deslocamento nas direções principais e o logaritmo da deformação.

As Figs. 37 e 38 apresentam a comparação entre os resultados obtidos com o elemento F-bar e híbrido. Nota-se novamente que não houve grande disparidade na solução produzido pelos dois elementos e ambos conseguiram reproduzir a deformação do tarugo conforme apresentado em (ELFEN, 2004).



(a) Malha 15x15 elementos

(b) Malha 10x10 elementos

Figura 37 – Logaritmo da deformação para o **elemento F-bar**

[1](ELFEN, 2004)

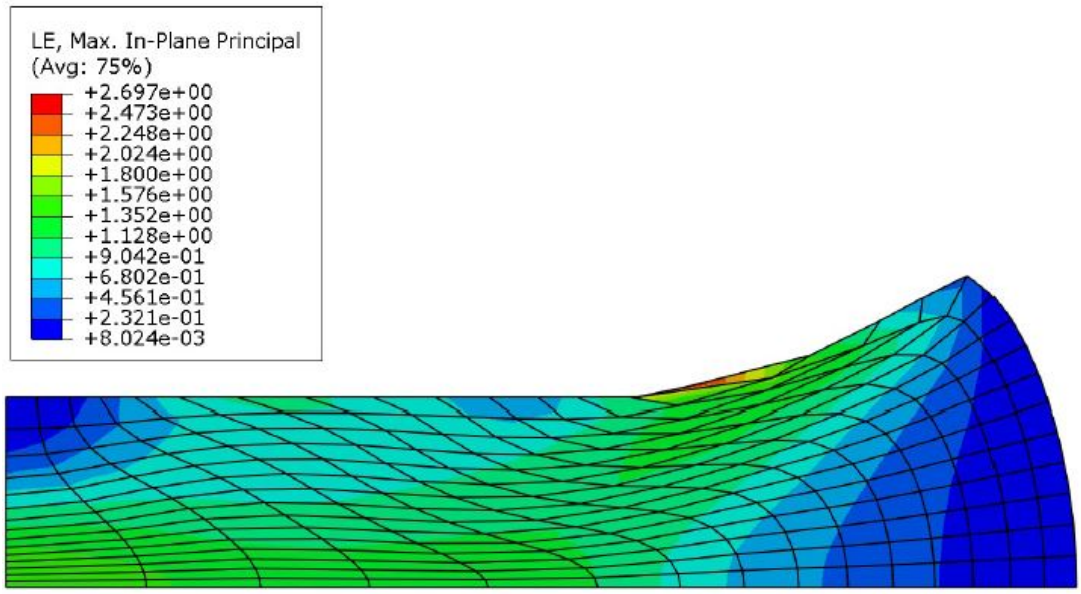


Figura 38 – Logaritmo da deformação para o **elemento híbrido** (Malha 15x15 elementos)

5 CONCLUSÕES

Ao longo deste trabalho buscou-se introduzir o elemento F-bar no programa Abaqus. Para isso, foi necessário utilizar-se da rotina UEL inicialmente desenvolvida por (HENANN; CHESTER; BERTOLDI, 2013), modificando e a adaptando à necessidade do trabalho. Foi constatado que a implementação de uma rotina UEL no Abaqus, além do procedimento de cálculo característico do novo elemento a ser introduzido, requer também a definição do modelo material constitutivo.

Desta forma, além do processo de modificação do gradiente de deformações, descrito na Seção 2.3, foi necessário introduzir no código as equações referentes ao modelo hiperelástico neo-Hookean (Seção 2.4) e suas implicações na computação da matriz rigidez tangente $\mathtt{K}$ e no vetor de força interna $\mathtt{F}^{INT}$.

Para validar e verificar o funcionamento do elemento F-bar implementado, quatro análises foram realizadas, uma de compressão, outra de tração, um teste de cisalhamento e um envolvendo contato. Nessas análises, foi verificado a capacidade que esse novo elemento tem de computar grandes deformações sem apresentar problemas de convergência e ainda produzir resultados adequados. Além disso, foi constatado também a concordância entre os resultados do elemento híbrido e do F-bar. Com exceção de uma simulação (compressão 100%), as soluções obtidas com o F-bar foram muito próximas às obtidas através do elemento híbrido.

Diferentemente do elemento F-bar, o elemento híbrido mostrou-se bastante instável na simulação de contato, ainda que ele tenha produzido solução para uma malha de 15x15 elementos, em malhas ligeiramente menores ou um pouco mais refinadas, esse apresentou problemas de convergência. Apesar da similaridade entre as soluções com o elemento F-bar e híbrido não mostrar vantagem clara em termos de qualidade de resultados, o F-bar apresentou vantagem em estabilidade e robustez.

Haja vista que uma das propostas para trabalhos futuros consiste na extensão da implementação do elemento F-bar para problemas tridimensionais, é esperado que nas simulações em 3D esse elemento demonstre de fato suas vantagens e justifique sua utilização.

Além disso, a impossibilidade de exibição através da ferramenta de pós-processamento do Abaqus das variáveis de saída calculadas (e.g., tensão e área de contato) em simulações que não utilizam elementos do próprio programa é uma deficiência a ser explorada em

trabalhos futuros. No entanto, já existem estudos na literatura, como em (ROTH et al., 2012), que descrevem ferramentas para leitura e visualização dessas informações.

Tendo em vista a capacidade do elemento F-bar em contornar problemas que envolvem grandes deformações e sua característica de reduzir modos espúrios, como *volumetric locking* e *Hourglassing* (NETO et al., 1996), espera-se que a implementação desse elemento no Abaqus por meio da rotina UEL possibilite a resolução de problemas complexos de engenharia que envolvem, por exemplo, plasticidade e contato. Ademais, a ausência de variáveis internas na formulação desse elemento permite também que ele seja utilizado em conjunto com ferramentas de malha adaptativa (NETO et al., 1996).

Referências

AZEVEDO Álvaro F. M. *Método dos Elementos Finitos*. [S.l.: s.n.], 2003. Citado 7 vezes nas páginas 1, 5, 6, 10, 11, 12 e 13.

COOK, R. D. et al. *Concepts and Applications of Finite Elements Analysis*. [S.l.: s.n.], 2002. Citado 3 vezes nas páginas 4, 9 e 11.

DOCUMENTATION, A. *Dassault Systemes Abaqus 6.14*. [S.l.: s.n.], 2014. Citado 3 vezes nas páginas 29, 30 e 31.

ELFEN, M. *Elfen Worked Example*. [S.l.: s.n.], 2004. Citado 4 vezes nas páginas 46, 47, 48 e 49.

FILHO, E. L. S. de A. N. *Estudo de Problemas de Contato com Atrito Seco Via Método dos Elementos Finitos*. Projeto de Graduação 2, Junho 2014. Citado na página 1.

GENT, A. N. A new constitutive relation for rubber. *Rubber Chemistry and Technology*, 1996. Citado na página 32.

HENANN, D. L.; CHESTER, S. A.; BERTOLDI, K. Modeling of dielectric elastomers: Design of actuators and energy harvesting devices. *Journal of the Mechanics and Physics of Solids*, 2013. Citado 3 vezes nas páginas 19, 32 e 51.

HORGAN, C. O. The remarkable gent constitutive model for hyperelastic materials. *International Journal of Non-Linear Mechanics*, 2014. Citado na página 32.

HUGHES, T. J.; TAYLOR, R. L.; SACKMAN, J. L. *Finite Element Formulation and Solution of Contact-impact Problems in Continuum Mechanics – III*. [S.l.: s.n.], 1975. Citado na página 2.

JORGE, R. M. N.; DINIS, L. M. J. S. *Teoria da Plasticidade*. [S.l.: s.n.], 2004. Citado na página 27.

MORAN, B.; ORTIZ, M.; SHIH, F. Formulation of implicit finite element methods for multiplicative finite deformation plasticity. *International Journal for Numerical Methods in Engineering*, 1990. Citado 2 vezes nas páginas 2 e 22.

