UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS
DEPARTAMENTO DE ENGENHARIA DE ESTRUTURAS

Processamento Paralelo Aplicado Em Análise Não Linear De Cascas.

Elias Calixto Carrijo

Tese apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo como parte dos requisitos para obtenção do título de Doutor em Engenharia de Estruturas.

Orientador: Prof^{o}. Dr. João Batista de Paiva

São Carlos
2001

SUMÁRIO

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS

LISTA DE SÍMBOLOS

RESUMO

ABSTRACT

1 **INTRODUÇÃO**..... 1

2 **INTRODUÇÃO À COMPUTAÇÃO PARALELA.....** 6
2.1 Introdução 6
2.2 A Máquina de von Neumman 13
2.3 Classificação dos Sistemas de Computação 15
2.4 Processadores Vetoriais..... 16
2.5 Multicomputadores 18
2.6 Granularidade e Sistemas Escalonáveis 22
2.7 Desenvolvimento e Análise de Softwares Paralelos 22
2.7.1 Troca de Mensagens 23
2.7.2 Sincronização de Processos 25
2.7.3 Autoparalelizadores 27

2.7.4 Análise de Softwares Paralelos 27
2.7.5 Benchmark 28
2.8 Algoritmos Numéricos e Estratégias Computacionais 29
2.8.1 Decomposição de Domínio 31
2.8.2 Geradores de Malha 33

3 **ANÁLISE NÃO LINEAR DE CASCAS** 34
3.1 Introdução 34
3.2 O Elemento Finito de Casca Plano 35
3.3 O Elemento Finito de Chapa 40
3.4 O Elemento Finito de Placa 45
3.5 O Modelo Elastoplástico 48
3.5.1 Procedimento Numérico 55

4 **ANÁLISE NÃO LINEAR EM PARALELO** 64
4.1 Introdução 64
4.2 Paralelismo no Método Dos Elementos Finitos 65
4.3 Cálculo da Matriz de Rigidez 66
4.3.1 Cálculo da Matriz de Rigidez – Procedimento Usual 66
4.3.2 O Cálculo Paralelo da Matriz de Rigidez 68
4.4 A Resolução do Sistema de Equações 71
4.4.1 O Método Dos Gradientes Conjugados 72
4.4.2 A Implementação em Paralelo do Método Dos Gradientes Conjugados..... 74
4.5 Análise Não Linear Em Paralelo 80

5 **EXEMPLOS** 93
5.1 Introdução 93
5.2 Placa Apoiada Estratificada 93
5.3 Casca Cilíndrica 97

6 **CONCLUSÕES 100**

7 **BIBLIOGRAFIA 103**

LISTA DE FIGURAS

Figura 2.1 Desenvolvimento dos processadores 8

Figura 2.2. Modelo de von Neumann. 13

Figura 2.3. Etapas da máquina de von Neumann..... 14

Figura 2.4. Execução seqüencial de multiplicação de vetor por escalar 17

Figura 2.5. Multiplicação de um vetor por um escalar em *pipeline* 18

Figura 2.6. Esquema de máquina SIMD. 18

Figura 2.7. Multicomputadores 19

Figura 2.8. Sistema multiprocessador com barramento 20

Figura 2.9. Sistema multiprocessador com barramento cruzado. 21

Figura 2.10. Sistema multiprocessador com rede matricial 21

Figura 2.11. Multiprocessador com interconexão hipercubo. 22

Figura 2.12 Comunicações coletivas 25

Figura 2.13. Sincronização estabelecida por Barreira 26

Figura 3.1 Elemento finito de chapa 45

Figura 3.2. Elemento finito de placa – DKT. 46

Figura 3.3. Representação do comportamento elastoplástico. 49

Figura 3.4 Superfície de plastificação 50

Figura 3.5 Representação geométrica do critério de von Mises. 52

Figura 3.6 Representação gráfica do procedimento incremental-iterativo ... 56

Figura 4.1 Etapas do Método dos Elementos Finitos..... 65

Figura 4.2. Estrutura discretizada com 4 elementos finitos. 68

Figura 4.3 Cálculo da matriz de rigidez com abordagem nodal 70

Figura 4.4 Cálculo da matriz de rigidez com dois processos 71

Figura 4.5 Cálculo do produto $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_{0.}$ 75

Figura 4.6 Armazenamento do produto matriz-vetor 75

Figura 4.7 Acumulação da adição entre dois vetores 76

Figura 4.8 Armazenamento do resíduo em vetor auxiliar 76

Figura 4.9 Procedimento para cálculo do produto escalar 77

Figura 4.10 Atualização do vetor **x**. 78

Figura 4.11 Análise não-linear em modo sequencial 81

Figura 4.12 Análise não-linear em paralelo..... 82

Figura 4.13 Armazenamento da matriz de rigidez. 86

Figura 4.14 Matriz com condição de contorno. 87

Figura 4.12 Malha para cálculo do vetor de forças nodais 89

Figura 5.1 Placa quadrada apoiada nos quatro lados 94

Figura 5.2 Curva carga deslocamento do ponto central da placa 95

Figura 5.3 Speed-up para placa apoiada 96

Figura 5.4 Eficiência para placa apoiada 96

Figura 5.5 Casca cilíndrica apoiada em paredes rígidas 97

Figura 5.6 Deslocamento transversal do ponto intermediário da borda livre 98

Figura 5.7 Speed-up para casca cilíndrica 99

Figura 5.8 Eficiência para casca cilíndrica 99

LISTA DE TABELAS

Tabela 5.1 Tempo total de processamento para placa apoiada 95

Tabela 5.2 Tempo total de processamento para casca cilíndrica 98

LISTA DE ALGORITMOS

Algoritmo 4.1 Cálculo da matriz de rigidez em modo sequencial 67

Algoritmo 4.2 Cálculo da matriz de rigidez 69

Algoritmo 4.3 Método dos gradientes conjugados 73

Algoritmo 4.4 Divisão de intervalos nodais 83

Algoritmo 4.5 Cálculo do vetor de forças externas 84

Algoritmo 4.6 Imposição de condições de contorno 87

Algoritmo 4.7 Cálculo do resíduo 91

SÍMBOLOS E ABREVIATURAS

SISD	Single Instruction Single Data
SIMD	Sigle Instruction Multile Data)
MISD	Multiple Instruction Multiple Data
MIMD	Multiple Instruction Multiple Data
PIM	Parallel Iterative Methods
GC	Gradientes Conjugados
QGC	Gradientes Conjugados Quadrados
Bi-GC	Bi Gradientes Conjugados
Bi-GCSTAB	Bi-Gradientes Conjugados Estabilizados
DKT	Discrete Kirchhof Theory
FF	Free Formulation
Flops	Flow Point Operation Per Second
CISC	Centro de Informática de São Carlos
MPI	Message Passing Interface
PVM	Parallel Virtual Machine
SP	Scalable PowerParallel

LISTA DE SÍMBOLOS

$u(x,y,z)$	Deslocamento na direção x .
$v(x,y,z)$	Deslocamento na direção y.
$w(x,y,z)$	Deslocamentos na direção z.
$u'(x,y)$	Deslocamento na direção x em relação aos eixos locais.
$v'(x,y)$	Deslocamento na direção y em relação aos eixos locais.
$w'(x,y)$	Deslocamento na direção w em relação aos eixos locais.
ε_x	Vetor de deformações na direção x.
ε_y	Vetor de deformações na direção y
γ	Vetor de deformações angular
σ_x	Tensão normal na direção x.
σ_y	Tensão normal na direção y
τ_{xy}	Tensão cisalhante no plano xy.
$\boldsymbol{\varepsilon}$	Vetor de deformações totais
$\boldsymbol{\varepsilon}_c$	Vetor de deformações totais devido efeito de chapa.
ε_p	Vetor de deformações totais devido efeito de placa.
$\mathbf{U}_c$	Deslocamentos no domínio do elemento.
$\boldsymbol{\Phi}_c$	Matriz de funções de forma.

$\mathbf{V}$	Deslocamentos nodais do elemento.
$\mathbf{L}_c$	Matriz de derivadas.
$\mathbf{B}$	Matriz de derivadas das funções de forma.
$\mathbf{K}$	Matriz de rigidez
$\mathbf{K}_{cc}$	Matriz de rigidez do elemento de chapa
$\mathbf{K}_{pp}$	Matriz de rigidez do elemento de placa.
$\mathbf{K}_{cp}$	Matriz de rigidez de acoplamento de placa e chapa.
$\mathbf{K}_g$	Matriz de rigidez do elemento em coordenadas globais.
$\mathbf{R}$	Matriz de rotação
$\mathbf{N}$	Vetor de esforços normais.
$\mathbf{M}$	Vetor de momentos fletores.
Ψ	Vetor resíduo
$\mathbf{F}_{int,c}$	Vetor de forças internas devido efeito de chapa.
$\mathbf{F}_{int,p}$	Vetor de forças internas devido efeito de placa.
$\mathbf{U}$	Vetor de deslocamentos no domínio do elemento.
$\mathbf{N}_{rc}$	Expansão polinomial para modos de corpo rígido e deformação constante.
$\mathbf{N}_s$	Expansão polinomial para modos superiores.
α_{rc}	Vetor de parâmetros generalizados para modos de corpo rígido e deformação constante.
α_s	Vetor de par6ametros generalizados para modos superiores.
$\mathbf{V}_c$	Vetor de deslocamentos nodais.
$\mathbf{G}_{rc}$	Matriz de coeficientes generalizados
$\mathbf{t}_{rc}$	Forças nodais no elemento.

$\boldsymbol{\sigma}_{rc}$	Tensões distribuídas nas faces do elemento.
$\mathbf{F}_{rc}$	Vetor de forças nos pontos nodais do elemento.
U	Energia potencial
$\mathbf{K}_{rc}$	Matriz de rigidez dos modos de corpo rígido e deformação constante.
$\mathbf{K}_{s}$	Matriz de rigidez dos modos superiores.
$\mathbf{K}_{rcs}$	Matriz de rigidez de acoplamento
$\mathbf{L}$	Matriz amotoadora
$\mathbf{K}_{b}$	Matriz dos modos básicos.
$\mathbf{K}_{s}$	Matriz dos modos superiores.
θ_z	Rotação nos vértices do elemento
$\mathbf{u}_{b}$	Campo de deslocamentos dos modos básicos.
$\mathbf{u}_{s}$	Campo de deslocamento dos modos superiores
ξ, η	Coordenadas adimensionais
β_x	Rotações do plano médio do elemento em relação ao eixo x
β_y	Rotações do plano médio do elemento em relação ao eixo y.
$\mathbf{H}_x$ $\mathbf{H}_y$	Matrizes de funções de forma
$\mathbf{D}_{b}$	Matriz de coeficientes elásticos.
$\boldsymbol{\varepsilon}$	Deformações totais
$\boldsymbol{\varepsilon}_e$	Deformações elásticas
$\boldsymbol{\varepsilon}_p$	Deformações plásticas.
F	Função de plastificação
$f(\sigma)$	Função de estado atual
$Y(\kappa)$	Tensão de escoamento
$\mathbf{D}_{ep}$	Matriz de coeficientes elastoplásticos.

$\mathbf{F}_{ext}$	Vetor de forças externas.
$\mathbf{K}_T$	Matriz de rigidez elastoplástica tangente
σ_y	Limite elástico.
σ^0_y	Limite elástico inicial
$\mathbf{D}_{ep,p}$	Matriz de coeficientes elastoplásticos para elemento de placa
$\mathbf{D}_{ep,c}$	Matriz de coeficientes elastoplásticos para elemento de chapa
$\mathbf{D}_{ep,cP}$	Matriz de coeficientes elastoplásticos acoplamento.
$\mathbf{K}_{T,p}$	Matriz de rigidez tangente de placa.
$\mathbf{K}_{T,c}$	Matriz de rigidez tangente de chapa.
$\mathbf{K}_{T,c}$	Matriz de rigidez tangente de acoplamento.
$\mathbf{A}$	Matriz de coeficientes do sistema linear
$\mathbf{x}$	Vetor de incógnitas
$\mathbf{b}$	Vetor de parâmetros independentes
$\mathbf{x}_0$	Solução do sistema linear
$\mathbf{x}$	Solução do sistema linear na iteração
$\mathbf{y}$	Erro
$\mathbf{r}$	Resíduo
$\mathbf{x}_{k+1}$	Solução na iteração k+1
$\mathbf{x}_k$	Solução na iteração k
$\mathbf{y}_{k+1}$	Erro na iteração k+1
$\mathbf{y}_k$	Erro na iteração k

RESUMO

CARRIJO, E. C. (2001) Processamento paralelo aplicado em análise não linear de estruturas. São Carlos, 2001, 108p Tese (Doutorado). Escola de Engenharia de São Carlos, Universidade de São Paulo.

Este trabalho tem o intuito de fazer uso do processamento paralelo na análise não linear de cascas pelo Método dos Elementos Finitos. O elemento finito de casca é obtido com o acoplamento de um elemento de placa e um de chapa. O elemento de placa utiliza formulação de Kirchhof (DKT) para placas delgadas e o elemento de chapa faz uso da formulação livre (FF), introduzindo um grau de liberdade rotacional nos vértices. A análise não-linear com plasticidade utiliza o modelo de plasticidade associada com algoritmo de integração explícito, modelo de escoamento de von Mises com integração em camadas (modelo estratificado), para materiais isotrópicos.

A implementação em paralelo é realizada em um sistema com memória distribuída e biblioteca de troca de mensagens PVM (Parallel Virtual Machine). O procedimento não-linear é completamente paralelizado, excetuando a impressão final de resultados. As etapas que constituem o Método dos Elementos Finitos, matriz de rigidez da estrutura e resolução do sistema de equações lineares são paralelizadas. Para o cálculo da matriz de rigidez utiliza-se um algoritmo com decomposição de domínio explícito. Para resolução do sistema de equações lineares utiliza-se o método dos Gradientes Conjugados com implementação em paralelo. É apresentada uma breve revisão bibliográfica sobre o paralelismo, com comentários sobre perspectivas em análise estrutural.

Palavras-chave: Casca, Plasticidade, Elementos Finitos, Paralelismo.

ABSTRACT

CARRIJO, E. C. (2001) Parallel processing Applied to Nonlinear Structural Analysis. São Carlos, 2001. Thesis (Doctorate). São Carlos Engineering School, São Paulo University.

This work aims at using parallel processing for nonlinear analysis of shells through Finite Element Method. The shell finite element is obtained by coupling a plate element with a membrane one. The plate element uses Kirchhoff's formulation (DKT) for thin plates and the membrane element makes use of the free formulation (FF), introducing a rotational degree of freedom at the vertexes. Nonlinear plastic analysis uses associated plasticity model with explicit integration algorithm, von Mises yelding model with layer integration (stratified model) for isotropic materials.

Parallel implementation is done on a distributed memory system and message exchange library PVM (Parallel Virtual Machine). Nonlinear procedure is completely parallelised, but final printing of results. The Finite Element Method steps, structural stifness matrix and solution of the linear equation system are parallelised. An explicit domain decomposition algorithm is used for the stifness matrix evaluation. To solve the linear equation system, one uses conjugated gradients method, with parallel implementation. A brief bibliography about parallelism is presented, with comments on structural analysis perspectives.

Keywords: Shell, Plasticity, Finite Elements, Parallelism.

CAPÍTULO I

INTRODUÇÃO

A análise estrutural foi uma das áreas que mais se desenvolveram nos últimos anos, amparada principalmente pelo surgimento concomitante dos computadores e dos métodos numéricos de análise matemática. Problemas estruturais com solução analítica difícil passaram a dispor de métodos numéricos eficientes na obtenção de resultados. O surgimento desses métodos, acompanhado de máquinas que viabilizassem sua aplicação, não só possibilitou a solução dos problemas existentes, como também abriu uma perspectiva de investigação científica sem limites não só na análise estrutural, mas em todos os campos das ciências da natureza.

Munida de ferramentas tais como Métodos dos Elementos Finitos (MEF) e Método dos Elementos de Contorno (MEC), a análise de estruturas pôde se estender a fenômenos e modelos antes impensáveis, tais como não-linearidade, fadiga, fratura, dano, etc. Hoje o uso desses métodos numéricos, e das máquinas que os amparam, passou do campo científico para a prática do cálculo estrutural.

Entretanto, mesmo com os recursos hoje disponíveis (métodos numéricos - MEF, MEC- e modernos computadores pessoais), o uso dos modelos mecânicos (não-linearidades física e geométrica, plasticidade, fratura, fadiga, dano) no

dimensionamento de estruturas é ainda pouco utilizado, pois se trata de procedimentos iterativos com elevado custo computacional.

Pode-se esperar que o desenvolvimento dos computadores alcance um patamar tal que essa transferência se concretize naturalmente (isso em nível de processamento - atuais microprocessadores). Entretanto, esses computadores, dito seqüenciais (executam suas tarefam seqüencialmente, uma após a outra), tem um limite físico de aperfeiçoamento, que é a velocidade da luz no vácuo, o qual não está muito distante de ser alcançado, comprometendo assim o uso desses modelos como critério de dimensionamento.

Com o intuito de contornar essa limitação física, comum aos computadores pessoais e aos supercomputadores, surgiu uma nova categoria de computadores com arquitetura diferente das máquinas seqüenciais, os computadores com arquitetura paralela. A principal característica dessa arquitetura é possuir vários processadores trabalhando concomitantemente na execução de uma mesma tarefa. O conceito de paralelismo apresentou grande evolução em relação às máquinas seriais, rompendo com a necessidade de se executar tarefas seqüencialmente.

Essa nova modalidade de máquinas aparece como uma das mais promissoras no campo do desenvolvimento dos computadores, pois permite que se alcance o desempenho dos atuais supercomputadores com um custo bem inferior. O desenvolvimento e uso dessas máquinas têm encontrado espaço crescente no ambiente acadêmico, inclusive no âmbito da pesquisa da engenharia estrutural. Esse interesse pelo paralelismo é motivado pela perspectiva que o mesmo apresenta em relação ao desenvolvimento dos computadores e pode ser avaliado pelo volume de trabalhos explorando o tema. Portanto, é importante nesse momento familiarizar-se com essa nova arquitetura de computadores e com o modo de conceber programas voltados aos mesmos a fim de tirar o máximo proveito de seus recursos.

Diferente das máquinas seqüenciais, quando se propõe elaborar um programa para ser usado em um computador com arquitetura paralela deve-se conhecer não só as linguagens de alto nível que serão usadas, mas também a configuração da máquina disponível, pois nesse caso o programa desenvolvido será voltado especificamente para a máquina que se quer empregar.

Nesse caso pode-se dispor de processadores vetoriais, que são máquinas especializadas no tratamento de operações com vetores e matrizes, e os multicomputadores, que são arquiteturas compostas de um conjunto de processadores conectados entre si por uma rede de interconexão (com ou sem módulo de memória próprio). O modo segundo o qual se processa a troca de informações (rede de interconexão) pode ter diferentes arranjos tais como barramento, barramento cruzado, matricial, hipercubo e multiestágio. Com relação à organização da memória os computadores paralelos podem possuir uma memória global e vários processadores com acesso à mesma (memória compartilhada ou global), ou cada processador pode possuir sua própria memória (memória distribuída ou local). Mais detalhes sobre estas características serão apresentados no próximo ítem.

No que se refere à engenharia estrutural, mais especificamente à análise numérica, fazer uso dos recursos que essas máquinas oferecem significa ter que preparar os métodos numéricos atualmente empregados (MEF, MEC) para que sejam usados com máxima eficiência no paralelismo.

Fazendo uso do Método dos Elementos Finitos, pode-se extrair paralelismos das etapas que constituem o emprego deste método. Nesse caso, pode-se investir no processo de montagem da matriz de rigidez global e na resolução do sistema de equações lineares da estrutura. A resolução do sistema de equações lineares é a etapa que requer maior esforço computacional (caso elástico-linear), sendo objeto de pesquisa dos matemáticos. No caso da análise não-linear, o método de Newton para resolução de sistemas de equações não-lineares tem particular interesse dado a sua intensa utilização. Possíveis alternativas de se paralelizar as etapas que compõem método de Newton serão posteriormente comentadas. Para a análise não-linear de estruturas pode-se também concentrar esforços no cálculo do resíduo.

Na etapa de cálculo da matriz de rigidez, caso seja intenção do pesquisador utilizar o algoritmo usual, que consiste em alocar os coeficientes de rigidez dos elementos para suas posições na matriz da estrutura a partir de um laço sobre os elementos, verifica-se que este algoritmo mostra-se inadequado, o que será demonstrado a posteriori. Sendo uma das etapas que envolvem a utilização do MEF, deve-se envidar esforços no sentido de se buscar o paralelismo dessa fase. Esse

assunto tem sido motivo de vários trabalhos, que serão comentados oportunamente na revisão bibliográfica.