NETO, E. A. de S.; PERIĆ, D.; OWEN, D. R. J. *Computational Methods for Plasticity*. [S.l.: s.n.], 2008. Citado 10 vezes nas páginas 2, 14, 15, 16, 17, 19, 20, 24, 25 e 46.

NETO, E. de S. et al. Design of simple low order finite elements for large strain analysis of nearly incompressible solids. *International Journal of Solids and Structures*, 1996. Citado 7 vezes nas páginas 2, 13, 19, 20, 22, 23 e 52.

OGDEN, R. *Non-linear Elastic Deformations*. [S.l.: s.n.], 1986. Citado na página 24.

ROTH, S. et al. Visualisation of user defined finite elements with abaqus/viewer. *GACM Report*, 2012. Citado na página 52.

SIMO, J.; TAYLOR, R. L.; PISTER, K. S. Variational and projection methods for the volume constraint in finite deformation elasto-plasticity. *Computer Methods in Applied Mechanics and Engineering*, 1985. Citado 3 vezes nas páginas 2, 18 e 22.

SUSSMAN, T.; BATHE, K. J. *A Finite Element Formulation for Nonlinear Incompressible Elastic and Inelastic Analysis*. [S.l.: s.n.], 1987. Citado na página 2.

TIMOSHENKO, S. P.; GOODIER, J. N. *Theory of Elasticity*. [S.l.: s.n.], 1988. Citado na página 4.

ZIENKIEWICZ, O.; TAYLOR, R. L. *The Finite Element Method*. [S.l.: s.n.], 1988. Citado 3 vezes nas páginas 9, 10 e 11.

APÊNDICES

A MÓDULO TANGENTE ESPACIAL

O módulo tangente espacial $[a]$ é definido:

$$a_{ijkl} = \frac{1}{\det \boldsymbol{F}} \boldsymbol{F}_{jp} \boldsymbol{F}_{tq} A_{ipkq}$$

em que A_{ipkq} é o modulo tangente material, obtido a partir da derivação parcial do primeiro tensor de Piola-Kirchhoff:

$$A_{ipkq} = \frac{\partial P_{ip}}{\partial \boldsymbol{F}_{kq}} =$$

$$= \det \boldsymbol{F}^{-\frac{2}{3}} \mu \left(-\frac{2}{3} F_{ip} {F_{qk}}^{-1} + \frac{2}{9} I_1^* {F_{pi}}^{-1} {F_{qk}}^{-1} + I_{ik} I_{pq} + \frac{1}{3} I_1^* {F_{qi}}^{-1} {F_{pk}}^{-1} - \frac{2}{3} {F_{pi}}^{-1} F_{kq} \right)$$
$$+ \det \boldsymbol{F}\, K + \left((\det \boldsymbol{F} - 1) {F_{pi}}^{-1} {F_{kq}}^{-1} + \det \boldsymbol{F}\, {F_{pi}}^{-1} {F_{qk}}^{-1} - (\det \boldsymbol{F} - 1) {F_{qi}}^{-1} {F_{pk}}^{-1} \right)$$

B OBTENÇÃO DO PRIMEIRO TENSOR DE PIOLA-KIRCHHOFF

O primeiro tensor de Piola-Kirchhoff é definido como:

$$P = P(\boldsymbol{F}) = \bar{\rho}\,\frac{\partial \psi(\boldsymbol{F})}{\partial \psi \boldsymbol{F}}$$

em que a função de energia ψ para o modelo neo-Hookean é:

$$\bar{\rho}\,\psi = \frac{1}{2}\mu(\mathrm{tr}\,\mathtt{B}_{iso} - 3) + \frac{1}{2}K(\ln \det \boldsymbol{F})^2$$

derivando parcialmente a função de energia ψ em relação a F

$$\bar{\rho}\,\frac{\partial \psi}{\partial \boldsymbol{F}} = \frac{1}{2}\mu\,\underbrace{\frac{\partial}{\partial \boldsymbol{F}}(\mathrm{tr}\,\mathtt{B}_{iso} - 3)}_{\mathbf{A1}} + \frac{1}{2}K\,\underbrace{\frac{\partial}{\partial \boldsymbol{F}}(\ln \det \boldsymbol{F})^2}_{\mathbf{A2}}$$

Desenvolvendo **A1**:

$$\frac{\partial}{\partial \boldsymbol{F}}(\mathrm{tr}\,\mathtt{B}_{iso} - 3) = \mathtt{B}_{iso}\frac{\partial \mathtt{B}_{iso}}{\partial \boldsymbol{F}}$$

$$\frac{\partial \mathtt{B}_{iso}}{\partial \boldsymbol{F}} = \frac{\partial}{\partial \boldsymbol{F}}\left(\det \boldsymbol{F}^{-\frac{2}{3}}\boldsymbol{F}\boldsymbol{F}^{-T}\right)$$

$$\frac{\partial}{\partial \boldsymbol{F}}(\det \boldsymbol{F}^{\frac{2}{3}}) = -\frac{2}{3}\left(\det \boldsymbol{F}^{-\frac{5}{3}}\det \boldsymbol{F}\boldsymbol{F}^{-T}\right)$$

$$\frac{\partial(F)}{\partial \boldsymbol{F}} = \frac{\partial(F^{-T})}{\partial \boldsymbol{F}} = I$$

Assim, **A1**:

$$\mathbf{A1} = \det \boldsymbol{F}^{-\frac{2}{3}}\boldsymbol{F}\boldsymbol{F}^{T}\left(-\frac{2}{3}\det \boldsymbol{F}^{-\frac{5}{3}}\det \boldsymbol{F}\boldsymbol{F}^{-T}\right)\boldsymbol{F}\boldsymbol{F}^{T} + \det \boldsymbol{F}^{-\frac{2}{3}}\frac{\partial F}{\partial \boldsymbol{F}}\boldsymbol{F}^{T} + \det \boldsymbol{F}^{-\frac{2}{3}}\boldsymbol{F}\frac{\partial F^{T}}{\partial \boldsymbol{F}}$$

$$= \det \boldsymbol{F}^{-\frac{2}{3}}\boldsymbol{F}\boldsymbol{F}^{T}\left(-\frac{2}{3}\det \boldsymbol{F}^{-\frac{2}{3}}\boldsymbol{F}^{-T}\right)\boldsymbol{F}\boldsymbol{F}^{T} + \det \boldsymbol{F}^{-\frac{2}{3}}\,I\,\boldsymbol{F}^{T} + \det \boldsymbol{F}^{-\frac{2}{3}}\boldsymbol{F}\,I$$

$$= \det \boldsymbol{F}^{-\frac{2}{3}} \boldsymbol{F}\boldsymbol{F}^T \left(-\frac{2}{3} \det \boldsymbol{F}^{-\frac{2}{3}} \boldsymbol{F} \right) \; + \; \det \boldsymbol{F}^{-\frac{2}{3}} \, \boldsymbol{F}^T \; + \; \det \boldsymbol{F}^{-\frac{2}{3}} \boldsymbol{F}$$

$$\mathbf{A1} \; = \; \det \boldsymbol{F}^{-\frac{2}{3}} \boldsymbol{F}\boldsymbol{F}^T \left(\det \boldsymbol{F}^{-\frac{2}{3}} (\frac{1}{3} \boldsymbol{F} \; + \; \boldsymbol{F}^T) \right)$$

Agora desenvolve-se **A2**:

$$\frac{\partial}{\partial \boldsymbol{F}} (\ln \det \boldsymbol{F})^2 \; = \; 2 \, \boldsymbol{F}^{-1} \; \det \boldsymbol{F} \, \boldsymbol{F}^{-T}$$

Finalmente temos que o primeiro tensor de Piola-Kirchhoff é:

$$P(\boldsymbol{F}) = \; \bar{\rho} \, \frac{\partial \psi(\boldsymbol{F})}{\partial \psi \boldsymbol{F}} = \frac{1}{2} \mu \left(\det \boldsymbol{F}^{-\frac{2}{3}} \boldsymbol{F}\boldsymbol{F}^T \right) \left(\det \boldsymbol{F}^{-\frac{2}{3}} (\frac{1}{3} \boldsymbol{F} \; + \; \boldsymbol{F}^T) \right) \; + \; \frac{1}{2} K \left(2 \, \boldsymbol{F}^{-1} \; \det \boldsymbol{F} \, \boldsymbol{F}^{-T} \right)$$

C SUBROTINA UEL

```
!**********************************************************************
! User element (UEL) for large-deformation Neo Hookean hyperelasticity in
! two-dimensions -- plane-strain and axisymmetric
!**********************************************************************
! Element details:
!**********************************************************************
!
! Solution variables (or nodal variables) are the displacements (DOFs 1-2)
!
! Material behavior is modelled as Neo Hookean hyperelastic
!
! This subroutine is for a two-dimensional 4-node isoparametric
!  quadrilateral element as shown below with 4pt (full) or 1pt
!  (reduced) integration, as well as plane-strain or
!  axisymmetric settings.
!
! In order to avoid locking for the fully-integrated element, we
!  use the F-bar method of de Souza Neto (1996).
!
!  Mechanical, traction- and pressure-type boundary conditions
!  may be applied to the dummy mesh using the Abaqus built-in
!  commands *Dload or *Dsload.
!
!
!              A eta (=xi_2)
!  4-node      |
!   quad       | Face 3
!        4-----------3
!        |     |     |
!        |     |     |
! Face 4 |     ------|---> xi (=xi_1)
!        |           |
!        |           |  Face 2
!        1-----------2
!            Face 1
!
! Shawn A. Chester, July 2011
! David L. Henann,  August 2011
! Thiago Doca,      September 2017
!

!**********************************************************************
! Usage:
!**********************************************************************
!
! User element statement in the input file:
!  *User Element,Nodes=4,Type=U1,Iproperties=1,Properties=2,Coordinates=2,Variables=1,Unsymm
!  1,2
!
! Note: No local state variables are used in this element, thus we may set the above
!  parameter 'Variables' to any non-zero integer.
!
! In the subroutine UEL, set 'nInt' = number of integration points
!  Options are nInt=4 (full integration) or nInt=1 (reduced integration)
!
! In the input file, set the parameter pe=1 for plane-strain or
!  pe=0 for axisymmetric
!
!
```

```
!     Material Properties Vector
!     --------------------------------------------------------------
!     Gshear  = props(1)  ! Shear modulus
!     Kbulk   = props(2)  ! Bulk modulus
!     pe      = jprops(1) ! Plane strain=1, axisymmetric=0
!
!**********************************************************************

      subroutine UEL(rhs,amatrx,svars,energy,ndofel,nrhs,nsvars,
     +     props,nprops,coords,mcrd,nnode,uall,duall,vel,accn,jtype,
     +     time,dtime,kstep,kinc,jelem,params,ndload,jdltyp,adlmag,
     +     predef,npredf,lflags,mlvarx,ddlmag,mdload,pnewdt,jprops,
     +     njprop,period)

      implicit none
      !
      ! variables defined in uel, passed back to Abaqus
      !
      real*8 rhs(mlvarx,*),amatrx(ndofel,ndofel),svars(*),energy(8),
     +  pnewdt
      !
      ! variables passed into UEL
      !
      integer ndofel,nrhs,nsvars,nprops,mcrd,nnode,jtype,kstep,kinc,
     +  jelem,ndload,jdltyp(mdload,*),npredf,lflags(*),mlvarx,mdload,
     +  jprops(*),njprop
      !
      real*8 props(*),coords(mcrd,nnode),uall(ndofel),duall(mlvarx,*),
     +  vel(ndofel),accn(ndofel),time(2),dtime,params(*),
     +  adlmag(mdload,*),predef(2,npredf,nnode),ddlmag(mdload,*),period
      !
      ! variables defined and used in the UEL
      !
      integer i,j,k,A11,B11,A12,B12,nInt,nIntPt,intpt,nDim,stat,pe,face
      !
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      !
      parameter(nDim=2) ! number of spatial dimensions, do not change
      parameter(nInt=4) ! number of integration points
      !
      ! nInt=4: fully-integrated, nInt=1: reduced integration
      !
      ! When reduced integration is selected, be sure to
      !  specify an hourglass stiffness of about 0.005G
      !  for the dummy mesh in the input file.  Also, change
      !  the dummy mesh element to reduced.
      !
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      !
      real*8 u(nNode,2),du(nNode,ndofel),coordsC(mcrd,nNode),
     +  Ru(2*nNode,1),Kuu(2*nNode,2*nNode),
     +  Iden(3,3),xi(nInt,2),w(nInt),sh0(nNode),sh(nNode),dsh0(nNode,2),
     +  dshC0(nNode,2),dsh(nNode,2),dshC(nNode,2),dshxi(nNode,2),
     +  detMapJ0,detMapJ0C,detMapJ,detMapJC,Fc_tau(3,3),detFc_tau,
     +  F_tau(3,3),detF_tau,T_tau(3,3),SpTanMod(3,3,3,3),
     +  Smat(3,1),Bmat(3,2*nNode),Gmat(4,2*nNode),
     +  G0mat(4,2*nNode),Amat(4,4),Qmat(4,4),
     +  Gmatphi(2,nNode),body(3),BodyForceRes(2*nNode,1),
     +  SmatAx(4,1),BmatAx(4,2*nNode),GmatAx(5,2*nNode),
```

```
     +  G0MatAx(5,2*nNode),AmatAx(5,5),QmatAx(5,5),
     +  BodyForceResAx(2*nNode,1),
     +  AR,AR0,ARc,Le
      !
      real*8 zero,one,two,half,Pi,three,third
      parameter(zero=0.d0,one=1.d0,two=2.d0,half=0.5d0,Pi=3.141592653d0,
     +     three=3.d0,third=1.d0/3.d0)


      ! Check the procedure type; this should be a
      !  *Static step, which is either 1 or 2
      !
      if((lflags(1).eq.1).or.(lflags(1).eq.2)) then
         !
         ! correct procedure specified
         !
      else
         write(*,*) 'Abaqus does not have the right procedure'
         write(*,*) 'go back and check the procedure type'
         write(*,*) 'lflags(1)=',lflags(1)
         call xit
      endif


      ! Make sure Abaqus knows you are doing a large
      !  deformation problem
      !
      if(lflags(2).eq.0) then
         !
         ! lflags(2)=0 -> small disp.
         ! lflags(2)=1 -> large disp.
         !
         write(*,*) 'Abaqus thinks you are doing'
         write(*,*) 'a small displacement analysis'
         write(*,*) 'go in and set nlgeom=yes'
         call xit
      endif


      ! Do nothing if a ``dummy'' step
      !
      if(dtime.eq.zero) return


      ! Get flag for plane strain or axisymmetric
      !
      pe = jprops(1)


      ! Identity tensor
      !
      call onem(Iden)


      ! Initialize the residual and tangent matrices to zero.
      !
      Ru  = zero
      Kuu = zero
      Energy = zero
```

```
      ! Obtain nodal displacements and potentials
      !
      k = 0
      do i=1,nNode
         do j=1,nDim
            k = k + 1
            u(i,j) = Uall(k)
            du(i,j) = DUall(k,1)
         enddo
      enddo


      ! Obtain current nodal coordinates
      !
      do i=1,nNode
         do j=1,nDim
            coordsC(j,i) = coords(j,i) + u(i,j)
         enddo
      enddo


      ! Impose the heuristic restriction that the displacement
      !  increment is less than ten times the element diagonal,
      !  which is taken as an approximate element size.
      !
      Le = dsqrt(((coordsC(1,1)-coordsC(1,3))**two) +
     +                    ((coordsC(2,1)-coordsC(2,3))**two))
      !
      do i=1,nNode
         do j=1,nDim
            if(dabs(du(i,j)).gt.10.d0*Le) then
               pnewdt = 0.5
               return
            endif
         enddo
      enddo


      !------------------------------------------------------------
      !