Também como etapa do MEF, algoritmos para resolução de sistemas de equações lineares têm merecido atenção na pesquisa sobre o paralelismo, sendo a fase de maior demanda computacional. Tradicionalmente usado em programas sequenciais, o método de Gauss não oferece os mesmos resultados no paralelismo, uma vez que apresenta dependência de dados nas etapas de triangularização e retrosubstituição. Além do exposto, o método de Gauss apresenta um aspecto negativo quando se trabalha com problemas envolvendo muitos graus de liberdade, que é o erro provocado pelo arredondamento. Isso tem motivado o uso dos métodos iterativos, com destaque para Gauss-Jacobi, Gauss-Seidel, Gradientes Conjugados e Gradientes Conjugados com Pré Condicionadores.

Apesar de exigir uma nova mentalidade na concepção de algoritmos que serão implementados nos sistemas computacionais, diferentemente do modo sequencial de programação, o paralelismo tem conquistado espaço extenso e em ritmo crescente junto à pesquisa em análise numérica. Esse fato pode ser constatado pelo volume de trabalhos publicados que utilizam esses conceitos e recursos, que se estendem desde a elaboração de algoritmos paralelos para pré-processadores até a resolução de sistemas de equações lineares para os diversos modelos de análise estrutural (linearidade, não-linearidade, dinâmica, etc). Esse interesse despertado nos pesquisadores da área de análise estrutural pode ser avaliado na revisão bibliográfica que se apresenta no capítulo II.

No âmbito deste departamento, o trabalho de REZENDE (1995) foi o pioneiro na utilização do paralelismo na análise estrutural. Neste trabalho faz-se uma análise não-linear de estruturas em barras, apresentando um algoritmo alternativo para a montagem da matriz de rigidez da estrutura, além de estudar a eficiência de algoritmos para a resolução de sistemas de equações lineares. Para a resolução de sistema de equações lineares usa-se o método direto de Gauss com modificações para que se possa extrair o paralelismo do mesmo. Também em ALMEIDA (1999) encontra-se uma implementação, em máquinas com memória distribuída, de um software para resolução de pavimentos pelo MEF. Neste caso, utilizou-se o conceito

de abordagem nodal no cálculo da matriz de rigidez e o método dos Gradientes Conjugados na resolução do sistema de equações.

O presente trabalho tem o intuito de prosseguir na pesquisa sobre paralelismo, voltando a atenção para estruturas bidimensionais, chapas, placas e cascas, usando modelos conhecidos de não-linearidade física.

No capítulo II estão apresentados conceitos básicos de computação paralela bem como a revisão bibliográfica realizada. No capítulo III apresenta-se toda a formulação dos elementos finitos utilizados na composição do elemento de casca plano, além do modelo de não linearidade empregado neste trabalho. No capítulo IV mostra-se a implementação em paralelo do programa para análise não-linear de cascas com comentários detalhados sobre o procedimento adotado. Por fim, apresenta-se no capítulo V os resultados do processamento em paralelo dos exemplos propostos. Mostra-se também o comportamento elastoplástico dos respectivos exemplos.

O equipamento a ser usado é um IBM SP2, com três nós de processamento, instalado no Centro de Informática de São Carlos. Apesar de possuir apenas três nós, esse equipamento apresenta elevado desempenho além de permitir que se estude e desenvolva os conceitos de computação distribuída.

CAPÍTULO II

INTRODUÇÃO À COMPUTAÇÃO PARALELA

2.1 INTRODUÇÃO

Desde sua invenção, identifica-se cinco gerações de computadores correspondendo à tecnologia utilizada na fabricação dos mesmos: Tubos de vácuo (1939-1950), diodos e transistores (1950-1960), circuitos integrados em pequena e média escala – SSI e MSI (1960-1970), circuitos integrados em alta escala -VLSI (1970-1990) e ultra escala em circuitos integrados - ULSI (1990-presente data). No desenvolvimento dessa tecnologia, maior ênfase foi dada à redução de custo, dimensão, consumo, velocidade e confiabilidade dos seus componentes. A microeletrônica apresentou um grande avanço nos anos recentes, principalmente pela utilização dos microprocessadores com material semicondutor. Essa tecnologia permitiu o avanço contínuo em ganho de velocidade de processamento e capacidade de memória. Atualmente a pesquisa se direciona a melhorar a densidade dos circuitos integrados, buscando um nível de ciclo de máquina próximo de 0.5 ns ($0.5x10^{-9}$ seg).

A memória também apresentou um grande desenvolvimento tanto em termos de capacidade de armazenamento quanto em velocidade de resposta. A avaliação de desempenho da memória é medida em termos de latência (tempo necessário para que a memória retorne um pedido) e taxa com que ela aceita pedidos e retorna resultados

(*bandwidth*). A divisão hierárquica de níveis de memória (desde registradores do microprocessador, memória cache, memória principal e discos magnéticos e ópticos) é um dos fatores que proporcionaram esse desenvolvimento. Futuramente os microprocessadores e a memória principal serão incorporados em um único dispositivo, o que proporcionará um aumento maior na desempenho das máquinas. Apesar do avanço obtido, a velocidade de execução de tarefas apresentada pelos microprocessadores é superior à das memórias, como no caso do IBM 3090 onde o ciclo de máquina é de 18 ns e o tempo de acesso à memória está entre 200 ns e 500 ns, tornando-se um entrave ao ganho final em velocidade das máquinas (fenômeno conhecido como “gargalo de von Neumman”). Esse fato impulsiona a pesquisa por memórias com maior velocidade de resposta e capacidade de armazenamento.

Os dispositivos de armazenamento de dados (memória permanente) também experimentaram grande desenvolvimento. Com o aumento da velocidade de processamento dos computadores fez-se necessário que os demais componentes também atingissem o mesmo patamar de desenvolvimento, proporcionando melhor aproveitamento dos sistemas de computação. Estes dispositivos cresceram tanto em capacidade de armazenamento quanto em velocidade de busca de dados. Além dos dispositivos magnéticos, surgiram também os dispositivos ópticos. Atualmente encontram-se disponíveis elementos com capacidade de armazenamento de 10^{15} bytes (pentabytes).

Novos equipamentos surgiram em virtude do desenvolvimento da microeletrônica (sistemas de alto desempenho, computadores pessoais, computadores portáteis). Segundo NOOR (1997) a próxima geração de computadores será influenciada basicamente pelo desenvolvimento de três áreas: capacidade dos componentes eletrônicos, inteligência artificial e arquitetura dos computadores. Além desse desenvolvimento, o aumento final da velocidade de processamento decorre da execução simultânea de atividades no computador. Assim impõe-se a necessidade aos profissionais que necessitam desses equipamentos em suas atividades de familiarizar-se com esses novos recursos.

A maioria dos novos equipamentos consegue atingir alto desempenho através de procedimentos concorrentes no computador. A exploração desse procedimento concorrente é usualmente conhecida como processamento paralelo. Um processo

paralelo executado em sistemas fisicamente dispersos é usualmente conhecido como processos distribuídos. A concorrência é usada não para aumentar a velocidade individual de um determinado trabalho, mas de todo o processamento. Essa nova arquitetura proporcionou um significativo avanço dos sistemas computacionais em termos de desempenho, como pode ser visto na Figura 2.1.

Sistemas computacionais com elevado poder de processamento são de fundamental importância na análise numérica, permitindo que modelos com maior complexidade sejam elaborados. Trabalhos em análise numérica para computadores com arquitetura paralela tem sido publicados desde o surgimento desses equipamentos, como CARROLL e WETHERALD (1967) e LEHMAN (1966), voltados principalmente para otimização matemática, cálculo de raízes, solução de equações diferenciais e resolução de sistemas de equações lineares. Para as aplicações em análise de estruturas cita-se os trabalhos de NOOR e HARTLEY (1978) sobre o cálculo da matriz de rigidez e NOOR e LAMBIOTTE (1979) sobre respostas dinâmicas com o método dos elementos finitos.

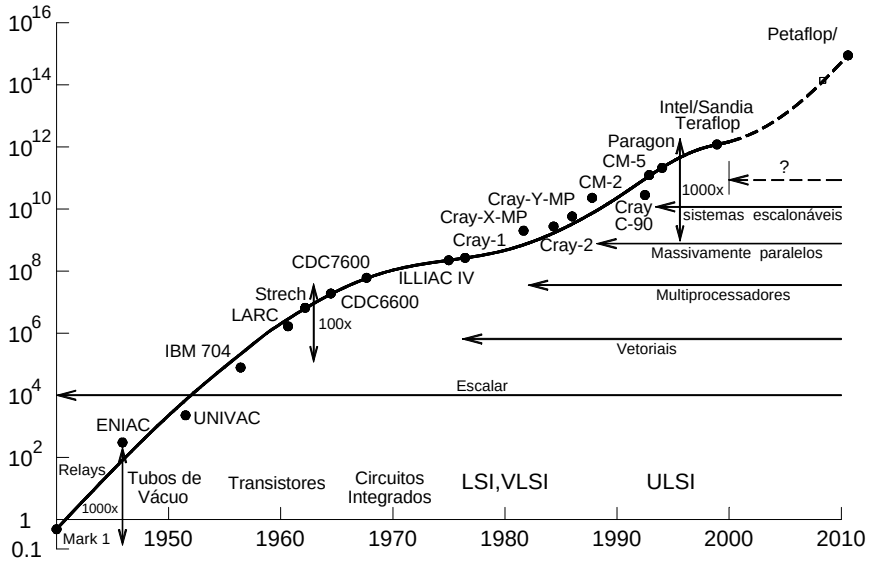


Figura 2.1 Desenvolvimento dos processadores - *apud* NOOR (1997)

Desde então muitos trabalhos tem sido apresentados procurando algoritmos que possam extrair o máximo proveito dos recursos dessa arquitetura. Cita-se por

exemplo LAW(1986) que propõe um método para encontrar os deslocamentos nodais de uma estrutura pelo método dos elementos finitos, em um sistema computacional MIMD sem que seja formada a matriz de rigidez da estrutura usando a técnica da subestruturação.

OU e FULTON(1988) aplicam o processamento paralelo em um modelo de não-linearidade dinâmica, mostrando a eficiência do paralelismo junto aos algoritmos seqüenciais.

FARHAT e WILSON (1988) apresentam algoritmos paralelos para resolução de sistemas de equações lineares usando a fatorização LDL^T do método de Cholesky, tanto para máquinas com memória compartilhada quanto distribuida.

No trabalho de BARRAGY e CAREY (1988) é apresentado um procedimento paralelo para se resolver problemas de valor de contorno com o método dos elementos finitos usando o esquema elemento por elemento. Utiliza-se o método dos Gradientes Conjugados na resolução do sistema de equações lineares.

CARTER et al. (1989) estudam a implementação paralela do método dos elementos finitos usando a técnica de subestruturação sem que seja necessário o cálculo da matriz de rigidez da estrutura. Na resolução do sistema de equações lineares é utilizado o método dos Gradientes Conjugados com Pré-Condicionadores também em paralelo. A implementação é desenvolvida em uma máquina MIMD com rede de interconexão hipercubo.

CHIEN e SUN(1989) apresentam uma proposta de cálculo da matriz de rigidez da estrutura em paralelo, baseada numa numeração especial dos elementos e na divisão da estrutura em subestruturas que são executadas em processadores distintos. Através dessa numeração especial e da divisão em subestruturas, os processadores chegarão aos pontos comuns das subestruturas em tempos distintos, dispensando a sincronização dos processos.

GOEHLICH et al. (1989) apresentam a implementação e o desenvolvimento de um algoritmo paralelo para o método de Cholesky para resolução de sistemas de equações lineares. A implementação é realizada em um FLEX/32 e posteriormente testada em um CRAY X-MP, IBM 3090 e em um ALLIANT.

EL-SAYED e HSIUNG (1990) aplicam a técnica da subestruturação para desenvolver um algoritmo para análise de estruturas em paralelo. O algoritmo é testado em uma estrutura de barras através de um supercomputador CRAY X-MP.

CHIANG e FULTON(1990) mostram avanços na solução de problemas envolvendo não-linearidade e dinâmica, usando-se o processamento paralelo na decomposição de Cholesky para resolver sistemas de equações lineares.

Em JOHNSSON e MATHUR (1990) apresentam-se algoritmos paralelos para pré-processadores, para cálculo das matrizes de rigidez dos elementos finitos da estrutura, e para a resolução do sistema de equações lineares usando o método dos Gradientes Conjugados. Os resultados são obtido em um CM-2

YAGAWA et al. (1991) apresentam um sistema para processamento em paralelo com a técnica de decomposição de domínio em conjunto com o método dos Gradientes Conjugados.

ADELI e KAMAL(1992-I) apresentam duas propostas para o cálculo da matriz de rigidez da estrutura em paralelo. Na primeira divide-se a malha em subestruturas vinculadas aos processos paralelos, recorrendo-se ao uso de semáforos na sincronização dos mesmos. Na segunda além da divisão em subestruturas, faz-se uso de áreas adicionais da memória para cada processo nas regiões da matriz acessíveis por mais de um processador.

No trabalho de SAXENA e PERUCCHIO (1992) é desenvolvido um algoritmo paralelo para geração automática de malhas, baseado no método RSD (Recursive Spatial Decomposition) em elementos sólidos.

SAXENA e PERUCCHIO (1993) apresentam um algoritmo paralelo de divisão de domínio para elementos finitos que pode ser usado com o gerador de malhas paralelo proposto em SAXENA e PERUCCHIO (1992).

LAW e MACKAY (1993) descrevem uma implementação paralela da fatorização LDL^T em computadores com memória distribuída. O algoritmo é verificado no computador Intel i860 com rede de interconexão hipercubo.

Também em RAO (1993) são apresentados algoritmos paralelos para resolução de sistemas de equações lineares em máquinas com memória distribuída. Neste caso estuda-se o método de Gauss com armazenamento em banda e fatorização LDL^T com skyline.

SCHMIT e LAI (1994) apresentam três tipos de pré-condicionadores para o método dos Gradientes Conjugados, que são decomposição incompleta de Cholesky, polinomial e fatorização, além de uma estudo para melhorar a convergência com o vetor inicial. Para verificação dessa proposta é utilizado como exemplo uma treliça espacial.

LIU e WU (1994) apresentam um algoritmo paralelo para análise de estruturas periódicas pelo método dos elementos finitos em computadores MIMD. Aproveita-se o isomorfismo dessas estruturas obtido a partir de um sistema de coordenadas simétrico para gerar sistemas de equações lineares que podem ser analisados separadamente. Os sistemas da estrutura são resolvidos em paralelo.

EL-SAYED e HSIUNG (1994) estabelecem uma comparação entre duas alternativas de otimização estrutural em paralelo. Na primeira, a estrutura é dividida entre os processadores da máquina onde cada processador desenvolve o processo de otimização completamente. Na segunda, a estrutura é analisada de forma completa dividindo-se o gradiente de restrições da otimização entre os processadores.

Em ADELI e KUMAR (1995) apresenta-se uma formulação paralela para o Algoritmo Genético (GA - Genetic Algorithm), utilizado em otimização estrutural.

Em ZHENG e CHANG (1995) propõe-se um algoritmo paralelo para resolução de sistemas de equações lineares armazenado em forma skyline, baseado no método de decomposição de Cholesky para máquinas MIMD com memória compartilhada.

SYNN e FULTON (1995) mostram um processo prático de avaliar a eficiência de um algoritmo paralelo na resolução de sistemas de equações lineares (direto ou iterativo), para os computadores BBN/Butterfly e KSR1.

JAQUES et al. (1996) exploram o paralelismo na análise não-linear geométrica de estruturas, dividindo o cálculo da matriz de rigidez entre dois processos distintos, um responsável pela parte linear e outro pela de correção geométrica. Posteriormente as duas partes são adicionadas, restando apenas um processo com a matriz da estrutura. Nesse trabalho é estabelecido também uma comparação desse procedimento com a técnica de divisão de domínio.

CHEN e BYREDDY (1997) apresentam dois esquemas de paralelismo para o método das faixas finitas para análise de flexão de placas delgadas, baseados nos

paradigmas mestre-escravo e multi-nível mestre-escravo, fazendo uso do PVM em uma rede de workstations.

GEERS e KLEES (1997) desenvolveram um algoritmo paralelo para resolução de sistemas de equações lineares, empregado em máquinas vetoriais SIMD com matriz de coeficientes completa.

No âmbito local, pode-se citar o trabalho de REZENDE (1995) onde apresenta-se com detalhes aspectos introdutórios do processamento paralelo, desenvolvendo algoritmos tanto para o cálculo da matriz de rigidez de uma estrutura quanto para a resolução do sistema de equações lineares. Esses algoritmos são usados na análise não-linear de estruturas em barras em máquinas com memória compartilhada. Ainda na esfera local, ALMEIDA (1999) apresenta a implementação em paralelo de um sistema para resolução de pavimentos utilizando-se o conjunto PIM na resolução do sistema de equações lineares, implementado para máquinas com memória distribuída e biblioteca PVM.

Nesta breve revisão bibliográfica fica claro o interesse que o paralelismo tem despertado nos pesquisadores em análise numérica. Os trabalhos apresentados discutem procedimentos em paralelo para pré-processadores, decomposição de domínio, cálculo da matriz de rigidez, resolução do sistema de equações e otimização, o que evidencia a importância do paralelismo na pesquisa.

Na elaboração de programas em linguagens de alto nível para uso em máquinas seqüenciais, não se necessita conhecer detalhes da arquitetura das mesmas, sendo suficiente o domínio da linguagem que se pretende utilizar. Essa tem sido a conduta adotada até então, de um modo geral, na pesquisa em análise numérica de estruturas. Neste caso, o pesquisador tem como única preocupação a elaboração de modelos mecânicos que melhor representem o comportamento dos elementos estruturais, bastando-lhe o domínio da linguagem de alto nível para verificar a validade do modelo elaborado.

Diferente da postura adotada para os sistemas seqüenciais, quando se pretende desenvolver programas para serem usados em máquinas paralelas é necessário não apenas o domínio das linguagens de programação e ferramentas do paralelismo, mas também as características do equipamento a ser usado, pois nesse caso existe uma relação unívoca entre *software* e máquina. Este fato fica claro na

revisão bibliográfica apresentada. Portanto, ao se desejar fazer uso das potencialidades do paralelismo, obrigatoriamente ter-se-á que estar familiarizado com a estrutura e a organização das arquiteturas paralelas, sendo que a estrutura refere-se ao arranjo estático dos componentes do sistema e organização à interação dinâmica entre os mesmos. A seguir pretende-se abordar aspectos introdutórios da estrutura e organização do paralelismo, além das ferramentas de desenvolvimento e análise dos *softwares* paralelos. Sobre este assunto existe vasta bibliografia disponível, podendo citar-se DeCEGAMA (1989), FREEMAN e PHILLIPS (1992), ALMEIDA e ÁRABE (1991).

2.2 A MÁQUINA DE VON NEUMANN

A organização da grande maioria dos atuais computadores seqüenciais e paralelos deriva do sistema de computação idealizado por von Neumann (1945), conhecido como *máquina de von Neumann* e apresentado na figura 2.2. Esse sistema gerou o conceito de programa armazenado, onde dados e programas compartilham a memória da máquina.

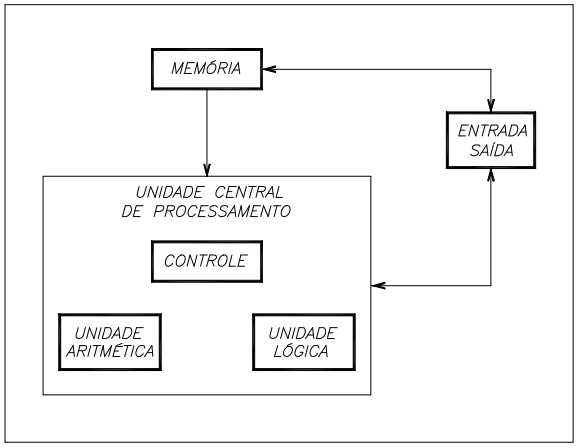


Figura 2.2. Modelo de von Neumann.

Onde :

Entrada : Transmite dados e instruções do ambiente exterior para a máquina

Memória : Armazena instruções, dados e resultados aritméticos.

Unidade Lógico-Aritmética (ALU): Executa as operações lógicas e aritméticas.

Controle : Interpreta as instruções e providencia a execução .

Saída : Transmite os resultados finais para o ambiente exterior.