      ! Take this opportunity to perform calculations at the element
      !  centroid.  Get the deformation gradient for use in the 'F-bar' method.
      !
      ! Reference for the F-bar method:
      !  de Souza Neto, E.A., Peric, D., Dutko, M., Owen, D.R.J., 1996.
      !  Design of simple low order finite elements for large strain
      !  analysis of nearly incompressible solids. International Journal
      !  of Solids and Structures 33, 3277-3296.
      !
      ! Obtain shape functions and their local gradients at the element
      !  centroid, that means xi=eta=zeta=0.0, and nIntPt=1
      !
      if(nNode.eq.4) then
         call calcShape2DLinear(1,zero,1,sh0,dshxi)
      else
         write(*,*) 'Incorrect number of nodes: nNode.ne.4'
         call xit
```

```
      endif


      ! Map shape functions from local to global reference coordinate system
      !
      if(mcrd.eq.2) then
         call mapShape2Da(nNode,dshxi,coords,dsh0,detMapJ0,stat)
         if(stat.eq.0) then
            pnewdt = 0.5
            return
         endif
      elseif(mcrd.eq.3) then
         call mapShape2D(nNode,dshxi,coords,dsh0,detMapJ0,stat)
         if(stat.eq.0) then
            pnewdt = 0.5
            return
         endif
      else
         ! big problem
         write(*,*) 'Unexpected error, mcrd=',mcrd
         call xit
      endif


      ! Map shape functions from local to global current coordinate system
      !
      if(mcrd.eq.2) then
         call mapShape2Da(nNode,dshxi,coordsC,dshC0,detMapJ0C,stat)
         if(stat.eq.0) then
            pnewdt = 0.5
            return
         endif
      elseif(mcrd.eq.3) then
         call mapShape2D(nNode,dshxi,coordsC,dshC0,detMapJ0C,stat)
         if(stat.eq.0) then
            pnewdt = 0.5
            return
         endif
      else
         ! big problem
         write(*,*) 'Unexpected error, mcrd=',mcrd
         call xit
      endif


      AR0  = one ! for plane-strain you could put the depth here
      ARc  = one !
      if(pe.ne.1) then ! this is axisymmetric
         AR0 = zero
         ARc = zero
         do i=1,nNode
            ! radial coord in ref config at centroid
            AR0 = AR0 + sh0(i)*coords(1,i)
            ! radial coord in current config at centroid
            ARc = ARc + sh0(i)*(coords(1,i) + u(i,1))
         enddo
      endif
```

```
      ! Calculate the deformation gradient at the element centroid
      !  at the end of the increment for use in the 'F-bar' method.
      !  The subscript tau denotes the time at the end of the increment.
      !
      Fc_tau = Iden
      do i=1,nDim
         do j=1,nDim
            do k=1,nNode
               Fc_tau(i,j) = Fc_tau(i,j) + dsh0(k,j)*u(k,i)
            enddo
         enddo
      enddo
      if(pe.ne.1) then
         !
         ! modify for axisymmetric
         !
         Fc_tau(3,3) = ARc/AR0 ! 'hoop' stretch
         !
         ! axisymmetric implementation of detF
         !
         call mdet(Fc_tau,detFc_tau)
      else
         !
         ! modify for plane-strain
         !
         Fc_tau(3,3) = one
         !
         ! 2D plane-strain implementation detF
         !
         detFc_tau = Fc_tau(1,1)*Fc_tau(2,2) - Fc_tau(1,2)*Fc_tau(2,1)
      endif
      !
      ! With the deformation gradient known at the element centroid
      !  we are now able to implement the `F-bar' method later
      !
      !------------------------------------------------------------------




      !------------------------------------------------------------------
      ! Begin the loop over integration points
      !
      ! Obtain integration point local coordinates and weights
      !
      if(nNode.eq.4) then
         !
         ! gauss integration for a rectangular element
         !
         if(nInt.eq.4) then
            call xint2D4pt(xi,w,nIntPt) ! 4-pt integration, nInt=4 above
         elseif(nInt.eq.1) then
            call xint2D1pt(xi,w,nIntPt) ! 1-pt integration, nInt=1 above
         else
            write(*,*) 'Invalid number of int points, nInt=',nInt
            call xit
         endif
      else
         write(*,*) 'Incorrect number of nodes: nNode.ne.4'
```

```
         call xit
      endif



      ! Loop over integration points
      !
      do intpt=1,nIntPt


         ! Obtain shape functions and their local gradients
         !
         if(nNode.eq.4) then
            call calcShape2DLinear(nIntPt,xi,intpt,sh,dshxi)
         else
            write(*,*) 'Incorrect number of nodes: nNode.ne.4'
            call xit
         endif


         ! Map shape functions from local to global reference coordinate system
         !
         if(mcrd.eq.2) then
            call mapShape2Da(nNode,dshxi,coords,dsh,detMapJ,stat)
            if(stat.eq.0) then
               pnewdt = 0.5
               return
            endif
         elseif(mcrd.eq.3) then
            call mapShape2D(nNode,dshxi,coords,dsh,detMapJ,stat)
            if(stat.eq.0) then
               pnewdt = 0.5
               return
            endif
         else
            ! big problem
            write(*,*) 'Unexpected error, mcrd=',mcrd
            call xit
         endif


         ! Map shape functions from local to global current coordinate system
         !
         if(mcrd.eq.2) then
            call mapShape2Da(nNode,dshxi,coordsC,dshC,detMapJC,stat)
            if(stat.eq.0) then
               pnewdt = 0.5
               return
            endif
         elseif(mcrd.eq.3) then
            call mapShape2D(nNode,dshxi,coordsC,dshC,detMapJC,stat)
            if(stat.eq.0) then
               pnewdt = 0.5
               return
            endif
         else
            ! big problem
            write(*,*) 'Unexpected error, mcrd=',mcrd
            call xit
```

```
      endif


      AR0  = one ! for plane-strain you could put the depth here
      AR   = one !
      if(pe.ne.1) then ! this is axisymmetric
         AR0  = zero
         AR   = zero
         do i=1,nNode
            ! radial coord in reference config
            AR0 = AR0 + sh(i)*coords(1,i)
            ! radial coord in current config
            AR  = AR  + sh(i)*(coords(1,i) + u(i,1))
         enddo
         AR0  = two*Pi*AR0
         AR   = two*Pi*AR
      endif


      ! Obtain the deformation gradient at this integration point.
      ! The subscript tau denotes the time at the end of the
      !  increment.
      !
      F_tau = Iden
      do i=1,nDim
         do j=1,nDim
            do k=1,nNode
               F_tau(i,j) = F_tau(i,j) + dsh(k,j)*u(k,i)
            enddo
         enddo
      enddo
      !
      ! modify F(3,3) for plane-strain or axisymmetric
      !
      if(pe.ne.1) then
         !
         ! this is axisymmetric
         !
         F_tau(3,3) = AR/AR0
      else
         !
         ! this is plane strain
         !
         F_tau(3,3) = one
      endif
      !
      ! Modify the deformation gradient for the `F-bar' method
      !  only when using the 4 node fully integrated linear
      !  element, do not use the `F-bar' method for any other element
      !
      if((nNode.eq.4).and.(nInt.eq.4)) then
         if(pe.eq.1) then
            !
            !  2D plane-strain implementation
            !
            detF_tau = F_tau(1,1)*F_tau(2,2) - F_tau(1,2)*F_tau(2,1)
            do i=1,nDim
               do j=1,nDim
                  F_tau(i,j) =((detFc_tau/detF_tau)**half)*F_tau(i,j)
```

```
                  enddo
               enddo
            else
               !
               ! 2D axisymmetric implementation
               !
               call mdet(F_tau,detF_tau)
               F_tau = ((detFc_tau/detF_tau)**third)*F_tau
            endif
         endif


         !@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
         !
         ! Perform the constitutive update at this integ. point
         !
         call NeoHookean(props,nprops,F_tau,T_tau,SpTanMod,stat)
         !
         if(stat.eq.0) then
            pnewdt = 0.5
            return
         endif
         !
         !
         !@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@


         ! Compute/update the displacement residual vector
         !
         if(pe.eq.1) then
            !
            ! this is plane strain
            !
            Smat(1,1) = T_tau(1,1)
            Smat(2,1) = T_tau(2,2)
            Smat(3,1) = T_tau(1,2)
            !
            Bmat = zero
            do k=1,nNode
               Bmat(1,1+nDim*(k-1)) = dshC(k,1)
               Bmat(2,2+nDim*(k-1)) = dshC(k,2)
               Bmat(3,1+nDim*(k-1)) = dshC(k,2)
               Bmat(3,2+nDim*(k-1)) = dshC(k,1)
            enddo
            !
            body = zero ! The body force vector may be specified here
            !
            BodyForceRes = zero
            do k=1,nNode
               BodyForceRes(1+nDim*(k-1),1) = sh(k)*body(1)
               BodyForceRes(2+nDim*(k-1),1) = sh(k)*body(2)
            enddo
            !
            Ru = Ru
     +           - matmul(transpose(Bmat),Smat)*detmapJC*w(intpt)*AR
     +           + BodyForceRes*detmapJC*w(intpt)*AR
         else
            !
            ! this is axisymetric
```

```
            !
            SmatAx(1,1) = T_tau(1,1)
            SmatAx(2,1) = T_tau(2,2)
            SmatAx(3,1) = T_tau(1,2)
            SmatAx(4,1) = T_tau(3,3)
            !
            BmatAx = zero
            do k=1,nNode
               BmatAx(1,1+nDim*(k-1)) = dshC(k,1)
               BmatAx(2,2+nDim*(k-1)) = dshC(k,2)
               BmatAx(3,1+nDim*(k-1)) = dshC(k,2)
               BmatAx(3,2+nDim*(k-1)) = dshC(k,1)
               BmatAx(4,1+nDim*(k-1)) = sh(k)/(AR/(two*Pi))
            enddo
            !
            body = zero ! The body force vector may be specified here
            !
            BodyForceResAx = zero
            do k=1,nNode
               BodyForceResAx(1+nDim*(k-1),1) = sh(k)*body(1)
               BodyForceResAx(2+nDim*(k-1),1) = sh(k)*body(2)
            enddo
            !
            Ru = Ru
+           - matmul(transpose(BmatAx),SmatAx)*detmapJC*w(intpt)*AR
+           + BodyForceResAx*detmapJC*w(intpt)*AR
         endif


         ! Compute/update the displacement tangent matrix
         !
         if(pe.eq.1) then
            !
            ! this is plane strain
            !
            Gmat = zero
            do k=1,nNode
               Gmat(1,1+nDim*(k-1)) = dshC(k,1)
               Gmat(2,2+nDim*(k-1)) = dshC(k,1)
               Gmat(3,1+nDim*(k-1)) = dshC(k,2)
               Gmat(4,2+nDim*(k-1)) = dshC(k,2)
            enddo

            G0mat = zero
            do k=1,nNode
               G0mat(1,1+nDim*(k-1)) = dshC0(k,1)
               G0mat(2,2+nDim*(k-1)) = dshC0(k,1)
               G0mat(3,1+nDim*(k-1)) = dshC0(k,2)
               G0mat(4,2+nDim*(k-1)) = dshC0(k,2)
            enddo
            !
            Amat = zero
            Amat(1,1) = SpTanMod(1,1,1,1)
            Amat(1,2) = SpTanMod(1,1,2,1)
            Amat(1,3) = SpTanMod(1,1,1,2)
            Amat(1,4) = SpTanMod(1,1,2,2)
            Amat(2,1) = SpTanMod(2,1,1,1)
            Amat(2,2) = SpTanMod(2,1,2,1)
            Amat(2,3) = SpTanMod(2,1,1,2)
```

```
            Amat(2,4) = SpTanMod(2,1,2,2)
            Amat(3,1) = SpTanMod(1,2,1,1)
            Amat(3,2) = SpTanMod(1,2,2,1)
            Amat(3,3) = SpTanMod(1,2,1,2)
            Amat(3,4) = SpTanMod(1,2,2,2)
            Amat(4,1) = SpTanMod(2,2,1,1)
            Amat(4,2) = SpTanMod(2,2,2,1)
            Amat(4,3) = SpTanMod(2,2,1,2)
            Amat(4,4) = SpTanMod(2,2,2,2)
            !
            Qmat = zero
            Qmat(1,1) = half*(Amat(1,1)+Amat(1,4)) - half*T_tau(1,1)
            Qmat(2,1) = half*(Amat(2,1)+Amat(2,4)) - half*T_tau(1,2)
            Qmat(3,1) = half*(Amat(3,1)+Amat(3,4)) - half*T_tau(1,2)
            Qmat(4,1) = half*(Amat(4,1)+Amat(4,4)) - half*T_tau(2,2)
            Qmat(1,4) = half*(Amat(1,1)+Amat(1,4)) - half*T_tau(1,1)
            Qmat(2,4) = half*(Amat(2,1)+Amat(2,4)) - half*T_tau(1,2)
            Qmat(3,4) = half*(Amat(3,1)+Amat(3,4)) - half*T_tau(1,2)
            Qmat(4,4) = half*(Amat(4,1)+Amat(4,4)) - half*T_tau(2,2)
            !
            if((nNode.eq.4).and.(nInt.eq.4)) then
               !
               ! This is the tangent using the F-bar method with the
               !  4 node fully integrated linear element
               !
               Kuu = Kuu
     +               + matmul(matmul(transpose(Gmat),Amat),Gmat)
     +               *detMapJC*w(intpt)*AR
     +               +matmul(transpose(Gmat),matmul(Qmat,(G0mat - Gmat)))
     +               *detMapJC*w(intpt)*AR
            else
               !
               ! This is the tangent not using the F-bar method with all
               !  other elements
               !
               Kuu = Kuu
     +               + matmul(matmul(transpose(Gmat),Amat),Gmat)
     +               *detMapJC*w(intpt)*AR
            endif
            !
         else
            !
            ! this is axisymmetric
            !
            GmatAx = zero
            do k=1,nNode
               GmatAx(1,1+nDim*(k-1)) = dshC(k,1)
               GmatAx(2,2+nDim*(k-1)) = dshC(k,1)
               GmatAx(3,1+nDim*(k-1)) = dshC(k,2)
               GmatAx(4,2+nDim*(k-1)) = dshC(k,2)
               GmatAx(5,1+nDim*(k-1)) = sh(k)/(AR/(two*Pi))
            enddo
            !
            G0matAx = zero
            do k=1,nNode
               G0matAx(1,1+nDim*(k-1)) = dshC0(k,1)
               G0matAx(2,2+nDim*(k-1)) = dshC0(k,1)
               G0matAx(3,1+nDim*(k-1)) = dshC0(k,2)
               G0matAx(4,2+nDim*(k-1)) = dshC0(k,2)
```

```
            G0matAx(5,1+nDim*(k-1)) = sh0(k)/ARc
         enddo
         !
         AmatAx = zero
         AmatAx(1,1) = SpTanMod(1,1,1,1)
         AmatAx(1,2) = SpTanMod(1,1,2,1)
         AmatAx(1,3) = SpTanMod(1,1,1,2)
         AmatAx(1,4) = SpTanMod(1,1,2,2)
         AmatAx(1,5) = SpTanMod(1,1,3,3)
         AmatAx(2,1) = SpTanMod(2,1,1,1)
         AmatAx(2,2) = SpTanMod(2,1,2,1)
         AmatAx(2,3) = SpTanMod(2,1,1,2)
         AmatAx(2,4) = SpTanMod(2,1,2,2)
         AmatAx(2,5) = SpTanMod(2,1,3,3)
         AmatAx(3,1) = SpTanMod(1,2,1,1)
         AmatAx(3,2) = SpTanMod(1,2,2,1)
         AmatAx(3,3) = SpTanMod(1,2,1,2)
         AmatAx(3,4) = SpTanMod(1,2,2,2)
         AmatAx(3,5) = SpTanMod(1,2,3,3)
         AmatAx(4,1) = SpTanMod(2,2,1,1)
         AmatAx(4,2) = SpTanMod(2,2,2,1)
         AmatAx(4,3) = SpTanMod(2,2,1,2)
         AmatAx(4,4) = SpTanMod(2,2,2,2)
         AmatAx(4,5) = SpTanMod(2,2,3,3)
         AmatAx(5,1) = SpTanMod(3,3,1,1)
         AmatAx(5,2) = SpTanMod(3,3,2,1)
         AmatAx(5,3) = SpTanMod(3,3,1,2)
         AmatAx(5,4) = SpTanMod(3,3,2,2)
         AmatAx(5,5) = SpTanMod(3,3,3,3)
         !
         QmatAx = zero
         QmatAx(1,1) = third*(AmatAx(1,1)+AmatAx(1,4)+AmatAx(1,5))
     +         - (two/three)*T_tau(1,1)
         QmatAx(2,1) = third*(AmatAx(2,1)+AmatAx(2,4)+AmatAx(2,5))
     +         - (two/three)*T_tau(1,2)