O processador é o componente que interpreta um programa, sendo constituído de uma memória local (formada por registradores), uma unidade de controle e uma unidade de operação de dados.

O funcionamento da máquina de von Neumann se dá pela execução sequencial de instruções. A execução de cada uma das instruções é feita em um ciclo de seis etapas, conforme apresentado na figura 2.3:

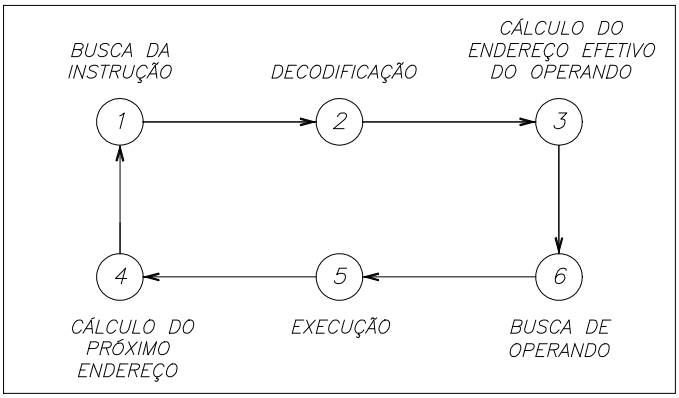


FIGURA 2.3. ETAPAS DA MÁQUINA DE VON NEUMANN

Onde:

1. **Busca da instrução** : A Unidade Central de Processamento (CPU) busca na memória a próxima instrução a ser executada;
2. **Decodificação**: A instrução é decodificada na CPU;
3. **Cálculo do endereço efetivo dos operandos envolvidos na instrução;**
4. **Busca dos operandos na memória**;
5. **Execução da operação**;

6. Cálculo do endereço efetivo da próxima instrução a ser executada.

O fato de existir um caminho de dados entre o processador e a memória consiste na maior deficiência dos processadores convencionais, uma vez que o ciclo da memória é mais lento que o ciclo do processador. Isso torna o tempo de execução da instrução dependente do tempo de acesso à memória. Esse fenômeno é conhecido como “gargalo de von Neumann”.

Além dessa limitação de acesso no sistema de computação proposto por von Neumann, fica claro que o ganho de desempenho em sistemas convencionais é estritamente dependente do desenvolvimento dos componentes desse sistema. Processadores mais rápidos e maior capacidade de armazenamento e rapidez de acesso na memória seriam, então, o alvo natural desse desenvolvimento. Entretanto, além de apresentar elevado custo, esse desenvolvimento, como já se afirmou, tem-se uma limitação física de aperfeiçoamento dos processadores (velocidade da luz no vácuo), o qual não se encontra distante de ser alcançado.

Possibilitando a execução simultânea de uma mesma tarefa em vários processadores, o paralelismo aparece como uma alternativa quase natural de superação das limitações dos sistemas convencionais, obtendo maior desempenho combinado com baixo custo de produção. Com essa arquitetura conseguiu-se desempenho sem que o mesmo estivesse diretamente dependente do desenvolvimento dos componentes eletrônicos.

2.3 CLASSIFICAÇÃO DOS SISTEMAS DE COMPUTAÇÃO.

Antes de discorrer sobre as arquiteturas paralelas disponíveis, é necessário que se estabeleça um padrão em que os sistemas computacionais sejam classificados quanto ao paralelismo que existe na seqüência de instruções e na seqüência de dados. É comum usar a taxonamia proposta por M. FLYNN (1972), apresentada a seguir :

SISD (Single Instruction Single Data) : Processamento de uma sequência de instruções sobre uma sequência de dados, que é o modelo clássico da máquina de von Neumann.

SIMD (Sigle Instruction Multile Data) : Processamento simultâneo da mesma sequência de instruções sobre diferentes dados. Neste caso, a unidade de controle do sistema busca e decodifica a próxima instrução, enviando-a para que seja executada nos vários processadores desse sistema. Existem processadores vetoriais que se enquadram nessa categoria pois efetuam paralelamente a mesma sequência de instrução sobre diferentes elementos de um vetor.

MISD (Multiple Instruction Multiple Data) : Processamento de diferentes instruções sobre mesma sequência de dados. FREEMAN (1992) afirma que ainda não existem máquinas que se enquadram nessa categoria.

MIMD (Multiple Instruction Multiple Data) : Processamento simultâneo de diferentes instruções sobre diferentes tipos de dados. Abrangem amplo espectro de máquinas com processamento paralelo. Um equipamento que pertença a essa categoria pode ser entendido como um conjunto de processadores trabalhando de forma independente ou em conjunto em um problema comum, estando sob o controle de um único sistema operacional. Quantidade de processadores, forma de conexão entre eles, tipo de memória, são os fatores que distinguem essas máquinas.

2.4 PROCESSADORES VETORIAIS

Complexos modelos matemáticos que representam fenômenos da natureza freqüentemente envolvem operações com matrizes e vetores. Com o intuito de dinamizar esses cálculos, foram desenvolvidos processadores que pudessem oferecer formas mais eficazes no tratamento desses problemas. Esses processadores são conhecidos como *vetoriais*. As formas mais usuais como os mesmos trabalham são o *pipelining* (que se traduz por *processamento em duto*), e máquinas SIMD “puras”.

O *pipelining* é semelhante à idéia da linha de montagem em que uma tarefa é dividida em subtarefas executadas em estágios sucessivos. Para exemplificar essa técnica, considere a operação de multiplicar um vetor **U** com **N** elementos, por um escalar **C** resultando no vetor **V**, conforme dado a seguir:

PARA I=1 **ATÉ** N **FAÇA**
V(I) = U (I) * C
FIM FAÇA

Considere o processamento representado apenas pela fase de execução. A tarefa de processar a multiplicação do vetor **U** pelo escalar **C** pode ser dividida em três subtarefas, representadas pelos estágios de busca do elemento de **U** (**U(I)**), realização do produto **C** ∗ **U(I)**, e armazenamento de **C** ∗ **U(I)** em **V(I)**.

Em uma máquina seqüencial esse processamento seria representado pelo diagrama da Figura 2.4. Neste caso, a tarefa completa seria executada em três ciclos de máquina.

ESTÁGIO 1 : Busca U(i)
ESTÁGIO 2 : Faz U(i) * C
ESTÁGIO 3 : Atribuir U(i) * C em V(i)

CICLO	**ESTÁGIOS**
1	1 - Busca U(1)
2	2 - Faz U(1) * C
3	3 - Faz V(1) <= U(1) * C
4	1 - Busca U(2)
5	2 - Faz U(2) * C
6	3 - Faz U(2) <= U(2) * C

Figura 2.4. Execução seqüencial do produto vetor-escalar

Em uma máquina com esquema em *pipeline*, o processamento da tarefa de multiplicar um vetor por um escalar também poderia ser dividida em três estágios diferentes, com cada estágio sendo responsável por uma fase específica do processamento como pode ser visto no diagrama da Figura 2.5. Nota-se então que, neste caso, em cada ciclo de máquina se realizam as três subtarefas que compreendem a multiplicação de um elemento do vetor **U** e seu respectivo armazenamento em **V**. Nesse caso, após iniciado o esquema em pipeline, o tempo para se completar a tarefa é dividido por três.

Ciclo	Estágio 1	Estágio 2	Estágio 3
1	Busca U(1)	Xxxxx	xxxxx
2	Busca U(2)	Faz U(1)*C	xxxxx
3	Busca U(3)	Faz U(2)*C	Faz V(1) ⇐ U(1)*C
4	Busca U(4)	Faz U(3)*C	Faz V(2) ⇐ U(2)*C
5	Busca U(5)	Faz U(4)*C	Faz V(3) ⇐ U(3)*C

Figura 2.5. Multiplicação de um vetor por um escalar em *pipeline*.

Em máquinas SIMD associam-se vários processadores trabalhando com sincronização sob o gerenciamento de uma unidade de controle (Figura 2.6). Neste caso, todos os processadores realizam a mesma instrução sobre diferentes elementos do vetor.

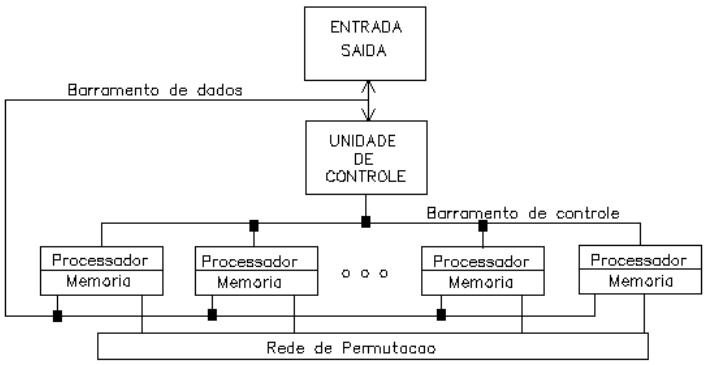


Figura 2.6. Máquina SIMD.

2.5 MULTICOMPUTADORES

Os *multicomputadores* são sistemas de computação compostos de vários processadores independentes apesar da existência de uma rede de comunicação entre si, executando tarefas sobre diferentes sequências de dados. Essas máquinas pertencem à categoria MIMD (múltiplas sequências de instruções e múltiplas sequências de dados). A forma mais comum de um multicomputador está representada na Figura 2.7, consistindo basicamente de um conjunto de processadores com uma rede que proporciona comunicação entre os mesmos. De um

Figura 2.7.Representação gráfica de multicomputadores.

modo geral, pode-se distinguir os seguintes aspectos básicos dessa arquitetura : Organização da memória e formas de interconexão.

No que diz respeito à disposição da memória, os computadores paralelos podem ter vários processadores com acesso a uma única memória (memória global ou compartilhada), ou vários processadores com memória própria (memória local ou distribuída). As máquinas configuradas com memória global apresentam como desvantagem o custo elevado da ampliação da capacidade do conjunto quando se necessita resolver problemas com elevado custo computacional, o que não ocorre com máquinas com memória distribuída. Os sistemas com memória global também estão sujeitos a conflitos de acesso à memória, tanto de hardware quanto de *software*. O conflito de hardware ocorre com dois (ou mais) pedidos de leitura ou gravação simultâneos, implicando na necessidade de sincronização. Quando um processador

altera o valor de uma variável localizada na memória global, que também está em uso por outro processador, verifica-se um conflito de *software*. Nessa situação, deve-se lançar mão das alternativas que o paralelismo oferece quanto à programação.

Os sistemas baseados em memória local apresentam como pontos negativos o baixo desempenho de *softwares* com alta taxa de comunicação entre os processos e a pouca disponibilidade de *softwares* que automaticamente paralelisam o programa. Sua principal vantagem é o menor custo em ampliar o sistema, principalmente se comparado com os sistemas com memória global.

Os sistemas de computação multicomputadores apresentam também outra característica que interfere intrinsecamente no desempenho final da máquina.. Neste tópico apresenta-se resumidamente os modelos básicos dessas redes.

Barramento: É a forma mais simples de interconexão de sistemas multicomputadores. Nesse caso, conecta-se todas as unidade funcionais (processadores, memórias, dispositivos de entrada e saída de dados) a uma rede comum. Todos os processadores tem acesso ao barramento comum, sendo o mesmo conectado à memória global. Nesse caso, somente um acesso ao barramento pode ser efetuado por vez, o que representa um fator limitante de desempenho. Na Figura 2.8 está representado esquematicamente um sistema multiprocessador com barramento.

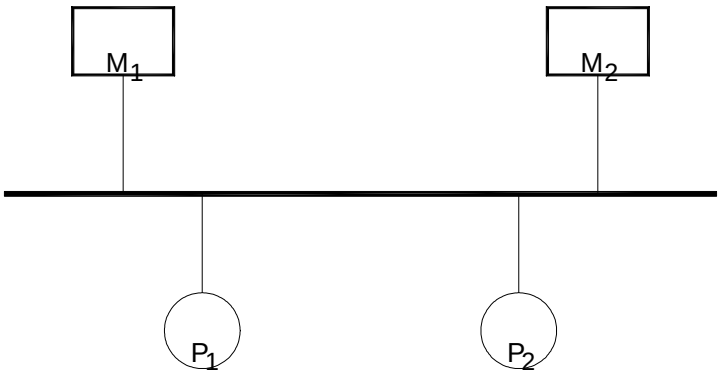


Figura 2.8 Sistema multiprocessador com barramento.

Barramento Cruzado (crossbar): Nesse modelo de interconexão tem-se completa comunicação entre os processadores e os módulos de memória, proporcionando melhor desempenho que o barramento simples, não havendo portanto, atrasos no sistema. Em contrapartida tem custo maior por causa do elevado número de chaveamento (Figura 2.9).

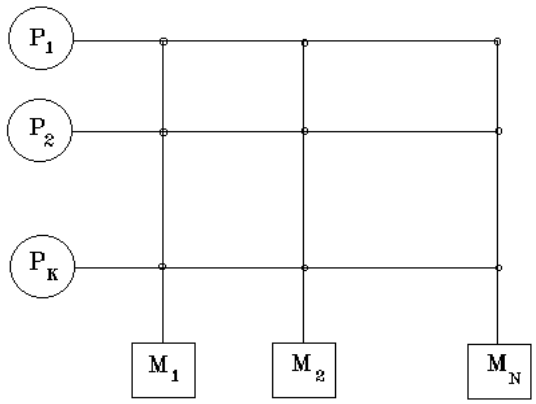


Figura 2.9. Sistema multiprocessador com barramento cruzado.

Rede Matricial: Nesse caso a rede forma uma *malha* bidimensional, sendo que cada processador está conectado aos seus vizinhos. A representação esquemática se encontra na Figura 2.10.

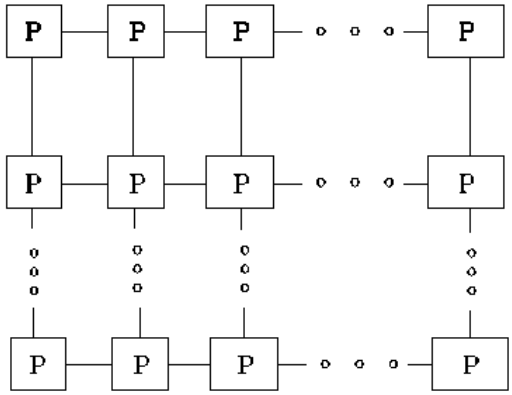


Figura 2.10. Sistema multiprocessador com rede matricial.

Hipercubos: Nessa forma de interconexão tem-se $N = 2^n$ processadores, onde *n* é o número de processadores ligados a cada processador. Pode ser

representado por um cubo como observado na Figura 2.11. Sendo n=3 totaliza-se 8 processadores cada qual ligado a três processadores. Essa classe de multiprocessadores possui memória distribuída e a comunicação entre eles é efetuada por troca de mensagens que pode ser realizada fazendo uso das bibliotecas específicas.

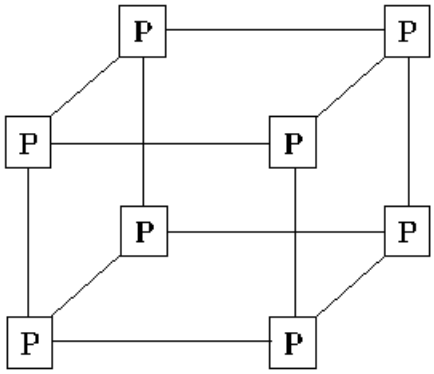


Figura 2.11. Multiprocessador com interconexão hipercubo.

2.6 GRANULARIDADE E SISTEMAS ESCALONÁVEIS

Sistemas de computação em paralelo podem ser descritos como fina, média e grossa granularidade. Sistemas com poucos processadores executando tarefas complexas são ditos de granularidade grossa. Como exemplo citam-se os sistemas CRAY-2, CRAY X-MP, ETA-10 e IBM SP2. Um sistema com granularidade fina é constituído de muitos processadores executando tarefas simples, que podem ser em nível algébrico. O sistema Connection Machine com 65536 processadores é um exemplo de granularidade fina. Sistemas com média granularidade ocupam posição intermediária entre os de fina e grossa, possuindo entre 100 e1000 processadores. Um exemplo dessa máquina é o Sequent Balance 8000.

Sistemas escalonáveis são máquinas que permitem acréscimo de *nós* em seu conjunto inicial (*nó* - conjunto constituído de processador e memória para o caso de sistemas com memória distribuída), possibilitando o aumento da capacidade de

processamento final. Como exemplo desse sistema tem-se o IBM SP2, com número máximo de em 128 nós.

2.7 DESENVOLVIMENTO E ANÁLISE DE SOFTWARES PARALELOS.

Na confecção e análise de *softwares* paralelos existem ferramentas que auxiliam o trabalho do programador. Essas ferramentas estão em forma de bibliotecas de troca de mensagens (PVM, MPI, LINDA, P4, etc) e de parâmetros de análise de desempenho. As bibliotecas são *softwares* que estabelecem um ambiente de comunicação onde é possível a elaboração de programas paralelos. Para esse tipo de paralelismo (sistemas com memória distribuída), o completo domínio dos conceitos de uma biblioteca é condição básica para a implementação em paralelo de qualquer algoritmo. No presente trabalho fez-se uso do PVM (Parallel Virtual Machine), recomendando-se GEIST et al. (1994) como leitura inicial, o qual encontra-se disponível em *post-script* em www.netlib.org/pvm3/book/pvm-book.html . Para o MPI (Message Passing Interface) pode ser encontrado em www.epcc.ed.ac.uk um texto introdutório. Essas duas bibliotecas encontram-se disponíveis no IBM SP2 no Centro de Informática de São Carlos (CISC).

No desenvolvimento de softwares paralelos destacam-se os elementos de sincronização e os parâmetros de análise de desempenho. Esses elementos de desenvolvimento podem estar disponíveis em forma de rotinas, quando se trabalha em sistemas de computação com memória distribuída, ou em forma de comandos para sistemas com memória compartilhada.

Os elementos de análise são definições que permitem avaliar o proveito que o programa desenvolvidos tem extraído dos conceitos do paralelismo. Nos itens anteriores apresentou-se aspectos básicos das configurações e arranjos dos sistemas de computação mais comuns, procurando-se destacar os principais aspectos de cada configuração. A seguir apresentam-se instrumentos para o desenvolvimento e análise de *softwares* paralelos com os quais se procura extrair o máximo proveito dos recursos dos sistemas paralelos já apresentados.

2.7.1 TROCA DE MENSAGENS

Um programa paralelo é caracterizado pela existência de mais de um processo em execução simultânea. É possível que um processo de um programa paralelo necessite de informações que foram executadas em outro processo. Nesse caso deve haver uma troca de informações (mensagens) entre os mesmos para que o programa seja executado corretamente. Uma troca de mensagens ocorre quando um dado é transferido de uma variável em um sub-programa a outra variável em outro sub-programa. Em sistemas com memória distribuída essa troca é efetuada fazendo uso das rotinas disponíveis para esse fim. Para que ocorra uma troca de mensagem satisfatória, tanto para sistemas de memória compartilhada quanto para memória distribuída, as seguintes questões têm que ser respondidas:

- Que processo está enviando a mensagem.
- Onde está o dado no processo que envia a mensagem .
- Que tipo de dado está sendo enviado (inteiro, real, real de dupla precisão, etc).
- Que processo está recebendo a mensagem.
- Onde o dado será alocado (em que variável).
- A quantidade de dados que o processo receptor está preparado para receber.

Além dos requisitos apresentados, tanto o processo emissor quanto o processo receptor têm que estar aptos para efetuar esse procedimento de troca de mensagens de modo que se assegure a correta execução da mesma.

Quanto aos processos envolvidos as trocas de mensagens podem ocorrer de um processo a outro processo, de um processo a vários processos ou de vários processos a um processo. Esses conceitos são comentados em seguida.

Uma troca de mensagens entre dois processos um emissor e outro receptor, é conhecida como Comunicação Ponto-a-Ponto. Mesmo existindo apenas dois processos nessa troca de mensagens, assegurar que a mesma seja efetuada corretamente é fundamental para que o programa tenha execução satisfatória.

Comunicações Coletivas ocorrem quando vários processos estão envolvidos na troca de mensagens. Podem ser de *um processo* a *vários processos* ou ao contrário.

Quando um processo envia uma mensagem a vários processos ocorre o que se chama de *Broadcast* . Esse conceito pode ser esclarecido com a ilustração da Figura 2.11 onde um indivíduo faz uma comunicação simultânea a todos os outros. O mesmo ocorre no processamento paralelo quando um processo envia determinados dados a todos os processos pertencentes a seu grupo através de um comando ou rotina.