         QmatAx(3,1) = third*(AmatAx(3,1)+AmatAx(3,4)+AmatAx(3,5))
     +         - (two/three)*T_tau(1,2)
         QmatAx(4,1) = third*(AmatAx(4,1)+AmatAx(4,4)+AmatAx(4,5))
     +         - (two/three)*T_tau(2,2)
         QmatAx(5,1) = third*(AmatAx(5,1)+AmatAx(5,4)+AmatAx(5,5))
     +         - (two/three)*T_tau(3,3)
         QmatAx(1,4) = QmatAx(1,1)
         QmatAx(2,4) = QmatAx(2,1)
         QmatAx(3,4) = QmatAx(3,1)
         QmatAx(4,4) = QmatAx(4,1)
         QmatAx(5,4) = QmatAx(5,1)
         QmatAx(1,5) = QmatAx(1,1)
         QmatAx(2,5) = QmatAx(2,1)
         QmatAx(3,5) = QmatAx(3,1)
         QmatAx(4,5) = QmatAx(4,1)
         QmatAx(5,5) = QmatAx(5,1)
         !
         if((nNode.eq.4).and.(nInt.eq.4)) then
            !
            ! This is the tangent using the F-bar method with the
            !  4 node fully integrated linear element
            !
            Kuu = Kuu
     +           + matmul(matmul(transpose(GmatAx),AmatAx),GmatAx)
```

```
     +               *detMapJC*w(intpt)*AR
     +               + matmul(transpose(GmatAx),matmul(QmatAx,
     +               (G0matAx-GmatAx)))*detMapJC*w(intpt)*AR
            else
               !
               ! This is the tangent NOT using the F-bar method with all
               !  other elements
               !
               Kuu = Kuu
     +               + matmul(matmul(transpose(GmatAx),AmatAx),GmatAx)
     +               *detMapJC*w(intpt)*AR
            endif
            !
         endif
      enddo
      !
      ! End the loop over integration points
      !----------------------------------------------------------------



      !----------------------------------------------------------------
      ! Return Abaqus the RHS vector and the Stiffness matrix.  This
      !  is essentially giving Abaqus the residual and the tangent matrix.
      !
      ! Return Abaqus the right hand side vector
      !
      do i=1,nNode
         A11 = (nDim+2)*(i-1)+1
         A12 = nDim*(i-1)+1
         !
         ! displacement
         !
         rhs(A12,1) = Ru(A12,1)
         rhs(A12+1,1) = Ru(A12+1,1)
         !
      enddo
      !
      ! Return Abaqus the tangent matrix
      !
      amatrx = Kuu
      !
      ! End return of RHS and AMATRX
      !----------------------------------------------------------------


      return
      end subroutine uel

!**********************************************************************
!     Material subroutine
!**********************************************************************

      subroutine NeoHookean(props,nprops,F_tau,T_tau,SpTanMod,stat)

      implicit none
      !
      integer i,j,k,l,m,n,nprops,stat
      !
```

```
      real*8 props(nprops),F_tau(3,3),T_tau(3,3),
     +  SpTanMod(3,3,3,3),Iden(3,3),Gshear,Kbulk,
     +  detF,Finv(3,3),FT(3,3),FinvT(3,3),C_tau(3,3),
     +  I1_tau,I1bar,TR_tau(3,3),dTRdF(3,3,3,3)
      !
      real*8 zero,one,two,three,fourth,third,half
      parameter(zero=0.d0,one=1.d0,two=2.d0,three=3.d0,fourth=1.d0/4.d0,
     +     third=1.d0/3.d0,half=1.d0/2.d0)


      ! Identity matrix
      !
      call onem(Iden)


      ! Obtain material properties
      !
      Gshear = props(1) ! Shear modulus
      Kbulk  = props(2) ! Bulk modulus

      ! Compute the determinant, the inverse, the transpose,
      !  and the inverse transpose of the deformation gradient
      !
      call matInv3D(F_tau,Finv,detF,stat)
      FT = transpose(F_tau)
      FinvT = transpose(Finv)


      ! Compute the right Cauchy-Green tensor
      !  and the first stretch invariant
      !
      C_tau = matmul(transpose(F_tau),F_tau)
      I1_tau = C_tau(1,1) + C_tau(2,2) + C_tau(3,3)


      ! Compute the trace of the distortional right Cauchy-Green tensor
      !
      I1bar = (detF**(-two/three))*I1_tau


      ! Compute the 1st Piola stress
      !
      TR_tau = (detF**(-two/three))*GShear*(F_tau-third*I1bar*FinvT)
     +     + Kbulk*detF*(detF  - one)*FinvT


      ! Compute the Cauchy stress
      !
      T_tau = (one/detF)*matmul(TR_tau,transpose(F_tau))


      ! Calculate the material tangent modulus
      !
      dTRdF = zero
      do i=1,3
         do j=1,3
            do k=1,3
               do l=1,3
                  dTRdF(i,j,k,l) = dTRdF(i,j,k,l)
```

```
     +                  + (detF**(-two/three))*GShear*
     +                  (
     +                  (-two/three)*F_tau(i,j)*Finv(l,k)
     +                  + (two/9.d0)*I1bar*Finv(j,i)*Finv(l,k)
     +                  + Iden(i,k)*Iden(j,l)
     +                  + third*I1bar*Finv(l,i)*Finv(j,k)
     +                  - (two/three)*Finv(j,i)*F_tau(k,l)
     +                  )
     +                  + detF*Kbulk*
     +                  (
     +                  (detF-one)*Finv(j,i)*Finv(l,k)
     +                  + detF*Finv(j,i)*Finv(l,k)
     +                  - (detF-one)*Finv(l,i)*Finv(j,k)
     +                  )
                     enddo
                  enddo
               enddo
            enddo
            !
            ! Calculate the spatial tangent modulus
            !
            SpTanMod = zero
            do i=1,3
               do j=1,3
                  do k=1,3
                     do l=1,3
                        do m=1,3
                           do n=1,3
                              SpTanMod(i,j,k,l) = SpTanMod(i,j,k,l) +
     +                           (dTRdF(i,m,k,n)*F_tau(j,m)*F_tau(l,n))/detF
                           enddo
                        enddo
                     enddo
                  enddo
               enddo
            enddo

            return
            end subroutine NeoHookean

!**********************************************************************
!     Element subroutines
!**********************************************************************

            subroutine xint2D1pt(xi,w,nIntPt)
            !
            ! This subroutine will get the integration point locations
            !  and corresponding gauss quadrature weights for 2D elements
            !  using 1 gauss point for integration
            !
            !  xi(nIntPt,2): xi,eta coordinates for the integration pts
            !  w(nIntPt):    corresponding integration weights
            !
            implicit none
            !
            integer nIntPt,nDim
            !
            real*8 xi(1,2), w(1)
```

```
      ! Initialize
      !
      w = 0.d0
      xi = 0.d0


      ! Number of Gauss points
      !
      nIntPt = 1


      ! Gauss weights
      !
      w = 4.d0


      ! Gauss pt location in master element
      !
      xi(1,1) = 0.d0
      xi(1,2) = 0.d0


      return
      end subroutine xint2D1pt

!**********************************************************************

      subroutine xint2D4pt(xi,w,nIntPt)
      !
      ! This subroutine will get the integration point locations
      !  and corresponding gauss quadrature weights for 2D elements
      !  using 4 gauss points for integration
      !
      !  xi(nIntPt,2): xi,eta coordinates for the integration pts
      !  w(nIntPt):    corresponding integration weights
      !
      implicit none
      !
      integer nIntPt,nDim
      !
      real*8 xi(4,2), w(4)


      ! Initialize
      !
      w = 0.d0
      xi = 0.d0


      ! Number of Gauss points
      !
      nIntPt = 4


      ! Gauss weights
      !
      w(1) = 1.d0
      w(2) = 1.d0
```

```
      w(3) = 1.d0
      w(4) = 1.d0


      ! Gauss pt locations in master element
      !
      xi(1,1) = -dsqrt(1.d0/3.d0)
      xi(1,2) = -dsqrt(1.d0/3.d0)
      xi(2,1) = dsqrt(1.d0/3.d0)
      xi(2,2) = -dsqrt(1.d0/3.d0)
      xi(3,1) = -dsqrt(1.d0/3.d0)
      xi(3,2) = dsqrt(1.d0/3.d0)
      xi(4,1) = dsqrt(1.d0/3.d0)
      xi(4,2) = dsqrt(1.d0/3.d0)


      return
      end subroutine xint2D4pt

!**********************************************************************

      subroutine calcShape2DLinear(nIntPt,xi_int,intpt,sh,dshxi)
      !
      ! Calculate the shape functions and their derivatives at the
      ! given integration point in the master element
      !
      ! sh(i) = shape function of node i at the intpt.
      ! dshxi(i,j) = derivative wrt j direction of shape fn of node i
      !
      implicit none
      !
      integer intpt,nDim,nIntPt
      !
      real*8 xi_int(nIntPt,2),sh(4),dshxi(4,2),xi,eta
      !
      real*8 zero,one,fourth
      parameter(zero=0.d0,one=1.d0,fourth=1.d0/4.d0)


      ! Location in the master element
      !
      xi = xi_int(intpt,1)
      eta = xi_int(intpt,2)


      ! The shape functions
      !
      sh(1) = fourth*(one - xi)*(one - eta)
      sh(2) = fourth*(one + xi)*(one - eta)
      sh(3) = fourth*(one + xi)*(one + eta)
      sh(4) = fourth*(one - xi)*(one + eta)


      ! The first derivatives
      !
      dshxi(1,1) = -fourth*(one - eta)
      dshxi(1,2) = -fourth*(one - xi)
      dshxi(2,1) = fourth*(one - eta)
      dshxi(2,2) = -fourth*(one + xi)
```

```
      dshxi(3,1) = fourth*(one + eta)
      dshxi(3,2) = fourth*(one + xi)
      dshxi(4,1) = -fourth*(one + eta)
      dshxi(4,2) = fourth*(one  - xi)


      return
      end subroutine calcShape2DLinear

!**********************************************************************

      subroutine mapShape2D(nNode,dshxi,coords,dsh,detMapJ,stat)
      !
      ! Map derivatives of shape fns from xi-eta-zeta domain
      !  to x-y-z domain.
      !
      implicit none
      !
      integer i,j,k,nNode,ieror,stat
      !
      real*8 dshxi(nNode,2),dsh(nNode,2),coords(3,nNode),mapJ(2,2),
     +  mapJ_inv(2,2),detmapJ
      !
      real*8 zero,one,two,half,fourth,eighth
      parameter(zero=0.d0,one=1.d0,two=2.d0,half=0.5d0,fourth=0.25d0,
     +     eighth=1.d0/8.d0)


      ! Calculate the mapping Jacobian matrix:
      !
      mapJ = zero
      do i=1,2
        do j=1,2
          do k=1,nNode
              mapJ(i,j) = mapJ(i,j) + dshxi(k,i)*coords(j,k)
          end do
        end do
      end do


      ! Calculate the inverse and the determinant of Jacobian
      !
      call matInv2D(mapJ,mapJ_inv,detMapJ,stat)


      ! Calculate first derivatives wrt x, y, z
      !
      dsh = transpose(matmul(mapJ_inv,transpose(dshxi)))


      return
      end subroutine mapShape2D

!**********************************************************************

      subroutine mapShape2Da(nNode,dshxi,coords,dsh,detMapJ,stat)
      !
      ! Map derivatives of shape fns from xi-eta-zeta domain
      !  to x-y-z domain.
```

```
      !
      ! This subroutine is exactly the same as the regular mapShape2D
      !  with the exception that coords(2,nNode) here and coords(3,nNode)
      !  in the regular.  I have noticed that a "heat transfer" and
      !  "static" step uses MCRD=2, but for "coupled-temperature-displacement"
      !  you will get MCRD=3, even for a plane analysis.
      !
      implicit none
      !
      integer i,j,k,nNode,ieror,stat
      !
      real*8 dshxi(nNode,2),dsh(nNode,2),coords(2,nNode),mapJ(2,2),
     +  mapJ_inv(2,2),detmapJ
      !
      real*8 zero,one,two,half,fourth,eighth
      parameter(zero=0.d0,one=1.d0,two=2.d0,half=0.5d0,fourth=0.25d0,
     +     eighth=1.d0/8.d0)


      ! Calculate the mapping Jacobian matrix:
      !
      mapJ = zero
      do i=1,2
        do j=1,2
          do k=1,nNode
              mapJ(i,j) = mapJ(i,j) + dshxi(k,i)*coords(j,k)
          end do
        end do
      end do


      ! Calculate the inverse and the determinant of Jacobian
      !
      call matInv2D(mapJ,mapJ_inv,detMapJ,stat)


      ! Calculate first derivatives wrt x, y, z
      !
      dsh = transpose(matmul(mapJ_inv,transpose(dshxi)))


      return
      end subroutine mapShape2Da

!**********************************************************************
!     Utility subroutines
!**********************************************************************

      subroutine matInv3D(A,A_inv,det_A,istat)
      !
      ! Returns A_inv, the inverse and det_A, the determinant
      ! Note that the det is of the original matrix, not the
      ! inverse
      !
      implicit none
      !
      integer istat
      !
      real*8 A(3,3),A_inv(3,3),det_A,det_A_inv
```

```
      istat = 1

      det_A = A(1,1)*(A(2,2)*A(3,3) - A(3,2)*A(2,3)) -
     +        A(2,1)*(A(1,2)*A(3,3) - A(3,2)*A(1,3)) +
     +        A(3,1)*(A(1,2)*A(2,3) - A(2,2)*A(1,3))

      if (det_A .le. 0.d0) then
        write(*,*) 'WARNING: subroutine matInv3D:'
        write(*,*) 'WARNING: det of mat=',det_A
        istat = 0
        return
      end if

      det_A_inv = 1.d0/det_A

      A_inv(1,1) = det_A_inv*(A(2,2)*A(3,3)-A(3,2)*A(2,3))
      A_inv(1,2) = det_A_inv*(A(3,2)*A(1,3)-A(1,2)*A(3,3))
      A_inv(1,3) = det_A_inv*(A(1,2)*A(2,3)-A(2,2)*A(1,3))
      A_inv(2,1) = det_A_inv*(A(3,1)*A(2,3)-A(2,1)*A(3,3))
      A_inv(2,2) = det_A_inv*(A(1,1)*A(3,3)-A(3,1)*A(1,3))
      A_inv(2,3) = det_A_inv*(A(2,1)*A(1,3)-A(1,1)*A(2,3))
      A_inv(3,1) = det_A_inv*(A(2,1)*A(3,2)-A(3,1)*A(2,2))
      A_inv(3,2) = det_A_inv*(A(3,1)*A(1,2)-A(1,1)*A(3,2))
      A_inv(3,3) = det_A_inv*(A(1,1)*A(2,2)-A(2,1)*A(1,2))


      return
      end subroutine matInv3D

!*****************************************************************************

      subroutine matInv2D(A,A_inv,det_A,istat)
      !
      ! Returns A_inv, the inverse, and det_A, the determinant
      ! Note that the det is of the original matrix, not the
      ! inverse
      !
      implicit none
      !
      integer istat
      !
      real*8 A(2,2),A_inv(2,2),det_A,det_A_inv


      istat = 1

      det_A = A(1,1)*A(2,2) - A(1,2)*A(2,1)

      if (det_A .le. 0.d0) then
        write(*,*) 'WARNING: subroutine matInv2D:'
        write(*,*) 'WARNING: det of mat=',det_A
        istat = 0
        return
      end if

      det_A_inv = 1.d0/det_A
```

```
      A_inv(1,1) =  det_A_inv*A(2,2)
      A_inv(1,2) = -det_A_inv*A(1,2)
      A_inv(2,1) = -det_A_inv*A(2,1)
      A_inv(2,2) =  det_A_inv*A(1,1)


      return
      end subroutine matInv2D

!***********************************************************************

      subroutine mdet(A,det)
      !
      ! This subroutine calculates the determinant
      ! of a 3 by 3 matrix [A]
      !
      implicit none
      !
      real*8  A(3,3),det


      det = A(1,1)*A(2,2)*A(3,3)
     +      + A(1,2)*A(2,3)*A(3,1)
     +      + A(1,3)*A(2,1)*A(3,2)
     +      - A(3,1)*A(2,2)*A(1,3)
     +      - A(3,2)*A(2,3)*A(1,1)
     +      - A(3,3)*A(2,1)*A(1,2)


      return
      end subroutine mdet

!***********************************************************************

      subroutine onem(A)
      !
      ! This subroutine stores the identity matrix in the
      ! 3 by 3 matrix [A]
      !
      implicit none
      !
      integer i,j
      !
      real*8 A(3,3)


      do i=1,3
         do J=1,3
          if (i .eq. j) then
              A(i,j) = 1.0
            else
              A(i,j) = 0.0
            end if
         end do
      end do


      return
      end subroutine onem
```