Se vários processos em execução em um programa paralelo necessitam enviar uma mensagem a apenas um processo, ocorre o que se chama de *Reduction.* Uma ilustração conhecida, mas útil, é a eleição, onde vários votos se transformam em uma decisão.

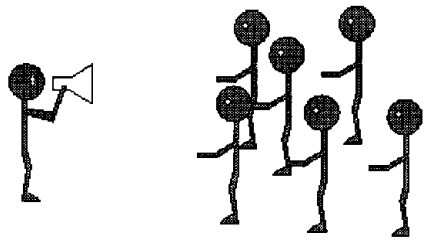


Figura 2.12. Comunicações Coletivas – Broadcast (*apud* www.epcc.ed.ac.uk)

Troca de mensagens podem ocorrer de forma síncrona ou assíncrona. Uma transmissão síncrona não se completa até que a mensagem seja recebida completamente. Na transmissão assíncrona, sabe-se apenas que a mensagem foi enviada, desconhecendo se a mensagem foi recebida ou não. No caso da necessidade de sincronismo de troca de mensagens, os ambientes para desenvolvimento de *softwares* paralelos possuem instrumentos que permitem efetuar uma troca síncrona ou assíncrona. O sincronismo pode ser usado também na execução de processos, que serão comentados no próximo item.

Apesar de sua simplicidade de formulação, esses conceitos de comunicações e sincronismos em troca de mensagens apresentam grande utilidade na elaboração de

softwares paralelos, sendo que em alguns caso pertencem ao próprio algoritmo do programa apresentado.

2.7.2 SINCRONIZAÇÃO DE PROCESSOS.

Um dos fatores que dificulta a paralelização de um programa é a dependência de dados (que será abordada a posteriori). Nesse caso deve-se usar os recursos do paralelismo para se evitar possíveis erros e conflitos, extremamente comuns quando se desenvolve *softwares* paralelos. Recorre-se normalmente à sincronização dos processos que estejam em execução, como também na troca de mensagens efetuada entre os mesmos. Os elementos mais comuns usados na sincronização dos processos são os bloqueios, barreiras e semáforos. Por serem os mais utilizados, apresenta-se a seguir um comentário sobre as definições e utilização dos mesmos.

Bloqueios (*locks*): Esse instrumento de sincronização pode ser usado para impedir que dois ou mais processos acessem simultaneamente variáveis compartilhadas do programa. Esse comando apresenta sua real importância nos trechos onde ocorre a dependência de dados. Quando um determinado processo chega no trecho em que exista o comando bloqueio, o acesso a esse trecho estará impedido a outros processos. Caso os processos em execução tenham variáveis comuns, esse comando impedirá que em cada execução a variável tenha um valor diferente.

Barreiras (*barriers*): Trata-se de um processo síncrono. Permite que o processamento não passe de determinada instrução até que todos os processos cheguem a essa instrução. A dinâmica desse instrumento de sincronização pode ser melhor esclarecido com a Figura 2.12. Supondo que cada processo seja representado por um indivíduo dessa figura, fica claro que esse comando estabelece um ponto no programa a partir do qual os processos só prosseguem na execução quando todos os processos chegaram nesse ponto. Havendo execução paralela de um trecho de programa com uma variável comum a todos os processos, sendo que no trecho sequencial seguinte necessita-se dos valores obtidos em cada processo, esse

instrumento pode ser útil, impedindo que a execução do programa se processe, o que poderia estabelecer resultados incorretos para esta variável.

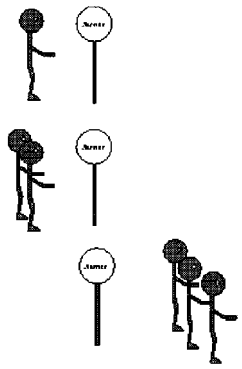


Figura 2.13. Sincronização estabelecida por Barreira (*apud* www.epcc.ed.ac.uk).

Semáforos (*semaphores*): À semelhança do bloqueio é também usado como sinalizador da presença ou não de um processo em uma área onde não se pode ter outro processo concorrente. Quando um processo entra na área de interesse comum, o acesso de outros processo estará proibido. Entretanto, diferente do bloqueio o processo em espera não fica impedido de executar outras instruções. Esse comando é usado em sistemas com memória compartilhada.

2.7.3 AUTOPARALELISADORES

Computação de alto desempenho é amplamente usada no meio científico onde a exigência computacional na execução dos programas é elevada. Pode-se então fazer uso dos sistemas de arquitetura paralela, uma vez que seu objetivo é o ganho de desempenho aliado ao baixo custo de produção. Neste caso é necessário codificar o programa fonte originalmente em modo seqüencial para o modo paralelo. Essa transposição do modo seqüencial para o modo paralelo não é elementar, uma vez que se trata de programas extensos e com elaboração complexa. Para se superar essa dificuldade surgiram os autoparalelisadores (*parallelising compilers*), que são compiladores que identificam trechos do código fonte que são possíveis de serem automaticamente paralelisados. Normalmente esses autoparalelisadores estão

voltados para máquinas com memória global. Apesar de sua praticidade, esses compiladores não conseguem extrair o máximo proveito do paralelismo, uma vez que são codificadas em paralelo apenas os trechos mais visíveis do programa, no que consiste sua maior deficiência.

2.7.4 ANÁLISE DE *SOFTWARES* PARALELOS.

Na confecção de *softwares* paralelos são usados conceitos que além de auxiliar no seu desenvolvimento, aparelham o programador de ferramentas de análise que permitem a verificação da qualidade de seu trabalho. Segue um breve comentário sobre parâmetros frequentemente usados.

Speed-up: Sempre que se desenvolve um *software* paralelo, é essencial saber se o mesmo está extraindo o máximo proveito possível dos recursos e conceitos do paralelismo. Nesse caso, o parâmetro usualmente adotado é o *Speed-up*, definido por:

$$Speed\ -\ up = \frac{tempo\ para\ 1\ processador}{tempo\ para\ \ n\ processadores}$$

Quando se paraleliza um programa, idealmente espera-se que se o mesmo é executado em um tempo *t* com um processador, então esse tempo será reduzido para *t/n* com *n* processadores em execução. Essa relação linear de redução de tempo não se verifica na prática do paralelismo, o que é ocasionado por trechos do programa que não permitem a paralelização, pelo tempo gasto pelo sistema operacional na preparação da execução em paralelo e na degradação de desempenho oriunda da necessidade de sincronização entre os processos.

Eficiência: Em associação com o Speed-up, a *Eficiência* é um importante parâmetro de verificação da qualidade de um *software* paralelo. Esse parâmetro informa o quanto um programa está paralelizável. Está definido por

$$Eficiência = \frac{100.Speed - up}{Número\,de\,Processadores}$$

2.7.5 BENCHMARK

O desempenho dos computadores seqüenciais é usualmente medido em três modos: (1) *Velocidade computacional máxima,* medida em operações de ponto flutuante por segundo (Mflops ou Gflops); (2) *Taxa de execução de operações,* medida em instruções por segundo (Mips ou Gips) (3) *Taxa de operações lógicas por segundo* (Mlips ou Glips). No caso de máquinas SIMD e MIMD a velocidade computacional máxima pode ser estimada. Entretanto, o desempenho destes sistemas de computaçã é difícil de ser medido, uma vez que ele depende do nível de paralelismo de cada sistema. Assim, esse desempenho depende da formulação do algoritmo numérico, da implementação, do compilador, do sistema operacional e também da arquitetura disponível. Neste caso, para se determinar o desempenho desses sistemas, faz-se uso de um conjunto de programa que avalia a capacidade da máquina disponível. Esse conjunto de programas é conhecido como *Benchmark* (*referência*), destacando-se Livermore Fortran Kernels, NAS Kernels, Linpack Benchmark.

2.8 ALGORITMOS NUMÉRICOS E ESTRATÉGIAS COMPUTACIONAIS.

Para se obter o melhor desempenho possível da máquina que se está usando, é necessário estabelecer a melhor estratégia computacional com um algoritmo adequado para o tipo de arquitetura que se dispõe. Intenso trabalho tem sido dedicado ao desenvolvimento de algoritmos vetoriais e paralelos que se aplicam a estes novos sistemas de computação. Para algoritmos paralelos, computações independentes são realizadas concomitantemente. Para se encontrar esse paralelismo o algoritmo é dividido em uma coleção de tarefas independentes, que podem ser executadas simultaneamente comunicando-se com outros processos durante a execução do mesmo. Algoritmos paralelos podem ser caracterizados pelos seguintes fatores:

- Máxima quantidade de cálculo realizada por um processo antes de comunicar-se com outro processo;

- Concordância do algoritmo com a topologia da rede de interconexão;

- Programador com controle sobre o programa no intuito de estabelecer comunicação entre os processos e extrair o máximo proveito do paralelismo.

O desenvolvimento de um algoritmo paralelo apresenta vários problemas que necessitam serem solucionados, tais como troca de dados, armazenamento de dados, acesso à memória, e comunicação entre processos. Geralmente, os algoritmos numéricos em paralelo descritos na literatura estão divididos em duas categorias: reformulação de algoritmos seqüenciais em algoritmos paralelos e algoritmos desenvolvidos especialmente para máquinas paralelas.

A maioria dos trabalhos em algoritmos numéricos paralelos segue a primeira categoria, decompondo os algoritmos seqüenciais em tarefas concorrentes. Exemplos são operações com matrizes, solução de sistemas de equações tanto diretos quanto iterativos, e cálculo de autovalores.

A segunda categoria inclui algoritmos que foram desenvolvidos especialmente para processamento paralelo. Neste caso, o desempenho dos algoritmos é superior ao primeiro caso. Quanto aos algoritmos paralelos é importante destacar as seguintes questões:

- Eficientes algoritmos em paralelo não necessariamente são algoritmos eficientes em modo seqüencial. Do mesmo modo, pode-se deixar de obter a mesma eficiência de um algoritmo seqüencial em um algoritmo em paralelo.

- Algoritmos paralelos podem apresentar propriedades matemáticas diferentes de algoritmos seqüenciais, assim como a taxa de convergência de métodos iterativos pode apresentar módulos diferentes para implementação tanto em paralelo quanto seqüencial.

- Para se conseguir o melhor desempenho possível, o algoritmo a ser implementado necessita de cuidadoso estudo direcionado à máquina disponível. Perde-se nesse caso a oportunidade de se explorar a portabilidade do algoritmo, dada a conjugação do mesmo com a máquina em questão. Também não se pode abrir mão do desempenho em função da portabilidade. Trata-se de uma importante questão na pesquisa em paralelismo. Pode-se resolver esse problema reestruturando os algoritmos em termos do produto matriz-matriz ou matriz-vetor ou implementando-se um algoritmo paralelo que seja independente da arquitetura da máquina.

Usualmente, quatro categorias de paralelismo são identificadas: perfeita, *pipeline*, completamente síncrona e assíncrona. Segue uma descrição sucinta de cada categoria:

Perfeitamente paralelo: São problemas em que os dados podem ser divididos em partes, e os cálculo de cada parte são executados independentemente. A paralelização é relativamente direta e provavelmente com alta eficiência.

Paralelismo em pipeline: É obtido sobrepondo o processamento. Os dados são transportados de um estágio de computação a outro.

Paralelismo completamente síncrono: Cada conjunto de processamento é realizado simultaneamente. Os cálculos seqüentes dependem do resultado do processamento anterior.

Paralelismo assíncrono: É realizado dividindo o trabalho entre vários processadores onde não há dependência de resultados.

2.8.1 DECOMPOSIÇÃO DE DOMÍNIO

A implementação em paralelo do MEF está baseada no conceito de dividir um problema com elevado custo computacional em problemas menores que podem ser tratados individualmente, com troca de informações entre os processos. Esta técnica é conhecida como *Decomposição de Domínio*. A Decomposição de Domínio pode ser descrita como a separação da estrutura em sub-regiões, as quais serão calculadas com troca de informações entre os respectivos processos quando necessário (*Divide-and-Conquer* – Dividir e Conquistar). Há vários modos de se aplicar esta técnica, tanto dividindo-se o domínio da estrutura a ser analisada quanto resolvendo-se o sistema de equações em paralelo. Segundo TOPPING e KHAN (1996) essa técnica apresenta a seguinte classificação:

- *Decomposição de Domínio Explícito* : O problema a ser analisado é dividido em sub-domínios com troca de dados quando necessário.

- *Decomposição de domínio Implícito* : Neste caso o sistema de equações da estrutura é resolvido em paralelo sem que o domínio seja dividido em sub-regiões.

A Decomposição de Domínio Explícito pode ser dividida em dois modos, o Método Iterativo e a Sub-estruturação:

- *Método Iterativo* : É usada para calcular a estrutura de modo iterativo com as informações necessárias para os nós de interface entre dois subdomínios sendo intercambiadas em cada iteração – LAW (1986).

- *Sub-estruturação* : A estrutura é dividida em subregiões, sendo as contribuições de rigidez dos nós internos de cada subregião transferidas para os nós de interface por meio de uma condensação estática. O sistema de equações resultante é resolvido para posteriormente se calcular os resultados dos nós internos – PRZEMIENIECKI (1962). A subestruturação pode ser identificada em

nível algébrico, uma vez que são eliminados graus de liberdade na matriz de rigidez correspondentes aos nós internos das subestruturas. Com isso diminui-se o número de equações do sistema. O desempenho dessa técnica depende do balanceamento de carga entre os processadores e de um modo de reduzir a necessidade de troca de dados entre os mesmos. Trata-se de um modo eficiente de reduzir o custo computacional da análise numérica que envolve elevado número de graus de liberdade - EL-SAYED e HSIUNG (1990).

Ao se aplicar a técnica da decomposição de domínio deve-se estar atento para se obter o melhor aproveitamento possível do paralelismo. Esse melhor aproveitamento pode ser alcançado com um balanceamento de carga adequado. Entende-se por balanceamento de carga a divisão de tarefas entre os processadores de tal modo que o tempo consumido em cada processador seja o mesmo (ou aproximadamente iguais). Com a técnica de Decomposição de Domínio, pode-se conseguir esse objetivo com a otimização do problema, isto é:

- Número de elementos finitos iguais em cada subdomínio;

- Número de nós de interface entre os subdomínios aproximadamente iguais.

No caso da Decomposição de Domínio Implícito, pode-se conseguir um balanceamento de carga adequado dividindo-se o sistema de equações lineares com igual número de linhas, ou colunas, entre os processadores.

2.8.2 GERADORES DE MALHAS.

O tratamento matemático de diversos problemas da natureza é, em muitos casos, de difícil solução. O Método dos Elementos Finitos (MEF) é uma técnica que permite encontrar a solução numérica destes problemas por meio da discretização do domínio do problema a ser estudado. Nesse caso, o resultado numérico tanto se aproxima da solução analítica quanto melhor se representa o domínio com uma

malha adequada. O desenvolvimento de algoritmos que automaticamente encontrem a malha adequada para o problema em questão (*Adaptive Mesh Generation*) é um modo eficiente de se evitar o erro humano na geração de malhas – JOHNSSON e MATHUR (1990), TOPPING e KHAN (1996). A implementação em máquinas de arquitetura paralela de algoritmos para geração automática de malhas constitui uma importante área de pesquisa do paralelismo em elementos finitos.

CAPÍTULO III

ANÁLISE NÃO-LINEAR DE CASCAS

3.1 INTRODUÇÃO

Uma casca é uma estrutura curva que possui uma de suas dimensões (a espessura) bastante inferior às outras. ZIENKIEWICKS (1991) afirma que pode-se obter uma casca a partir de uma placa delgada transformando o seu plano médio em uma superfície curva. Essa forma geométrica confere às cascas a presença simultânea de estados de flexão e de membrana. O estado de flexão decorre da distribuição não uniforme de tensões ao longo da espessura da casca, resultando em esforços que tendem a fletir a estrutura. Para o efeito de membrana as tensões são constantes na espessura, resultando em esforços normais tangenciais e cortantes. Essa combinação de estados permite às cascas a capacidade de vencer grandes vãos com estruturas delgadas.

Apesar de tratar-se de uma estrutura tridimensional, as teorias desenvolvidas considerando-se um elemento bidimensional apresentam resultados satisfatórios. Mesmo com a aproximação por uma estrutura bidimensional, o tratamento matemático clássico é difícil e trabalhoso. O Método dos Elementos Finitos (MEF) é uma poderosa ferramenta na análise de cascas. Usando-se este método pode ser feita uma análise considerando tanto uma estrutura tridimensional quanto bidimensional. A aproximação por uma estrutura bidimensional, entretanto, apresenta vantagens por conjugar maior simplicidade na sua formulação e resultados consistentes. Neste caso, pode-se dispor de elementos finitos curvos e planos. Os elementos finitos curvos são

elementos que se adaptam à curvatura da casca e tem como principal característica a rápida convergência, mesmo com malhas pouco refinadas. Apresentam, entretanto, uma formulação complexa e dificuldade de acoplamento com elementos de viga. A aproximação por elementos finitos planos, que se compõem de um elemento de chapa e um de placa, reduz a complexidade da formulação além de facilitar o acoplamento com o elemento de viga. Neste caso, a convergência depende dos elementos utilizados (chapa e placa) e do refinamento da malha adotada. Não possuem a mesma convergência que os elementos curvos, entretanto, dada a menor complexidade de implementação, têm sido largamente usados na análise de cascas. Tanto a formulação para elementos planos quanto para elementos curvos pode ser encontrada em ZIENKIEWICKS (1991) e OÑATE (1992).

A aproximação de cascas por elementos finitos planos é adotada neste trabalho. Neste capítulo apresenta-se a formulação com os respectivos elementos utilizados. É apresentado também o modelo de plasticidade adotado na análise não-linear de cascas.

3.2 O ELEMENTO FINITO DE CASCA PLANO

Na formulação adotada neste trabalho, considera-se como hipótese simplificadora que as tensões variam linearmente ao longo da espessura da casca quando da consideração do efeito de flexão. Também considera-se desprezível o efeito do esforço de cisalhamento. Essas considerações podem ser resumidas na hipótese de Kirchhoff para flexão de placas delgadas:" *Pontos sobre a normal à superfície média indeformada permanecem sobre a normal à superfície média deformada*". Além da hipótese de Kirchhoff, admite-se também que as tensões normais à superfície sejam desprezíveis em comparação com as demais, que a espessura da casca seja bem menor que as outras dimensões da mesma e que os deslocamentos sejam pequenos em comparação com suas dimensões.

Considerando-se válidas as hipótese anteriores, o deslocamento de um ponto sobre a normal à superfície média será dado por:

$$u(x,y,z) = u'(x,y) - z\,\partial w/\partial x$$

$$v(x,y,z) = v'(x,y) - z\,\partial w/\partial y \qquad (3.1)$$
$$w(x,y,z) = w'(x,y)$$

A partir dos deslocamentos dados pelas relações (3.1), pode-se encontrar as deformações para um ponto qualquer, que serão dadas por:

$$\begin{aligned} \varepsilon_x &= \frac{\partial u}{\partial x} = \frac{\partial u(x,y)}{\partial x} - z\frac{\partial^2 w}{\partial x^2} \\ \varepsilon_y &= \frac{\partial v}{\partial x} = \frac{\partial v(x,y)}{\partial x} - z\frac{\partial^2 w}{\partial y^2} \\ \gamma_{xy} &= \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = \frac{\partial u(x,y)}{\partial y} + \frac{\partial v(x,y)}{\partial y} - 2z\frac{\partial^2 w}{\partial x \partial y} \end{aligned} \qquad (3.2)$$

As equações (3.2) podem ser expressas por:

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}_c + \boldsymbol{\varepsilon}_p \qquad (3.3)$$

$$\boldsymbol{\varepsilon} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix}, \quad \boldsymbol{\varepsilon}_c = \begin{Bmatrix} \dfrac{\partial u}{\partial x} \\ \dfrac{\partial v}{\partial y} \\ \dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x} \end{Bmatrix} \quad \text{e} \quad \boldsymbol{\varepsilon}_p = -z\begin{Bmatrix} \dfrac{\partial^2 w}{\partial x^2} \\ \dfrac{\partial^2 w}{\partial y^2} \\ \dfrac{\partial^2 w}{\partial x \partial y} \end{Bmatrix}$$

Os subíndices *c* e *p* referem-se aos estados de chapa e placa, respectivamente. Assim, as tensões serão dadas por:

$$\boldsymbol{\sigma} = \mathbf{D}\,\boldsymbol{\varepsilon} = \mathbf{D}\,(\boldsymbol{\varepsilon}_c + \boldsymbol{\varepsilon}_p) = \boldsymbol{\sigma}_c + \boldsymbol{\sigma}_p, \qquad (3.4)$$

onde **D** é a matriz dos coeficientes elásticos.