D EXEMPLOS DE ARQUIVOS *INPUT FILES*

```
*Heading
** Job name: CP4E_Comp100% Model name: Model-1
** Generated by: Abaqus/CAE 6.14-1
*Preprint, echo=NO, model=NO, history=NO, contact=NO
**
** PARTS
**
*Part, name=Part-1
*Node
      1,           0.,           0.
      2,          15.,           0.
      3,           0.,          15.
      4,          15.,          15.
*Element, type=CPE4
1, 1, 2, 4, 3
*Nset, nset=Set-1, generate
 1,  4,  1
*Elset, elset=Set-1
 1,
** Section: Section-1
*Solid Section, elset=Set-1, material=Material-1
,
*End Part
**
**
** ASSEMBLY
**
*Assembly, name=Assembly
**
*Instance, name=Part-1-1, part=Part-1
*End Instance
**
*Nset, nset=Set-1, instance=Part-1-1
 1,
*Nset, nset=Set-2, instance=Part-1-1
 2,
*Nset, nset=Set-3, instance=Part-1-1
 3,
*Nset, nset=Set-4, instance=Part-1-1
 1,
*Nset, nset=Set-5, instance=Part-1-1
 2,
*Nset, nset=Set-6, instance=Part-1-1
 3,
*Nset, nset=Set-7, instance=Part-1-1
 3, 4
*Elset, elset=Set-7, instance=Part-1-1
 1,
*End Assembly
**
** MATERIALS
**
*Material, name=Material-1
*Hyperelastic, neo hooke
 0.0005,200.
**
** BOUNDARY CONDITIONS
**
** Name: BC-1 Type: Symmetry/Antisymmetry/Encastre
*Boundary
Set-4, ZASYMM
** Name: BC-2 Type: Symmetry/Antisymmetry/Encastre
*Boundary
Set-5, YSYMM
** Name: BC-3 Type: Symmetry/Antisymmetry/Encastre
*Boundary
Set-6, XSYMM
** ----------------------------------------------------------------
**
** STEP: Compression
**
*Step, name=Compression, nlgeom=YES, inc=1000
*Static
0.01, 1., 1e-08, 0.01
```

```
**
** BOUNDARY CONDITIONS
**
** Name: BC-4 Type: Displacement/Rotation
*Boundary
Set-7, 2, 2, -15.
**
** OUTPUT REQUESTS
**
*Restart, write, frequency=0
**
** FIELD OUTPUT: F-Output-1
**
*Output, field, variable=PRESELECT
**
** HISTORY OUTPUT: H-Output-1
**
*Output, history, variable=PRESELECT
*End Step
```

```
*Heading
Compression of a rubber plate
**********************************************************************
**          PARAMETERS
**********************************************************************
*Parameter
**Shear modulus (MPa)
Gshear = 0.001
**Bulk modulus (MPa)
Kbulk = 1.0
**Plane strain
pe = 1
** ----------------------------------------------------------------
**********************************************************************
**          MODEL DEFINITION
**********************************************************************
*Node, nset=nall
      1,           0.,          15.
      2,           0.,           0.
      3,          15.,          15.
      4,          15.,           0.
** ----------------------------------------------------------------
**********************************************************************
**          ELEMENT DEFINITION
**********************************************************************
*User Element,Nodes=4,Type=U1,Iproperties=1,Properties=2,Coordinates=2,Variables=1,Unsymm
1,2
*Element, Type=U1, Elset=FBAR
1, 1, 2, 4, 3
**
** MAKE THE DUMMY MESH,
** shares nodes with the real mesh,
** allows visualization of the results, access to
** hourglass controls, and access to mechanical,
** traction-type boundary conditions
**
*Element, type=CPE4, Elset=ElDummy
10001, 1, 2, 4, 3
**********************************************************************
**          MATERIAL DEFINITION
**********************************************************************
*UEL Property, Elset=FBAR
<Gshear>, <Kbulk>, <pe>
**
**********************************************************************
**
** DUMMY MESH material properties
**
*Solid section, elset=ElDummy, material=Dummy
*Material, name=Dummy
** The dummy mesh stiffness is very small
*Elastic
1e-20
**
**********************************************************************
**          INITIAL CONDITIONS
**********************************************************************
*Boundary
1, 1, 1
2, 1, 2
4, 2, 2
** ----------------------------------------------------------------
**********************************************************************
**          STEP DEFINITION
**********************************************************************
*Step, Name=Compression, nlgeom=yes, unsymm=yes, inc=2000
*Static
0.01, 1., 1e-8, 0.01
*CONTROLS, PARAMETERS=LINE SEARCH
10,1.0,0.0001,0.25,0.10
*CONTROLS, PARAMETERS=TIME INCREMENTATION
,,,,,,,10,,,,,,
*Boundary
1, 2, 2, -15
```

```
3, 2, 2, -15
** OUTPUT REQUESTS
*Output, field, time interval=0.01
*node output, nset=nall
u
*Element output, elset=ElDummy
le
**
*End Step
```