A partir das relações constitutivas da teoria da elasticidade é possível estabelecer as relações básicas do método dos elementos finitos. Para tanto é

necessário que os deslocamentos no domínio do elemento sejam definidos por uma expansão polinomial. Então, para o estado de chapa, impondo deslocamentos nodais nos vértices do elemento, obtém-se

$$\mathbf{U}_c = \mathbf{\Phi}_c \ \mathbf{V}_c \tag{3.5}$$

Na equação (3.5) $\mathbf{\Phi}_c$ representa a matriz das funções de forma dos deslocamentos e $\mathbf{V}_c$ representa o vetor de deslocamentos nodais nos vértices do elemento. Neste trabalho adota-se um elemento finito de chapa triangular com três pontos nodais e três parâmetros por nó. O vetor de deslocamentos é dado por $\mathbf{V}_c^T = \{ u_1 \ v_1 \ \theta_{z1} \ .... \ \theta_{z3} \}$

Com as relações anteriores obtém-se o vetor de deformações em função de parâmetros nodais e de funções de forma. Assim:

$$\boldsymbol{\varepsilon}_c = \mathbf{L}_c \mathbf{\Phi}_c \mathbf{V}_c = \mathbf{B}_c \ \mathbf{V}_c \tag{3.6}$$

$$\mathbf{B}_c = \mathbf{L}_c \mathbf{\Phi}_c \quad \text{e} \qquad \mathbf{L}_c^T = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & \frac{\partial}{\partial y} \\ 0 & \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix}$$

O elemento finito de placa adotado também é triangular com três pontos nodais e três parâmetros por nó. O vetor de deslocamentos nodais é dado por $\mathbf{V}_p^T = \{ w_1 \ \ \theta_{x1} \ \ \theta_{y1} \ \ \theta_{y3} \}$. Do mesmo modo obtém-se o vetor de deformações para o estado de placa, dado por :

$$\boldsymbol{\varepsilon}_p = z \, \mathbf{L}_p \ \mathbf{\Phi}_p \ \mathbf{V}_p = \mathbf{B}_p \mathbf{V}_p \tag{3.7}$$

$$\mathbf{B}_p = \mathbf{L}_p \mathbf{\Phi}_p , \quad \mathbf{L}_p^T = - \begin{bmatrix} \frac{\partial^2}{\partial x^2} & \frac{\partial^2}{\partial y^2} & \frac{\partial^2}{\partial x \partial y} \end{bmatrix}$$

Com as relações anteriores chega-se à deformação do elemento de casca:

$$\boldsymbol{\varepsilon}_s = \mathbf{B}_s \mathbf{V}_s = [\ \mathbf{B}_c \quad z\,\mathbf{B}_p\ ]\ \mathbf{V}_s \tag{3.8}$$

$$\mathbf{V}_s{}^T = \{\ \mathbf{V}_c \quad \mathbf{V}_p\ \}$$

A expressão da matriz de rigidez de um elemento qualquer é dada por

$$\mathbf{K} = \int_V \mathbf{B}_s{}^T\, \mathbf{D}\, \mathbf{B}_s\, dV \tag{3.9}$$

Assim, utilizando as expressões para $\mathbf{B}_s$ resulta em:

$$\mathbf{K} = \int_v \begin{bmatrix} \mathbf{B}_c^T \\ \mathbf{zB}_p^T \end{bmatrix} \mathbf{D} \begin{bmatrix} \mathbf{B}_c & \mathbf{zB}_p \end{bmatrix} dV \tag{3.10}$$

A matriz de rigidez pode ser expressa como:

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{cc} & \mathbf{K}_{cp} \\ \mathbf{K}_{cp}^T & \mathbf{K}_{pp} \end{bmatrix} \tag{3.11}$$

Os termos expressos na equação (3.11) referem-se às matrizes de rigidez dos elementos de chapa, placa e acoplamento dadas por:

$$\begin{aligned} \mathbf{K}_{cc} &= \int_V \mathbf{B}_c{}^T\, \mathbf{D}\, \mathbf{B}_c\, dV \\ \mathbf{K}_{pp} &= \int_V \mathbf{B}_p{}^T\, z^2\, \mathbf{D}\, \mathbf{B}_p\, dV = \int_V \mathbf{B}_p{}^T\, \mathbf{D}_p\, \mathbf{B}_p\, dV \\ \mathbf{K}_{cp} &= \int_V \mathbf{B}_c{}^T\, z\mathbf{D}\, \mathbf{B}_p\, dV \end{aligned} \tag{3.12}$$

A equação (3.11) representa a forma completa da matriz de rigidez do elemento de casca. Para análises elásticas de materiais homogêneos, o termo $\mathbf{K}_{cp}$ se iguala a zero, restando as matrizes dos elementos de chapa e placa.

É importante destacar que esse equacionamento refere-se ao elemento finito dado em coordenadas locais. Quando se calcula a matriz de rigidez da estrutura, as matrizes locais devem ser rotacionadas para um sistema de referência global. Usando a matriz de cossenos diretores adequada **R**, tem-se:

$$\mathbf{K}_g = \mathbf{R}^T \mathbf{K} \mathbf{R} \tag{3.13}$$

Com a relação anterior calcula-se a matriz de rigidez da estrutura. Após a resolução do sistema, os esforços podem ser determinados integrando-se as tensões ao longo da espessura da casca, ou seja:

$$\mathbf{N} = \begin{Bmatrix} N_x \\ N_y \\ N_{xy} \end{Bmatrix} = \int_{-h/2}^{h/2} \begin{Bmatrix} \acute{o}_x \\ \acute{o}_y \\ \acute{o}_{xy} \end{Bmatrix} dz = \int_{-h/2}^{h/2} \mathbf{\acute{o}}\, dz$$

$$\mathbf{M} = \begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} = \int_{-h/2}^{h/2} \begin{Bmatrix} z\,\acute{o}_x \\ z\,\acute{o}_y \\ z\,\acute{o}_{xy} \end{Bmatrix} dz = \int_{-h/2}^{h/2} z\,\mathbf{\acute{o}}\, dz \tag{3.14}$$

Na análise elastoplástica calcula-se o vetor resíduo pela diferença entre o carregamento externo e a resultante das forças internas. Em cada iteração em um incremento de carga, o resíduo é reaplicado na estrutura até que o erro atinja a tolerância desejada. Esse cálculo pode ser expresso pela relação:

$$\mathbf{\Psi} = \mathbf{F}_{ext} - \mathbf{F}_{int} \tag{3.15}$$

Na equação (3.15) $\mathbf{\Psi}$ representa o resíduo a ser determinado, $\mathbf{F}_{ext}$ é o vetor de forças externas dado pela parcela do carregamento total correspondente ao incremento aplicado e $\mathbf{F}_{int}$ é o vetor de forças internas, resultado da integração das tensões calculadas. Para o caso de análise elastoplástica com o método dos elementos finitos, $\mathbf{F}_{int}$ é determinado em cada ponto nodal. Então F_{int} é dado por :

$$\mathbf{F}_{int} = \begin{Bmatrix} \mathbf{F}_{int,c} \\ \mathbf{F}_{int,p} \end{Bmatrix} \tag{3.16}$$

$$\mathbf{F}_{int,c} = \int_v \mathbf{B}_c^T \, \sigma \, dv \qquad \mathbf{F}_{int,p} = \int_v \mathbf{B}_p^T \, \mathbf{M} \, dv$$

As integrais indicadas na equação (3.16) são calculadas no domínio do elemento. Para tanto realiza-se a integração numericamente utilizando a quadratura de Gauss.

3.3 O ELEMENTO FINITO DE CHAPA

O elemento finito de chapa utilizado na composição do elemento de casca plano é triangular com três pontos nodais e três parâmetros por nó. Foi apresentado por BERGAN e FELIPPA (1985) utilizando o conceito de Formulação Livre e apresenta como principal vantagem o fato de incorporar uma rotação no plano do elemento como grau de liberdade. A inclusão desse grau de liberdade possibilita que se evite a utilização da rigidez fictícia quando se implementa elementos de casca planos. Este elemento é desenvolvido pela aplicação direta do teste do elemento individual na sua formulação. Neste caso, supõe-se que o campo de deslocamentos no interior do elemento seja resultado da contribuição de modos de deformação constante e corpo rígido com modos superiores, ou seja:

$$\mathbf{u} = \mathbf{N}_{rc} \, \alpha_{rc} \; + \mathbf{N}_s \, \alpha_s \, , \tag{3.17}$$

onde $\mathbf{N}_{rc}$ representa a expansão polinomial completa para os modos de corpo rígido e deformação constante, $\mathbf{N}_s$ a expansão para os modos superiores e α_{rc} e α_s representam os respectivos vetores de parâmetros generalizados. Ao se aplicar o campo de deslocamentos da equação (3.17) nos pontos nodais, verifica-se um conjunto de equações dado por :

$$\mathbf{V}_e = [\, \mathbf{G}_{rc} \quad \mathbf{G}_s \,] \begin{Bmatrix} \mathbf{á}_{rc} \\ \mathbf{á}_s \end{Bmatrix} = \mathbf{G} \, \mathbf{á} \tag{3.18}$$

$$\alpha = \mathbf{G}^{-1}\,\mathbf{V}_e = \mathbf{H}\,\mathbf{V}_e$$

Da equação (3.18) verifica-se que :

$$\begin{aligned}
&\mathbf{H}_{rc}\,\mathbf{G}_{rc} = \mathbf{I} \\
&\mathbf{H}_{s}\,\mathbf{G}_{s} = \mathbf{I} \\
&\mathbf{H}_{rc}\,\mathbf{G}_{s} = \mathbf{0} \\
&\mathbf{H}_{s}\,\mathbf{G}_{rc} = \mathbf{0} \\
&\mathbf{G}\,\mathbf{H} = \mathbf{G}_{rc}\,\mathbf{H}_{rc} + \mathbf{G}_{a}\,\mathbf{H}_{a} = \mathbf{I} \\
&\mathbf{H}^{T}\,\mathbf{G}^{T} = \mathbf{H}_{rc}^{T}\,\mathbf{G}_{rc}^{T} + \mathbf{H}_{a}^{T}\,\mathbf{G}_{a}^{T} = \mathbf{I}
\end{aligned} \tag{3.19}$$

O teste do elemento individual estabelece que na interação de um elemento com seus vizinhos ele seja capaz de reproduzir um estado de corpo rígido e deformação constante. Assim, para um estado qualquer, as forças produzidas são dadas por:

$$\mathbf{t}_{rc} = \mathbf{L}_{rc}\,\sigma_{rc} \tag{3.20}$$

onde $\mathbf{t}_{rc}$ representa as forças nodais produzidas, $\mathbf{L}_{rc}$ é chamada de matriz amontoadora (distribui as tensões para os pontos nodais) e σ_{rc} as tensões distribuídas na face do elemento. Da equação anterior tem-se:

$$\mathbf{t}_{rc} = \mathbf{L}_{rc}\,\mathbf{D}\,\varepsilon_{rc} = \mathbf{L}_{rc}\,\mathbf{D}\,\mathbf{B}_{rc}\,\alpha_{rc} = \mathbf{P}_{rc}\,\alpha_{rc} \tag{3.21}$$

Considerando-se um elemento tem-se

$$\mathbf{F}_{rc} = \mathbf{K}^{e}\,\mathbf{V}_{rc}^{\,e} = \mathbf{K}^{e}\,\mathbf{G}_{rc}\,\alpha_{rc}\,, \tag{3.22}$$

onde $\mathbf{F}_{rc}$ é o vetor de forças nodais, $\mathbf{K}^{e}$ é a matriz de rigidez do elemento e $\mathbf{V}_{rc}^{\,e}$ e o vetor de deslocamentos nodais. Assim, o teste estabelece que:

$$\mathbf{t}_{rc} = \mathbf{F}_{rc} \tag{3.23}$$

Portanto

$$\mathbf{P}_{rc} = \mathbf{K}^{e}\,\mathbf{G}_{rc} \tag{3.24}$$

Utilizando a formulação do MEF pela definição da energia potencial, obtém-se a expressão da matriz de rigidez para elementos cuja expansão do campo de deslocamentos seja dado pela equação (3.17):

$$U = \frac{1}{2}\,\boldsymbol{\acute{a}}^{T} \int \mathbf{B}^{T}\mathbf{D}\,\mathbf{B}\;dV\,\boldsymbol{\acute{a}} = \frac{1}{2}\,\boldsymbol{\acute{a}}^{T}\,\mathbf{K}_{q}\,\boldsymbol{\acute{a}} \tag{3.25}$$

$$\mathbf{B}^{T} = [\ \mathbf{B}_{rc}{}^{T} \quad \mathbf{B}_{s}{}^{T}\ ], \qquad \mathbf{K}_{q} = \begin{bmatrix} \mathbf{K}_{rc} & \mathbf{K}_{rcs} \\ \mathbf{K}_{rcs}^{T} & \mathbf{K}_{s} \end{bmatrix}$$

Ao se utilizar a equação (3.18) tem-se:

$$U = \frac{1}{2}\,\mathbf{V}_{e}^{T}\,\mathbf{H}^{T}\,\mathbf{K}_{q}\,\mathbf{H}\,\mathbf{V}_{e}^{T} = \frac{1}{2}\,\mathbf{V}_{e}^{T}\,\mathbf{K}\,\mathbf{V}_{e} \tag{3.26}$$

Ao se expandir a equação anterior, obtém-se para a matriz de rigidez:

$$\mathbf{K} = \mathbf{H}_{rc}^{T}\,\mathbf{K}_{rc}\,\mathbf{H}_{rc} + \mathbf{H}_{rc}^{T}\mathbf{K}_{rcs}\,\mathbf{H}_{s} + \mathbf{H}_{s}^{T}\,\mathbf{K}_{rcs}^{T}\,\mathbf{H}_{rc} + \mathbf{H}_{s}^{T}\,\mathbf{K}_{s}\,\mathbf{H}_{s} \tag{3.27}$$

A equação anterior representa a matriz de rigidez para o elemento finito cujo campo de deslocamentos seja dado por uma expansão de modos de corpo rígido e deformação constante com modos superiores. Segundo BERGAN e NYGARD (1984), ao se aplicar as equações (3.19) na equação anterior, verifica-se que a mesma satisfaz o teste do elemento individual quando são impostas condições de ortogonalidade em força e energia, ou seja:

$$\mathbf{G}_{s}{}^{T}\,\mathbf{P}_{rc} = \mathbf{0} \tag{3.28}$$

$$\mathbf{K}_{rcs} = \int \mathbf{B}_{rc}{}^{T}\, \mathbf{D}\, \mathbf{B}_{s}\, dv = \mathbf{0} \tag{3.29}$$

A imposição das condições de ortogonalidade em força e energia na formação da matriz de rigidez do elemento, por fim, induziam a uma inconsistência ao produzir uma assimetria na matriz. Com o intuito de superar esse obstáculo, foi proposta uma modificação na mesma, acrescentando o termo $\mathbf{P}_{rc}{}^{T}\, \mathbf{G}_{s}$ na matriz generalizada, ou seja:

$$\mathbf{K}_{q} = \begin{bmatrix} \mathbf{K}_{rc} & \mathbf{P}_{rc}^{T}\mathbf{G}_{s} \\ \mathbf{G}_{h}^{T}\mathbf{P}_{rc} & \mathbf{K}_{s} \end{bmatrix} \tag{3.30}$$

A modificação proposta satisfaz o teste do elemento individual. Mesmo assim, ao se refinar a malha dos exemplos executado, não se obtinha a convergência desejada. Notou-se que a modificação proposta, mesmo satisfazendo o teste do elemento individual, originava autovalores negativos na matriz de rigidez. Desse modo, BERGAN e NYGARD (1984) sugeriram outra modificação na matriz de rigidez, dada pela adição do termo $\mathbf{G}_{s}{}^{T}\, \mathbf{K}_{rc}\, \mathbf{G}_{s}$ no termo correspondente aos modos superiores. Portanto tem-se:

$$\mathbf{K}_{q} = \begin{bmatrix} \mathbf{K}_{rc} & \mathbf{P}_{rc}^{T}\mathbf{G}_{s} \\ \mathbf{G}_{h}^{T}\mathbf{P}_{rc} & \mathbf{K}_{s} + \mathbf{G}_{s}^{T}\mathbf{K}_{s}\mathbf{G}_{s} \end{bmatrix} \tag{3.31}$$

Por fim, com a equação anterior obtém-se a matriz de rigidez do elemento finito, que será dada por:

$$\mathbf{K} = \begin{bmatrix} \mathbf{H}_{rc}^{T} & \mathbf{H}_{s}^{T} \end{bmatrix} \mathbf{K}_{q} \begin{bmatrix} \mathbf{H}_{rc} \\ \mathbf{H}_{s} \end{bmatrix} \tag{3.32}$$

Fazendo uso das equações (3.19), a equação anterior pode ser expressa por:

$$\mathbf{K} = \frac{1}{V}\, \mathbf{L}\ \mathbf{D}\ \mathbf{L}^{T} + \mathbf{H}_{s}^{T}\ \mathbf{K}_{s}\ \mathbf{H}_{s} \tag{3.33}$$

ou,

$$\mathbf{K} = \mathbf{K}_b + \mathbf{K}_h \tag{3.34}$$

Na equação (3.34) $\mathbf{K}_b$ representa a matriz referente aos modos básicos (**L** é a matriz amontoadora) e $\mathbf{K}_s$ a parcela referente aos modos superiores. Esta equação caracteriza a expressão “Formulação Livre”, pois elimina a exigência da condições de ortogonalidade em força e energia para satisfação do teste do elemento individual quando se expande o campo de deslocamentos em modos básicos (corpo rígido e deformação constante) e modos superiores.

Fazendo uso dessa formulação, BERGAN e FELIPPA (1986) desenvolveram um elemento finito de membrana que incorpora a rotação nos vértices como grau de liberdade. Para tanto utilizaram a definição da mecânica do contínuo dada por:

$$\grave{e}_z = \frac{1}{2}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) \tag{3.35}$$

O campo de deslocamentos dos modos básico e superior é dado por :

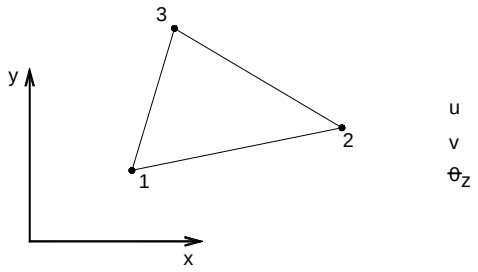
$$\mathbf{u}_b = \begin{bmatrix} 1 & 0 & -ç & \hat{i} & 0 & ç \\ 0 & 1 & \hat{i} & 0 & ç & \hat{i} \end{bmatrix} \mathbf{á}_{rc} \tag{3.36}$$

$$\mathbf{u}_s = \begin{Bmatrix} u_s \\ v_s \end{Bmatrix} = \begin{bmatrix} \hat{i}ç & -\frac{1}{2}ç^2 & \hat{i}^2 \\ -\frac{1}{2}\hat{i}^2 & \hat{i}ç & ç^2 \end{bmatrix} \mathbf{á}_s \tag{3.37}$$

onde *x* e *h* são coordenadas adimensionais dadas por :

$$\hat{i} = \frac{x - x_c}{\sqrt{A}} = ë\,(x - x_c); \qquad ç = \frac{y - y_c}{\sqrt{A}} = ë(x - x_c)$$

Nas relações para coordenadas adimensionais, *A* representa a área do elemento e x_c e y_c são as coordenadas do centro de gravidade. O vetor de deslocamentos nodais é dado por:



$$\mathbf{V}_e^T = \{\ u_1\ \ v_1\ \ \theta_{z1}\ \ u_1\ \ v_1\ \ \theta_{z2}\ \ u_1\ \ v_1\ \ \theta_{z3}\ \} \tag{3.38}$$

Figura 3.1 Elemento finito de chapa

Em análise não-linear, a matriz de coeficientes elásticos varia ponto a ponto no domínio do elemento. Portanto, é necessário calcular a matriz de rigidez pela integração numérica. Neste caso utiliza-se a relação:

$$\mathbf{K} = \int \mathbf{B}^T\, \mathbf{D}\, \mathbf{B}\, dV \tag{3.39}$$

$$\mathbf{B} = \frac{1}{V}\mathbf{L}^T + \sqrt{\hat{a}}\, \mathbf{B}_s\, \mathbf{H}_s$$

3.4 O ELEMENTO FINITO DE PLACA

Na análise de estruturas por elementos finitos é fundamental dispor de elementos que conjuguem rápida convergência e facilidade de formulação. Para análise de placas delgadas o elemento DKT (Discrete Kirchhoff Theory), apresentado em BATOZ (1980), mostrou-se eficiente para este estudo. Trata-se de um elemento triangular com três pontos nodais e três parâmetros por nó (um deslocamento perpendicular ao plano médio e duas rotações), conforme figura 3.2. Na sua formulação utiliza-se a hipótese de Kirchhoff para flexão de placas delgadas:" *Pontos da placa sobre a normal à superfície média indeformada permanecem sobre a normal à superfície média deformada* ". A hipótese de Kirchhoff é imposta discretamente nos nós do contorno do elemento.

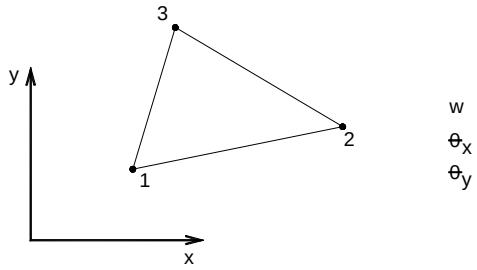


Figura 3.2. Elemento finito de placa – DKT.

Além da hipótese de Kirchhoff para flexão de placas delgadas, considera-se desprezíveis as tensões normais a esse plano. Assim, os deslocamentos de um ponto qualquer serão dados por :

$$\begin{aligned} u &= z\,\beta_x\,(x\,,\,y) \\ v &= z\,\beta_y\,(x\,,\,y) \\ w &= w\,(x\,,\,y) \end{aligned} \qquad (3.40)$$

Na equação (3.40) β_x e β_y são as rotações da normal ao plano médio indeformado e *z* a distância do ponto ao plano médio. Na formulação desse elemento, inicialmente considera-se as rotação β_x e β_y com acréscimo da contribuição do esforço cortante, e posteriormente impõe-se a hipótese de Kirchhoff nos nós, desprezando-se também a contribuição da energia de deformação correspondente a esse esforço.

A formulação desse elemento está baseada nas seguintes hipóteses:

1. *As rotações b_x e b_y variam quadraticamente sobre o elemento.*

2. *A hipótese de Kirchhoff é imposta ao longo dos lados do elemento.*

3. *O deslocamento vertical ao longo dos lados é cúbico.*

4. A rotação da normal tem comportamento linear ao longo dos lados.

Usando-se as hipóteses citadas e as relações geométricas no triângulo, obtém-se:

$$\beta_x = \mathbf{H}_x^T(\xi,\eta)\,\mathbf{U}$$
$$\beta_y = \mathbf{H}_y^T(\xi,\eta)\,\mathbf{U} \qquad (3.41)$$

Na equação (3.41) $\mathbf{H}_x$ e $\mathbf{H}_y$ representam as matrizes que compõem as funções de forma do elemento e **U** o vetor de parâmetros nodais. Com as relações anteriores tem-se:

$$\kappa = \begin{bmatrix} _{x,x} \\ _{y,y} \\ _{x,y} + _{y,x} \end{bmatrix} = \mathbf{B}\,\mathbf{U} \qquad (3.42)$$

$$\mathbf{B}(\hat{\imath},\varsigma) = \frac{1}{2A}\begin{bmatrix} y_{31}\,\mathbf{H}^T_{x,\hat{\imath}} + y_{21}\,\mathbf{H}^T_{x,\varsigma} \\ -x_{31}\,\mathbf{H}^T_{y,\hat{\imath}} - x_{12}\,\mathbf{H}^T_{x,\varsigma} \\ -x_{31}\,\mathbf{H}^T_{x,\hat{\imath}} - x_{12}\,\mathbf{H}^T_{x,\varsigma} + y_{31}\,\mathbf{H}^T_{y,\hat{\imath}} + \mathbf{y}_{21}\,H^T_{y,\varsigma} \end{bmatrix}$$

A partir da formulação do MEF pela energia potencial, e desprezando-se a contribuição do esforço de cisalhamento, obtém-se a matriz de rigidez do elemento DKT dada por :

$$\mathbf{K} = 2A \iint \mathbf{B}^T\,\mathbf{D}_b\,\mathbf{B}\,d\xi d\eta \qquad (3.43)$$

$$\mathbf{D}_b = \frac{E\,h^3}{12(1-\acute{\imath}^2)}\begin{bmatrix} 1 & \acute{\imath} & 0 \\ \acute{\imath} & 1 & 0 \\ 0 & 0 & \dfrac{1-\acute{\imath}}{2} \end{bmatrix}$$

De posse dos deslocamentos nodais, pode-se obter os esforços M_x, M_y e M_{xy} para o elemento dados por:

$$\mathbf{M}(x,y) = \begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} = \mathbf{D}_b \, \mathbf{B}(x,y) \, \mathbf{U} \tag{3.44}$$

O vetor de parâmetros nodais, bem como o vetor de forças nodais, para este elemento são dados por:

$$\mathbf{U}^T = \{ \quad w_1 \quad \theta_{x1} \quad \theta_{y1} \quad w_2 \quad \theta_{x2} \quad \theta_{y2} \quad w_3 \quad \theta_{x3} \quad \theta_{y3} \quad \} \tag{3.45}$$

$$\mathbf{F}_{ext}^T = \frac{q\,A}{3}\{1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\} \tag{3.46}$$

onde q representa o carregamento uniformemente distribuído na superfície do elemento e *A* a área do mesmo.

3.5 O MODELO ELASTOPLÁSTICO

Neste item apresenta-se o modelo de plasticidade adotado neste trabalho. São apresentadas as relações básicas necessárias, salientando-se apenas os conceitos fundamentais. Uma abordagem mais completa do assunto encontra-se em HILL (1950), CHEN (1988), OWEN (1980).

O principal objetivo da teoria da plasticidade é estudar o comportamento de estruturas quando submetidas a carregamentos que impõem à mesma esforços que excedem o limite do comportamento elástico. Neste caso procura-se estabelecer o tratamento matemático capaz de descrever corretamente esse comportamento. Essencialmente a plasticidade caracteriza-se pelo aparecimento de deformações permanentes mesmo após a retirada do carregamento – deformações irreversíveis. Pode-se visualizar melhor esse comportamento com a figura 3.3. Nesta figura, observa-se o aparecimento de deformações permanentes para uma situação de descarregamento.

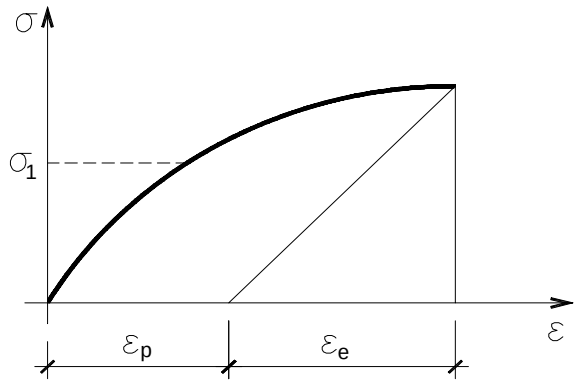


Figura 3.3. Representação do comportamento elastoplástico.

A elaboração do modelo matemático desse fenômeno está baseada nos seguintes princípios:

- Um critério que estabeleça um estado de tensão onde começa o fluxo plástico e o fim do comportamento elástico, ou *Critério de Plastificação*.

- Uma relação explícita entre as deformações elástica e plástica ou *Decomposição Aditiva das Deformações*.

- Uma relação entre tensão e deformação que possibilite o cálculo da deformação plástica após o limite da fase elástica ou *Lei de evolução da Deformações Plásticas*.

Neste item descreve-se de modo breve os princípios acima citados.

Critério de Plastificação

O critério de plastificação determina o estado de tensão no qual deformações plásticas tem início. A sua forma mais geral é dada por :

$$F(\sigma, \kappa) = f(\sigma) - Y(\kappa) \leq 0 \qquad (3.47)$$

Na equação (3.47) F representa o critério de plastificação, σ é o estado de tensões atual da estrutura, κ é um parâmetro relativo ao tipo de encruamento (isotrópico, cinemático e misto) denominado de parâmetro de endurecimento do material e $f(\sigma)$ é função do estado de tensão atual. A função $Y(\kappa)$ pode ser interpretada como um valor da tensão de escoamento

Supondo-se que as tensões principais de um estado de tensões qualquer coincidam com eixos de um sistema cartesiano de coordenadas, então o critério de plastificação F pode ser representado por uma superfície nesse espaço de tensões conforme figura 3.4. Os pontos no interior dessa superfície representam estados em regime elástico e os pontos fora dessa superfície são estados não permitidos, exigindo-se que o mesmo retorne para a superfície onde F=0.

Verifica-se na equação (3.47) que para situações onde $F < 0$ tem-se um caso de descarregamento elástico (ponto no interior da superfície de plastificação). Para $F = 0$ observa-se um estado de carregamento neutro e o estado de tensões deve permanecer na superfície de plastificação. Para $F > 0$ trata-se de estados não permitidos exigindo que o estado de tensões retorne para o nível onde $F = 0$. O retorno para a situação onde $F = 0$ pode ocorrer com endurecimento do material.

Neste caso pode-se ter modelos de comportamento elastoplástico com encruamento isotrópico, cinemático e misto. No caso isotrópico há um acréscimo no limite elástico. No caso cinemático tem-se uma translação da superfície de plastificação inicial e o caso misto conjuga os dois modelos anteriores. O comportamento isotrópico para os casos uniaxiais são representados por acréscimos nos valores de tensões a partir do quais o material perde seu comportamento linear.

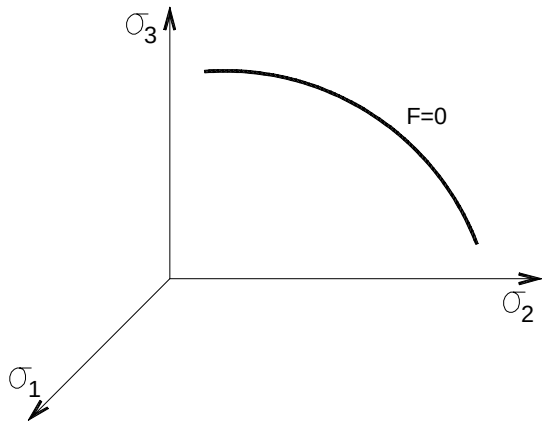


Figura 3.4 Superfície de plastificação

Neste trabalho adota-se o modelo isotrópico de encruamento, sendo que o desenvolvimento da superfície de plastificação é estabelecido a partir da função $Y(\kappa)$, que pode ser relacionada à deformação plástica através do parâmetro κ. Há duas hipótese para a definição deste parâmetro. A primeira considera que o mesmo é função do trabalho plástico acumulado, também conhecida como *work hardening*, ou seja:

$$d\kappa = dW_p = \boldsymbol{\sigma}^T d\boldsymbol{\varepsilon}_p \tag{3.48}$$

A Segunda hipótese estabelece que o parâmetro κ seja dado pela deformação plástica equivalente, denominado de *strain hardening*, ou seja:

$$d\hat{e} = d\bar{\varepsilon}^p = \sqrt{\frac{2}{3} d\boldsymbol{\varepsilon}^p \, d\boldsymbol{\varepsilon}^p} \tag{3.49}$$

No último caso o parâmetro κ é associado à norma dos incrementos de deformação plástica.

Neste trabalho adota-se como critério de plastificação o critério de Mises. Segundo von Mises o escoamento inicia quando a tensão octaédrica de cisalhamento atinge um valor crítico, ou seja:

$$\hat{o}_{oct} = \bar{k} = \sqrt{\frac{2}{3} J_2} \tag{3.50}$$

Ou,

$$\sqrt{\frac{2}{3} J_2} = \bar{k} \tag{3.51}$$

$$J_2 = \frac{1}{6}[(ó_1 - ó_2)^2 + (ó_1 - ó_3)^2 + (ó_2 - ó_3)^2]$$

Na equação anterior σ_1, σ_2 e σ_3 representam as tensões principais. A tensão de cisalhamento octaédrica é a tensão de cisalhamento que atua em um plano igualmente inclinado em relação aos eixos no espaço das tensões, chamado de plano octaédrico.

Pode-se visualizar geometricamente o critério de Mises na figura 3.5 onde a fase elástica se encontra no interior do cilindro no espaço das tensões.

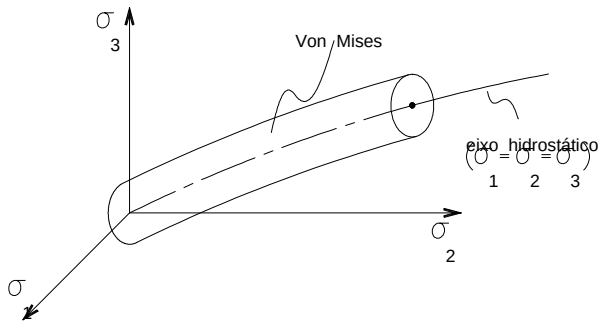


Figura 3.5 Representação geométrica do critério de von Mises.

Decomposição Aditiva das Deformações

Na situação em que se verifica o aparecimento de deformações irreversíveis nos materiais assume-se que o estado de deformação total é dado por uma relação aditiva, ou seja:

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}_e + \boldsymbol{\varepsilon}_p \tag{3.52}$$

Na equação (3.52) ε_e representa a parcela elástica da deformação total ε e ε_p a componente plástica (ou irreversível). Trata-se de uma relação fundamental na elaboração das relações constitutivas da teoria da plasticidade. A evolução da parcela elástica é dada pela lei de Hooke restando estabelecer o comportamento da deformação plástica.

Lei de Evolução das Deformações Plásticas

Além da relação que estabelece uma decomposição aditiva para a deformação total é importante dispor de uma relação explícita para a evolução das deformações. Em sistemas elásticos o trabalho realizado pelas cargas externas pode ser recuperado ao se retirar o carregamento aplicado. Pode-se então assumir o trabalho como sendo armazenado no corpo em forma de energia que é chamada de *energia de deformação*. Assim, pode-se expressar essa energia como o trabalho realizado, ou seja:

$$W(\varepsilon) = \int \sigma^T \, d\varepsilon \tag{3.53}$$

Essa energia de deformação corresponde à área abaixo da curva tensão-deformação. De modo semelhante, a área acima da dessa curva é chamada de *energia complementar* sendo expressa por :

$$\Omega(\sigma) = \int \varepsilon^T \, d\sigma \tag{3.54}$$

Com a equação anterior pode-se expressar a deformação em função da energia complementar, ou seja:

$$\text{å} = \frac{\partial \dot{U}(\acute{o})}{\partial \acute{o}} \tag{3.55}$$

De modo semelhante à expressão da deformação em função da energia complementar, von Mises em 1928 propôs que a deformação plástica fosse dada em

função de um potencial plástico denominado Q. Assim, a evolução das deformações plásticas é dada por:

$$d\mathring{a}_p = d\ddot{e}\,\frac{\partial Q}{\partial \acute{o}} \tag{3.56}$$

Na equação anterior o termo $d\lambda$ é denominado de multiplicador plástico. A partir da equação anterior se estabelece uma importante relação da teoria da plasticidade ao se considerar o potencial plástico como sendo a função de plastificação F, ou seja:

$$d\mathring{a}_p = d\ddot{e}\,\frac{\partial F}{\partial \mathbf{\acute{o}}} \tag{3.57}$$

Em plasticidade essa consideração é conhecida como *condição de normalidade* uma vez que o vetor $\partial F/\partial\sigma$ é normal à superfície de plastificação. Essa consideração é também conhecida como plasticidade associada e será adotada neste trabalho com o intuito de estabelecer o tratamento matemático da teoria da plasticidade.

Com os postulados anteriores pode-se finalmente apresentar o desenvolvimento matemático que descreve o comportamento dos materiais após o início do processo de plastificação. Do critério de plastificação dado pela equação (3.51) tem-se:

$$dF = \frac{\partial F}{\partial \mathbf{\acute{o}}}\,d\mathbf{\acute{o}} + \frac{\partial F}{\partial \hat{e}}\,d\hat{e} = 0 \tag{3.58}$$

$$\mathbf{F}_{,\acute{o}}^{T} = \frac{\partial F}{\partial \mathbf{\acute{o}}} = \left[\frac{\partial F}{\partial \acute{o}_x} \quad \frac{\partial F}{\partial \acute{o}_y} \quad \frac{\partial F}{\partial \acute{o}_z} \quad \frac{\partial F}{\partial \hat{o}_{yz}} \quad \frac{\partial F}{\partial \hat{o}_{zx}} \quad \frac{\partial F}{\partial \hat{o}_{xy}} \right]$$

Usando-se a decomposição aditiva das tensões e a lei de evolução das deformações plásticas tem-se:

$$d\boldsymbol{\varepsilon} = \mathbf{D}^{-1}\, d\boldsymbol{\sigma} + d\lambda \frac{\partial F}{\partial \boldsymbol{\sigma}} \tag{3.59}$$

Com o uso das equações (3.58) e (3.59) e do modo de endurecimento por trabalho obtém-se que:

$$d\lambda = \frac{\mathbf{F}_{,\sigma}^{T}\, \mathbf{D}\, \mathbf{F}_{,\sigma}}{H + \mathbf{F}_{,\sigma}^{T} \mathbf{D}\, \mathbf{F}_{,\sigma}^{T}} d\boldsymbol{\varepsilon} \tag{3.60}$$

A constante H, denominada de módulo plástico, pode ser determinada em um ensaio de tração uniaxial, sendo dada por:

$$H = \frac{d\sigma}{d\varepsilon_p} = \frac{E_T}{1 - E_T/E} \tag{3.61}$$

Na equação anterior E representa o módulo de Young do material, e E_T o módulo tangente elastoplástico.

Utilizando-se as equações (3.59) e (3.60), encontra-se a relação do material para o comportamento elastoplástico ou seja:

$$d\sigma = \mathbf{D}_{ep}\, d\varepsilon \tag{3.62}$$

A matriz D_{ep} é dada por:

$$\mathbf{D}_{ep} = \mathbf{D} - \frac{\mathbf{D}\mathbf{F}_{,\sigma}\, \mathbf{F}_{,\sigma}^{T}\, \mathbf{D}}{H + \mathbf{F}_{,\sigma}^{T} \mathbf{D}\, \mathbf{F}_{,\sigma}} \tag{3.63}$$

Com a equação (3.63) consegue-se estabelecer o modo de comportamento elastoplástico de materiais que obedeçam aos critérios já descritos. Essa relação

também possibilita o desenvolvimento do procedimento numérico em plasticidade, que será abordado no próximo item.

3.5.1 PROCEDIMENTO NUMÉRICO

O procedimento numérico para plasticidade com o Método dos Elementos Finitos é baseado na aplicação do Princípio dos Trabalhos Virtuais (PTV), considerando-se problemas não-lineares.

A aplicação do PTV em um corpo submetido a um carregamento qualquer, considerando-se os deslocamentos como uma aproximação polinomial, conduz à equação geral do MEF, dada por:

$$\mathbf{K}\,\mathbf{U} - \mathbf{F}_{ext} = \mathbf{0} \tag{3.64}$$

Na equação anterior **U** é o vetor de deslocamentos nodais a serem determinados, $\mathbf{F}_{ext}$ é o carregamento aplicado e **K** representa a matriz de rigidez da estrutura. Considerando **B** a matriz das derivadas das funções de forma tem-se

$$\mathbf{K} = \int \mathbf{B}^T\,\mathbf{D}\,\mathbf{B}\,dV \tag{3.65}$$

Essa relação é válida na elasticidade linear pois o material não perde suas características iniciais. Em problemas não-lineares, faz-se necessário adotar um procedimento incremental-iterativo na resolução do sistema. Consegue-se atingir esse objetivo dividindo-se o carregamento total em incrementos que serão aplicados na estrutura. Na aplicação incremental do carregamento, em cada iteração implementada gera-se um resíduo dado pela diferença entre o carregamento externo (valor do incremento de carga) e o vetor de forças internas verificado na estrutura. Assim, tem-se a expressão do procedimento:

$$\mathbf{\Psi} = \mathbf{F}_{ext} - \mathbf{F}_{int} \tag{3.66}$$

Na equação anterior $\boldsymbol{\Psi}$ representa o resíduo produzido em cada iteração do procedimento e que será reaplicado na estrutura até atingir a tolerância desejada, $\mathbf{F}_{ext}$ é a parcela do carregamento externo correspondendo ao incremento de carga aplicado, e $\mathbf{F}_{int}$ é o vetor de forças internas gerado pela aplicação do carregamento incremental. Assim, para cada iteração realizada em um incremento, calcula-se o resíduo após a resolução do sistema linear dado por (3.64). A solução final é encontrada quando o resíduo atinge a tolerância estabelecida, como pode ser visto na figura 3.6:

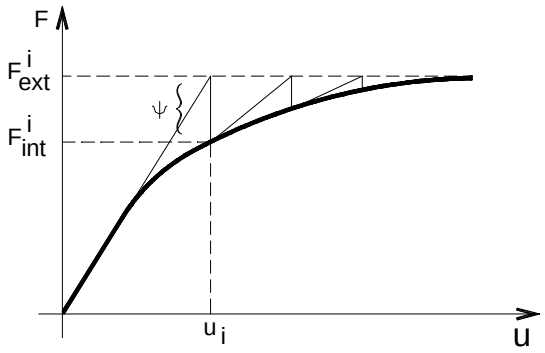


Figura 3.6 Representação gráfica do procedimento incremental-iterativo

Assim, no MEF faz-se necessário calcular a matriz de rigidez da estrutura em cada iteração realizada bem como o vetor de forças internas. Em plasticidade, após o carregamento superar o limite elástico da estrutura, a matriz de coeficientes elásticos **D** sofrerá modificação segundo a equação (3.63). Portanto, o cálculo da matriz de rigidez da estrutura deve ser atualizado em cada iteração, uma vez que a estrutura perdeu sua rigidez inicial. Essa nova matriz de rigidez utiliza a matriz de coeficientes elastoplásticos $\mathbf{D}_{ep}$. Desse modo tem-se:

$$\mathbf{K}_T = \int \mathbf{B}^T \mathbf{D}_{ep} \mathbf{B} \, dA \tag{3.67}$$

O vetor de forças internas é calculado após a resolução do sistema linear dado por (3.64). Com os deslocamentos dados tem-se:

$$\mathbf{F}_{int} = \int \mathbf{B}^T \boldsymbol{\sigma}^r \, dV \tag{3.68}$$

Na equação (3.68) σ^r é o vetor de tensões calculado em cada iteração a partir da relações elastoplásticas dadas por (3.62).O sobrescrito *r* denota a iteração em que se encontra o processo de plastificação. A integração anterior é realizada no domínio do elemento finito, para tanto recorre-se à integração numérica proposta por Gauss.

Assim, em cada incremento de carga, resolve-se o sistema de equações para cada iteração e posteriormente calcula-se o resíduo com a relação (3.66). Pode-se explicitar as seguintes etapas do procedimento incremental-iterativo na resolução do problema elastoplástico:

1ª. Calcular O Incremento De Carga

Calcula-se o valor do incremento $\Delta\mathbf{F}_{ext}$ a ser aplicado na estrutura. Os incrementos podem ter valores iguais ou diferentes. Esse critério depende do algoritmo desenvolvido.

2ª Resolver o Sistema De Equações.

Nesse passo resolve-se o sistema de equações dado por (3.64) e calcula-se os deslocamento nodais. O carregamento aplicado é dado no passo anterior. Portanto tem-se :

$$\mathbf{K}_T \, \Delta\mathbf{U} = \Delta\mathbf{F}_{ext} \tag{3.69}$$

Na equação anterior a matriz de rigidez $\mathbf{K}_T$ é dada pela equação (3.67), sendo necessária a atualização da mesma em cada iteração.

3ª . Calcular O Vetor De Forças Internas.

Após a resolução do sistema de equações do passo anterior, pode-se passar ao cálculo do vetor de forças internas. Para tanto, deve-se calcular o estado de tensões atual da estrutura representado por σ^r. Nesse caso deve-se adotar o seguinte procedimento:

PASSO A: Nesse passo calcula-se o incremento elástico de tensões. Para tanto, usa-se a relação:

$$\Delta\boldsymbol{\sigma}_e^r = \mathbf{D}\,\Delta\boldsymbol{\varepsilon}^r \tag{3.70}$$

Na equação anterior $\Delta\boldsymbol{\varepsilon}^r$ representa o incremento de deformação calculado com os deslocamentos obtidos na resolução do sistema de equações (3.64). Como se trata de uma casca deve-se computar as parcelas correspondentes aos efeitos de membrana e flexão. Assim o vetor de deformações será dado por:

$$\Delta\boldsymbol{\varepsilon}^r = \Delta\boldsymbol{\varepsilon}_c^r + \Delta\boldsymbol{\varepsilon}_p^r \tag{3.71}$$

Na equação anterior $\Delta\boldsymbol{\varepsilon}_c^r$ representa as deformações relativas ao efeito de membrana e $\Delta\boldsymbol{\varepsilon}_p^r$ as deformações relativas ao efeito de flexão. Para se calcular essas deformações emprega-se as seguintes relações:

$$\Delta\boldsymbol{\varepsilon}_c^r = \mathbf{B}_c\,\Delta\mathbf{U}_c \tag{3.72a}$$

$$\Delta\boldsymbol{\varepsilon}_p^r = z\,\mathbf{B}_p\,\Delta\mathbf{U}_p \tag{3.72b}$$

Na equação (3.72b) a coordenada *z* é dada em função do ponto de Gauss na espessura da casca. Para integrar as tensões ao longo da espessura também emprega-se este método.

PASSO B. De posse do estado atual de tensões calcula-se a tensão efetiva, dada pelo critério de Mises, ou seja

$$\bar{\sigma} = \sqrt{J_2} \tag{3.73}$$

PASSO C. Calcula-se o limite elástico atual da estrutura, dado por:

$$\acute{o}_Y = \acute{o}_Y^0 + H' \bar{\acute{a}}_p^{r-1} \tag{3.74}$$

Nesta equação, σ_Y^0 é o limite elástico inicial, H' o módulo elastoplástico e $\bar{e}_p^{r-1}$ é a deformação plástica da iteração *r-1*. A equação anterior é obtida a partir da relação (3.61), sendo válida para materiais com encruamento positivo.

PASSO D. Verifica se o ponto encontra-se em processo de plastificação. Para tanto, compara-se os valores da tensão efetiva $\bar{s}$ (*passo b*) com o limite elástico atual σ_Y (*passo c*). Se a tensão efetiva for menor que o limite elástico, tem-se um ponto fora do processo de plastificação. Se a tensão efetiva for maior que o limite elástico, tem-se um ponto em regime elastoplástico, caracterizando estados onde F > 0 (estados não permitidos). Necessita-se neste último caso voltar o estado de tensão para a superfície de plastificação. Para tanto emprega-se a relação (3.62). Assim ter-se-á:

$$\Delta\sigma = \mathbf{D}_{ep}\,\Delta\varepsilon = \mathbf{D}\,\Delta\varepsilon - d\lambda\,\mathbf{D}\,\mathbf{F}_{,\sigma}^{T}, \tag{3.75}$$

ou,

$$\sigma^r = \sigma^{r-1} + \Delta\sigma_e^r - d\lambda\,\mathbf{D}\,\mathbf{F}_{,\sigma}^{T} \tag{3.76}$$

Com o estado de tensões atualizado, pode-se calcular os esforços atuantes nos pontos de Gauss no domínio do elemento, ou seja:

$$\mathbf{N} = \begin{Bmatrix} N_x \\ N_y \\ N_{xy} \end{Bmatrix} = \int_{-h/2}^{h/2} \begin{Bmatrix} \acute{o}_x \\ \acute{o}_y \\ \hat{o}_{xy} \end{Bmatrix} dz = \int_{-h/2}^{h/2} \mathbf{\acute{o}}^r\, dz \tag{3.77a}$$

$$\mathbf{M} = \begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} = \int_{-h/2}^{h/2} z \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} dz = \int_{-h/2}^{h/2} z\, \boldsymbol{\sigma}^r \, dz \qquad (3.77b)$$

Nas relações (3.77) **N** representa o vetor de esforços normais, e **M** representa o vetor de momentos fletores.

PASSO E De posse do estado de tensões atual, após o retorno à superfície de plastificação, pode-se passar ao cálculo do vetor de forças internas dado por:

$$\mathbf{F}_{int,c} = \int \mathbf{B}_c^T \mathbf{N} \, dA \qquad (3.78\ a)$$

$$\mathbf{F}_{int,p} = \int \mathbf{B}_p^T \mathbf{M} \, dA \qquad (3.78b\ )$$

4ª Calcular O Resíduo.

Com o vetor de forças internas calculado no passo anterior, pode-se calcular o resíduo da iteração dada, ou seja :

$$\boldsymbol{\Psi} = \mathbf{F}_{ext} - \mathbf{F}_{int} \qquad (3.79)$$

5ª Verificar a Convergência.

Após o cálculo do resíduo verifica-se a convergência da iteração de acordo com a tolerância determinada. A convergência é verificada considerando-se a norma dos deslocamentos da iteração em relação à norma dos deslocamentos totais, ou seja:

$$\frac{\| \Delta \mathbf{U}^r \|}{\| \mathbf{U} \|} \text{x} 100 \leq \text{Tolerância} \qquad (3.80)$$

Na equação anterior $\| \Delta \mathbf{U}^{r} \|$ representa a norma dos deslocamentos na iteração *r* e $\| \mathbf{U} \|$ representa a norma dos deslocamentos totais. A mesma relação vale para verificação da convergência em força, ou seja:

$$\frac{\| \boldsymbol{\Phi}^{r} \|}{\| \mathbf{F}_{ext} \|} \text{x} 100 \leq \text{Tolerância} \tag{3.81}$$

Se a convergência atende à tolerância desejada, passa-se a um novo incremento de carga (1ª etapa). Caso contrário, volta-se à 2ª etapa, sendo o carregamento aplicado correspondente ao resíduo encontrado, até se conseguir a convergência.

A integração das tensões ao longo da espessura da casca é realizada utilizando o modelo estratificado, que considera a estrutura dividida em camadas. Trata-se de uma modelagem mais completa, uma vez que permite acompanhar o processo de plastificação no interior da estrutura. Essa formulação permite o estudo de materiais com diferentes composições como o concreto armado por exemplo, possibilitando inclusive a caracterização do comportamento da armadura da mesma. Para se realizar essa modelagem, integram-se as tensões e a matriz de coeficientes elastoplásticos ($\mathbf{D}_{ep}$), utilizando o método de integração numérica de Gauss.

Assim, ao se calcular a matriz de rigidez elastoplástica tangente (3.63), deve-se considerar a variação de $\mathbf{D}_{ep}$ ao longo da espessura. No caso do elemento finito de casca plano, as matrizes de coeficientes elastoplásticos para placa, chapa e acoplamento serão dadas por:

$$\mathbf{D}_{ep,p} = \int_{-h/2}^{h/2} z^{2}\, \mathbf{D}_{ep}\, dz = \sum_{i}^{Ng} \frac{h^{3}}{8} \hat{\imath}_{i}^{2}\, \mathbf{D}_{ep,i}\, \grave{u}_{i} \tag{3.82a}$$

$$\mathbf{D}_{ep,c} = \int_{-h/2}^{h/2} \mathbf{D}_{ep}\, dz = \sum_{i}^{Ng} \frac{h}{2}\, \mathbf{D}_{ep,i}\, \grave{u}_{i} \tag{3.82b}$$

$$\mathbf{D}_{ep,cp} = \int_{-h/2}^{h/2} z\,\mathbf{D}_{ep}\,dz = \sum_{i}^{Ng} \frac{h^2}{4} \hat{\imath}_i\ \mathbf{D}_{ep,i}\ \grave{u}_i \tag{3.82c}$$

Nas equações (3.82) N_g representa o número de pontos de Gauss ao longo da espessura da casca, ξ_i é o ponto de Gauss ao longo da espessura da casca e ω_i é o respectivo peso de Gauss para cada camada. O termo $\mathbf{D}_{ep,i}$ é a matriz dos coeficientes elastoplástico, dada em (3.63) para cada ponto de Gauss na espessura. Neste caso, o estado de tensões deve ser atualizado de acordo com o *passo D da terceira etapa* do procedimento numérico. Com as equações (3.82) calcula-se a matriz de rigidez tangente elastoplástica para os elemento planos, placa, chapa e acoplamento:

$$\mathbf{K}_{T,p} = \int \mathbf{B}_p{}^T\,\mathbf{D}_{ep,p}\,\mathbf{B}_p\,dA \tag{3.83a}$$

$$\mathbf{K}_{T,c} = \int \mathbf{B}_c{}^T\,\mathbf{D}_{ep,c}\,\mathbf{B}_c\,dA \tag{3.83b}$$

$$\mathbf{K}_{T,cp} = \int \mathbf{B}_c{}^T\,\mathbf{D}_{ep,cp}\,\mathbf{B}_p\,dA \tag{3.83c}$$

As integrais apresentadas nas relações (3.83) também são calculadas numericamente pelo método de Gauss para domínios triangulares.

No caso dos esforços, a integração ao longo da espessura também utiliza o método numérico já citado, podendo ser expressa como segue:

$$\mathbf{N} = \int_{-h/2}^{h/2} \boldsymbol{\acute{o}}^r\,dz = \sum_{i}^{Ng} \frac{h}{2} \boldsymbol{\acute{o}}_i^r\ w_i \tag{3.84 a}$$

$$\mathbf{M} = \int_{-h/2}^{h/2} z\,\boldsymbol{\acute{o}}^r\,dz = \sum_{i}^{Ng} \frac{h}{2} \hat{\imath}_i\ \boldsymbol{\acute{o}}_i^r\ w_i \tag{3.84 b}$$

Apresentou-se neste capítulo a formulação do MEF para análise elastoplástica de cascas usada neste trabalho. No próximo capítulo apresenta-se a paralelização

desta formulação. Salienta-se ainda que este texto tem o intuito de apresentar o modelo utilizado. Para maiores esclarecimentos recomenda-se a bibliografia já citada, que é referência para este trabalho.

CAPÍTULO IV

ANÁLISE NÃO-LINEAR EM PARALELO

4.1 INTRODUÇÃO

Neste capítulo desenvolve-se o paralelismo para o modelo elastoplástico apresentado no capítulo III. Como já visto no estudo da plasticidade, utilizou-se como ferramenta de cálculo o Método dos Elementos Finitos. Portanto, para se paralelizar totalmente este programa deve-se desenvolver o paralelismo junto às etapas características do MEF tais como cálculo da matriz de rigidez, resolução do sistema de equações lineares e cálculo dos esforços. Desse modo, apresenta-se neste capítulo algoritmos em paralelo para as etapas do MEF, bem como todo procedimento adotado para se paralelizar o programa de análise não-linear.

Como já salientado o paralelismo pode utilizar máquinas com memória global e local. A utilização de um ou outro equipamento interfere no modo de programação, tornando-se uma necessidade identificar o equipamento utilizado. Para máquinas com memória global, o paralelismo é implementado em forma de comandos no programa fonte. No caso de máquinas com memória local, o paralelismo é implementado com auxílio de bibliotecas de troca de mensagens, que possibilitam a comunicação entre os processos, uma vez que os mesmos executam suas tarefas de modo independente. Neste trabalho utilizou-se o IBM SP2, que é uma máquina com memória local, com possibilidade de acréscimo de até 128 nós de processamento. Cada nó refere-se a uma unidade local constituída de memória e processador. A biblioteca de troca de mensagens utilizada foi o PVM (Parallel Virtual Machine).

4.2 O PARALELISMO NO MÉTODO DOS ELEMENTOS FINITOS.

O Método dos Elementos Finitos (MEF) é um método numérico para se resolver problemas cuja solução analítica seja difícil ou mesmo inexistente. Ele baseia-se na decomposição de um domínio qualquer em elementos cujos campos de deslocamento e deformação sejam aproximados por uma função polinomial. À medida que se aumenta o número de elementos desse domínio a solução numérica tende para a solução exata. De um modo geral o MEF pode ser representado pelas etapas da figura 4.1.

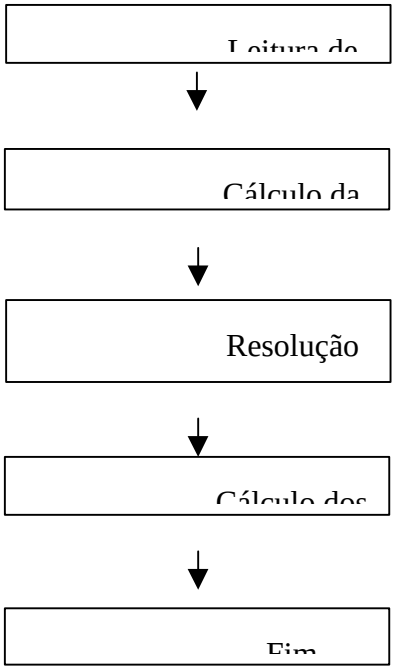


Figura 4.1 Etapas do Método dos Elementos Finitos.

Desse modo, quando se procurar paralelizar o MEF deve-se fazê-lo dentro das etapas dadas, ou seja, procura-se paralelizar as rotinas que representem estas etapas ou mesmo todo o programa.

Como já citado no Capítulo II, quando se pretende desenvolver um programa em paralelo existem dois caminhos a serem percorridos. Pode-se em primeira

instância paralelizar algoritmos seqüenciais em uso, ou desenvolver algoritmos em paralelo para a máquina disponível. Ambos procedimentos podem ser adotados para se implementar o paralelismo dentro do MEF. Deve-se contudo, estar atento a possíveis erros na elaboração do programa. Como obstáculos mais freqüentes na adaptação de algoritmos seqüenciais cita-se o problema da dependência de dados. Neste capítulo, exemplifica-se o problema da dependência de dados no procedimento usual de cálculo da matriz de rigidez. Apresenta-se também o procedimento paralelo para esse cálculo, para a rotina de resolução do sistema de equações e, em conjunto com a implementação em paralelo da análise não linear, para o cálculo dos esforços.

4.3 CÁLCULO DA MATRIZ DE RIGIDEZ DA ESTRUTURA.

O paralelismo é um modo eficiente e de baixo custo de reduzir o tempo de processamento em problemas que exigem computação de alto desempenho. Para se extrair o máximo proveito dos recursos do paralelismo, todas as etapas de um programa que são passíveis de serem paralelizadas devem ser submetidas a uma análise com o intuito de ajustá-las a esse tipo de arquitetura de computadores. Por esse motivo procura-se paralelizar o cálculo da matriz de rigidez da estrutura, uma vez que em conjunto com a resolução do sistema de equações, é uma das etapas que exigem maior esforço computacional.

Quando se pretende implementar um algoritmo seqüencial para o modo paralelo, podem surgir vários obstáculos. Um problema comum é a dependência de dados. Com o intuito de exemplificar o problema de adaptação de algoritmos seqüenciais em processamento paralelo, cita-se o conhecido algoritmo para o cálculo da matriz de rigidez da estrutura. Neste item mostra-se como a dependência de dados pode interferir no resultado final do processamento além de apresentar um algoritmo adequado ao processamento paralelo.

4.3.1 O CÁLCULO DA MATRIZ DE RIGIDEZ - PROCEDIMENTO USUAL.

A adaptação de algoritmos seqüenciais para serem usados no paralelismo nem sempre é feita de modo elementar. Pode-se inclusive incorrer em uma inviabilidade

de aplicação do mesmo para o paralelismo. Para o MEF pode-se citar o procedimento usual para o cálculo da matriz de rigidez. Esse procedimento pode ser identificado pelo algoritmo 4.1:

ALGORITMO 4.1 CÁLCULO DA MATRIZ DE RIGIDEZ EM MODO SEQUÊNCIAL.
ALGORITMO USUAL

NUMEL..... NÚMERO DE ELEMENTOS DA ESTRUTURA.
NNOS..... NÚMERO DE NÓS DO ELEMENTO.
NVARN..... NÚMERO DE VARIÁVEIS POR NÓ.
KE..... ARMAZENA A MATRIZ DE RIGIDEZ DA ESTRUTURA.
KMIN..... MATRIZ DE RIGIDEZ DO ELEMENTO.

```
1   PARA N = 1 ATÉ NUMEL FAÇA
2     PARA I = 1 ATÉ NNOS FAÇA
3       L = NP (N,I)
4       NLG = NVARN*(L-1) +1
5       PARA J =1 ATÉ NNOS FAÇA
6         K = NP(N,J)
7         NCG = NVARN*(K-1)+1
8         PARA II=1 ATÉ NVARN FAÇA
9           NUMLL=NVARN*(I-1)+II
10          NUMLG=NLG-1+II
11          PARA JJ=1 ATÉ NVARN FAÇA
12            NUMCL=NVARN*(J-1)+JJ
13            NUMCG=NCG-1+JJ
14            NCAUX=NUMCG-(NUMLG-1)
15            SE (NCAUX >0 ) ENTÃO
16              KE(NUMLG,NCAUX)=KE(NUMLG,NCAUX)+
17                                  KMIN(NUMLL,NUMCL)
18            FIM SE
19          FIM FAÇA
20        FIM FAÇA
21      FIM FAÇA
22  FIM FAÇA
```

Esse algoritmo tem sido usado com sucesso na elaboração de programas seqüenciais para o MEF. Percorrendo todos os elementos que compõem a estrutura (*loop* mais externo – linha 1), ele aloca e adiciona, quando necessário, os coeficientes de rigidez dos elementos para suas respectivas posições na matriz de rigidez global.

Contudo para o caso de elaboração de um programa paralelo, deve-se estar atento ao problema de atualização simultânea da matriz de rigidez.

Para exemplificar o problema da dependência de dados no processo usual de cálculo da matriz de rigidez, seja o exemplo de uma estrutura discretizada conforme a figura 4.2. Neste caso, supõe-se elementos com dois parâmetros por nó cuja matriz de rigidez será calculada conforme o algoritmo 4.1.

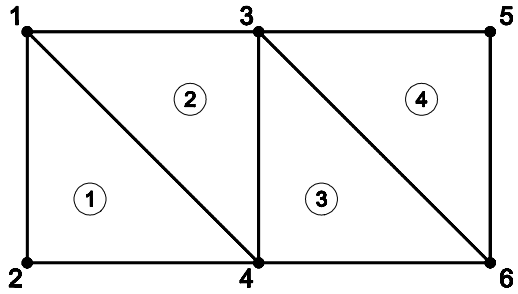


Figura 4.2. Estrutura discretizada com quatro elementos finitos.

Ao se efetuar esse cálculo, o coeficiente de rigidez global KE (8,1), correspondente ao segundo grau de liberdade do nó 4, será resultado da adição dos coeficientes locais nas posições (6,6) do elemento 1 e (4,4) dos elementos 2 e 3.

No modo paralelo suponha-se a utilização de dois processos nesse cálculo. Dispondo para o processo 1 os elementos 1 e 2 e para o processo 2 os elementos 3 e 4, a adição processada na montagem seqüencial visivelmente não se efetuará no processo paralelo. Neste caso, os elementos correspondentes estarão alocados em processos distintos.

Vários autores apresentaram trabalhos com o intuito de resolver esse problema, como pode ser visto na revisão bibliográfica realizada. Entretanto, a baixa eficiência das soluções propostas motiva a continuidade da pesquisa sobre esse tema.

4.3.2 O CÁLCULO PARALELO DA MATRIZ DE RIGIDEZ DA ESTRUTURA.

Neste item apresenta-se o cálculo da matriz de rigidez da estrutura para máquinas paralelas com memória distribuída. Como já foi visto, o procedimento usual de alocar os coeficientes de rigidez local para suas respectivas posições na matriz da estrutura através de um *loop* sobre os elementos finitos não é passível de ser usado diretamente no paralelismo, uma vez que ele incorre no problema da dependência de dados.

Vários trabalhos têm sido apresentados com o intuito de se resolver esse problema – LAW (1986), ADELI e KAMAL (1992), CHIEN e SUN (1989). Esses trabalhos envolvem desde o cálculo paralelo da matriz de rigidez da estrutura, até o processamento concorrente de toda estrutura com troca de informações entre os processos. REZENDE (1995) apresenta um algoritmo alternativo de cálculo da matriz de rigidez que aloca os coeficientes de rigidez local para a estrutura através de um *loop* sobre os *pontos nodais* da estrutura (abordagem nodal). Nesse algoritmo a matriz da estrutura é calculada *linha por linha*, ou seja, para cada ponto nodal da malha calcula-se os coeficientes de rigidez de todos os graus de liberdade desse ponto. Esse algoritmo é apresentado em seguida:

ALGORITMO 4.2 CÁLCULO DA MATRIZ DE RIGIDEZ
PROCEDIMENTO ALTERNATIVO COM ABORDAGEM NODAL

NUMNP..... NÚMERO DE PONTOS NODAIS DA ESTRUTURA.
NUMEL..... NÚMERO DE ELEMENTOS DA ESTRUTURA.
NVARN..... NÚMERO DE VARIÁVEIS POR NÓ DO ELEMENTO.
NNOS..... NÚMERO DE NÓS DO ELEMENTO.
NP[I,J]..... INCIDÊNCIA DO ELEMENTO I.
NCONC(I)..... NÚMERO DE ELEMENTOS LIGADOS AO PONTO I.
ELEM_CONC[K,I]..... NÚMERO DO I-ÉSIMO ELEMENTO LIGADO AO PONTO K.
NO_CORRESP[K,I]..... NUMERAÇÃO LOCAL DO PONTO K PARA O I-ÉSIMO ELEMENTO A ELE LIGADO.

```
1   ESTRUTURA DE DADOS AUXILIAR.
2    PARA I=1 ATÉ NUMEL FAÇA
3      PARA J=1 ATÉ NNOS FAÇA
4        P = NP(I,J)
5        NCONC(P) = NCONC(P)+1
6        ELEM_CONC(P,NCONC(P)) = I
7        NO_CORRESP(P,NCONC(P)) = J
8      FIM FAÇA
9    FIM FAÇA
10
11  CÁLCULO DA MATRIZ DA ESTRUTURA
12   NPAR=NVARN
13   PARA I=1 ATÉ NUMNP FAÇA
14     PARA J=1 ATÉ NCONC(I) FAÇA
15       ELEM=ELEM_CONC(I,J)
16       PON=NO_CORRESP(I,J)
17       MONTE A MATRIZ DO ELEMENTO MAT_L
18       LINHA_BASE=NPAR*I-(NPAR-1)
19       PARA K=1 ATÉ NNOS FAÇA
20         PAUX=NP(ELEM,K)
21         COLUNA=NPAR*PAUX-(NPAR-1)
22         PARA M1=1 ATÉ NPAR FAÇA
23           LINHA_LOCAL=NPAR*(PON-1)+M1
24           LINHA_GLOBAL=(LINHA_BASE-1)+M1
25           PARA N=1 ATÉ NPAR FAÇA
26             COLUNA_LOCAL=NPAR*(K-1)+N
27             COLUNA_GLOBAL=(COLUNA-1)+N
28             DIF=COLUNA_GLOBAL-LINHA_GLOBAL+1
29             SE DIF. MENOR QUE. 0 ENTÃO
30             MAT_G(LINHA_GLOBAL,DIF)=MAT_G(LINHA_GLOBAL,DIF)+
31                                     MAT_L(LINHA_LOCAL,COLUNA_LOCAL)
32             FIM SE
33           FIM FAÇA
34         FIM FAÇA
```

```
35                    FIM FAÇA
36                FIM FAÇA
37            FIM FAÇA
```

Para melhor compreensão deste algoritmo, pode-se recorrer ao exemplo da figura 4.2. Nesse caso para o ponto nodal 4, o algoritmo calculará as linhas 7 e 8 da matriz de rigidez global, sendo que os elementos finitos que participarão deste processo serão os elementos 1, 2 e 3, ou seja, os elementos que pertencem à vizinhança do ponto nodal. Na figura 4.3 ilustra-se este procedimento.

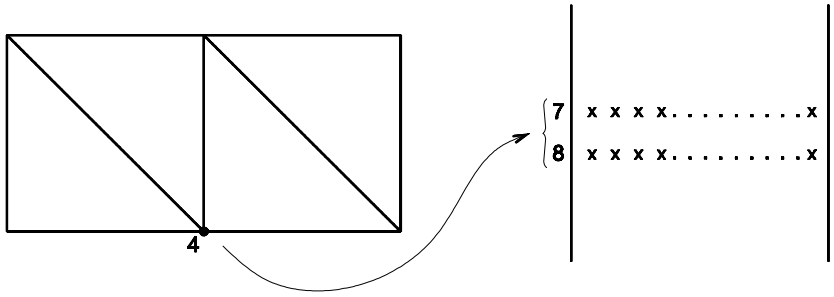


Figura 4.3 Cálculo da matriz de rigidez com abordagem nodal

A identificação dos elementos finitos que concorrem a um determinado ponto nodal é efetuada com a estrutura de dados auxiliar, correspondentes às linhas 1 a 9 do algoritmo 4.2.

Nesse caso não ocorre o problema da dependência de dados, ou seja, o cálculo de uma linha da matriz de rigidez é independente de outra linha desta matriz. Por esse motivo o algoritmo proposto pode ser usado diretamente no paralelismo. Assim, para uma máquina com memória distribuída, supondo-se a existência de dois processos em execução, pode-se, sem nenhuma dificuldade, alocar para o processo P1 os pontos nodais 1 a 3 e para o processo P2 os pontos 4 a 6. No algoritmo 4.2 essa operação se resumiria em mudar os limites do *loop* na linha 13. Graficamente pode-se representar esse procedimento através da figura 4.3:

Esse algoritmo de cálculo da matriz de rigidez com abordagem nodal foi utilizado neste trabalho em conjunto com o procedimento em paralelo para resolução do sistema de equações, que será abordado no próximo item.

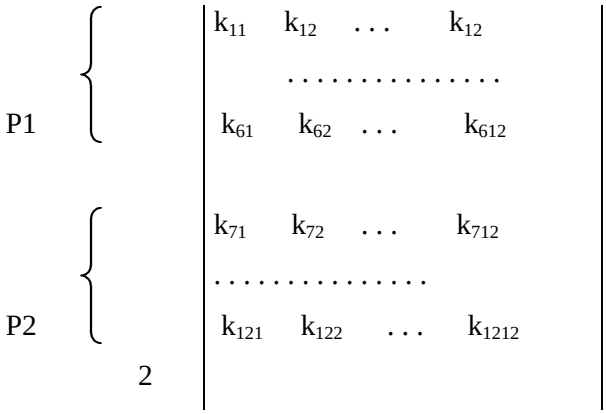


Figura 4.4 Cálculo da matriz de rigidez com dois processos

4.4 RESOLUÇÃO DO SISTEMA DE EQUAÇÕES.

A resolução do sistema de equações é a etapa do MEF que exige maior esforço computacional. Por esse motivo, a pesquisa por algoritmos que conjuguem confiabilidade e rapidez se faz necessária.

Os métodos de resolução de sistemas lineares podem ser classificados em diretos e iterativos. Os métodos diretos apresentam a vantagem de se realizar um número conhecido de operações. Entretanto, quando se resolve sistemas de grandes dimensões, pode-se ter erros de arredondamento.

Dentre os métodos diretos destaca-se o método de eliminação de Gauss, com maior utilização, além da decomposição incompleta de Choleski. A implementação de ambos em processamento paralelo apresenta como obstáculo a dependência de dados, já comentada no procedimento de cálculo da matriz de rigidez.

Os métodos iterativos apresentam como principal aspecto positivo a garantia de convergência para o resultado esperado. Além de garantir a convergência para o resultado esperado, pode ser implementado no paralelismo com maior facilidade.

Como exemplo de métodos iterativos cita-se Gauss-Seidel , Gauss-Jordam, Gradientes Conjugados.

Devido a suas características de convergência e confiabilidade, o método dos Gradientes Conjugados tem merecido atenção especial dos pesquisadores em análise numérica, como pode-se constatar na bibliografia apresentada. Além dessas características, a sua formulação possibilita fácil adaptação à computação paralela, tanto compartilhada quanto distribuída. Por esses motivos utilizou-se este método no trabalho.

4.4.1 O MÉTODO DOS GRADIENTES CONJUGADOS

O método dos Gradientes Conjugados baseia-se no Método de Maior Decréscimo, que é um método para resolução de problemas de minimização de funcionais. Seja então um sistema linear dado por:

$$\mathbf{A}\,\mathbf{x} = \mathbf{b} \tag{4.1}$$

Nesta equação **A** representa a matriz dos coeficientes do sistema, **x** é o vetor de incógnitas e **b** é o vetor de parâmetros independentes. Considerando-se $\mathbf{x}_0$ a solução do sistema anterior o erro será dado por:

$$\mathbf{y} = \mathbf{x}_0 - \mathbf{x} \tag{4.2}$$

O resíduo é dado por:

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\,\mathbf{x} \tag{4.3}$$

Assim, em cada passo desse método um vetor $\mathbf{p}_j$ é escolhido e o novo vetor que aproxima a solução do sistema é dado por:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\,\mathbf{p}_j \tag{4.4}$$

Também tem-se que:

$$\mathbf{y}_{k+1} = \mathbf{y}_k - \alpha_k \mathbf{p}_k \tag{4.5}$$

Assim, a norma do erro será dada por:

$$\| \mathbf{y}_{k+1} \|^2 = (\mathbf{A}\, \mathbf{y}_{k+1}, \mathbf{y}_{k+1})$$
$$= (\mathbf{A} [\mathbf{y}_k - \alpha_k \mathbf{p}_k], [\mathbf{y}_k - \alpha_k \mathbf{p}_k]) \tag{4.6}$$

Na equação anterior, o termo entre parêntesis representa o produto interno entre os respectivos vetores. Portanto da equação anterior tem-se:

$$\| \mathbf{y}_{k+1} \|^2 = (\mathbf{A}\, \mathbf{y}_k, \mathbf{y}_k) - 2\, \alpha_k (\mathbf{A}\, \mathbf{y}_k, \mathbf{y}_k) + \alpha_k^{\,2} (\mathbf{A}\, \mathbf{y}_k, \mathbf{y}_k) \tag{4.7}$$

Nesta última equação obtém-se o mínimo para:

$$á = \frac{(\mathbf{A}\mathbf{y}_k, \mathbf{p}_k)}{(\mathbf{A}\mathbf{p}_k, \mathbf{p}_k)} \tag{4.8}$$

O parâmetro anterior pode ser rescrito como:

$$á = \frac{\mathbf{r}_k^{\mathrm{T}}\, \mathbf{r}_k}{\mathbf{p}_k^{\mathrm{T}}\, \mathbf{A}\, \mathbf{p}_k} \tag{4.9}$$

Com as relações anteriores pode-se apresentar o algoritmo do método dos Gradientes Conjugados:

ALGORITMO 4.3 MÉTODO DOS GRADIENTES CONJUGADOS

Para k=0

1. Escolha $\mathbf{x}_0$
2. Calcule $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\, \mathbf{x}_0$

3. Faça $\mathbf{p}_0 = - \mathbf{r}_0$

4. **SE** $\| r \| < \varepsilon \Rightarrow$ **PASSO 12 , SENÃO** :

5. $\alpha_k = \dfrac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}$

6. $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$

7. $\mathbf{r}_{k+1} = \mathbf{r} - \alpha_k \mathbf{A} \mathbf{p}_k$

8. $\beta_k = \dfrac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$

9. $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_{k+1} \mathbf{p}_k$

10. $k = k + 1$

11. **FIM SE**

12. **FIM**

A formulação do método aqui apresentada está feita de modo claramente resumido. Para maiores esclarecimentos recomenda-se CIMMERMAN (1996), GOLUB (1984).

A partir do algoritmo do método dos Gradientes Conjugados pode-se desenvolver a implementação do mesmo em paralelo, o que será mostrado no próximo item.

4.4.2 IMPLEMENTAÇÃO EM PARALELO DO MÉTODO DOS GRADIENTES CONJUGADOS

Com base no algoritmo 4.3 apresenta-se a implementação em paralelo do CG. Essa implementação foi desenvolvida a partir do pacote de rotinas PIM (Parallel Iterative Methods), de autoria de CUNHA & HOPKINS (1998), o qual permite a resolução tanto de sistemas simétricos quanto não simétricos. Além do método dos Gradientes Conjugados, este pacote dispõe também de outras opções como Gradientes Conjugados Quadrados (QGC), Bi-Gradientes Conjugados (Bi-CG), Bi-Gradiente Conjugado Estabilizado (Bi-GCSTAB) e Resíduo Mínimo Generalizado. Neste conjunto de rotinas o armazenamento da matriz, o produto matriz-vetor e os produtos internos são tarefas a cargo do usuário.

Apresenta-se em seguida comentários sobre o procedimento em paralelo adotado. Em cada passo do algoritmo 4.3, cada processo em execução efetua sua parcela de computação. Com intuito ilustrativo e para melhor compreensão, supõe-se a execução com três processos concorrentes.

Passo 1 Escolher $\mathbf{x}_0$

Nesse passo a escolha do vetor de aproximação inicial do CG é feita em cada processo em execução.

Passo 2 Calcular $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$

Esse cálculo é melhor compreendido com a ilustração da figura a seguir. Nessa operação o produto $\mathbf{A}\,\mathbf{x}_0$ é realizado simultaneamente nos três processos em execução, sendo que cada processo efetua sua parcela de cálculo. A matriz **A** se encontra armazenada nestes processos bem como o vetor $\mathbf{x}_0$. Ao se efetuar a multiplicação, cada processo efetua sua respectiva parcela do produto.

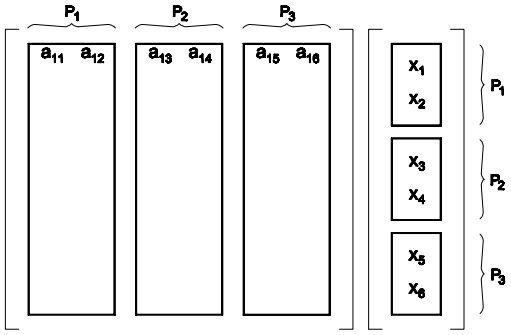


Figura 4.5 Cálculo do produto $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$.

Na figura (4.5), em cada processo está armazenado a respectiva parcela da matriz (calculada na etapa anterior do MEF). Assim, cada processo efetua de modo independente seu respectivo cálculo e armazena em uma variável auxiliar S, ou seja:

Processo P_1 $\Rightarrow$ $S_1 \leftarrow a_{11}\, x_1 + a_{12}\, x_2$

Processo P_2 $\Rightarrow$ $S_2 \leftarrow a_{13}\, x_3 + a_{14}\, x_4$

Processo P_3 $\Rightarrow$ $S_3 \leftarrow a_{15}\, x_5 + a_{16}\, x_6$

Figura 4.6 Armazenamento do produto matriz-vetor em variável auxiliar.

O resultado da multiplicação da *primeira linha* da matriz **A** pelo vetor $\mathbf{x}_0$ deve ser armazenado *somente no processo P_1*. Assim, após efetuarem os seus produtos, os processos P_2 e P_3 devem enviar o resultado para o processo P_1 que recebe e totaliza a soma. No *processo P_1* armazena-se também o resultado para a *segunda linha* da matriz **A**. Os resultados das *3^a e 4^a linhas* serão armazenados no *processo P_2* e das *5^a e 6^a linhas* armazenados no *processo P_3*. Em cada processo, armazena-se o resultado da multiplicação $\mathbf{A}\,\mathbf{x}_0$ em um vetor auxiliar **v**.

Após os respectivos cálculos e suas acumulações, cada processo está apto a finalizar esta etapa, ou seja, realizar a operação $\mathbf{b} - \mathbf{A}\,\mathbf{x}_0$. Note-se que em cada processo está armazenada somente sua respectiva parcela, tanto do vetor **b** quanto do resultado $\mathbf{A}\,\mathbf{x}_0$, isto é, o processo P_1 armazena os valores b_1 e b_2, e assim sucessivamente. Portanto, o resultado final será:

Processo	P_1	P_2	P_3
Resultado	$r_1 \leftarrow b_1 - v_1$ $r_2 \leftarrow b_1 - v_1$	$r_3 \leftarrow b_3 - v_3$ $r_4 \leftarrow b_4 - v_4$	$r_5 \leftarrow b_5 - v_5$ $r_6 \leftarrow b_6 - v_6$

Figura 4.7 Acumulação da adição entre dois vetores.

Com os resultados para o vetor resíduo da primeira iteração, pode-se passar ao passo seguinte.

Passo 3 Fazer $\mathbf{p} = -\,\mathbf{r}_0$

Cada processo assume os respectivos valores do resíduo e armazena no vetor **p**. Assim:

Processo	P_1	P_2	P_3
Resultado	$p_1 \leftarrow -r_1$ $p_2 \leftarrow -r_1$	$p_3 \leftarrow -r_3$ $p_4 \leftarrow -r_4$	$p_5 \leftarrow -r_5$ $p_6 \leftarrow -r_6$

Figura 4.8 Armazenamento do resíduo em vetor auxiliar.

Passo 4 Verificar **SE** $\| \mathbf{r} \| < \varepsilon$

Na realização deste passo, necessita-se calcular a norma do resíduo. A norma do vetor **r** é dada pela raiz quadrada do produto interno, ou seja:

$$\| \mathbf{r} \| = [(\mathbf{r} , \mathbf{r})]^{1/2} \qquad (4.10)$$

Neste caso, cada processo realiza o produto interno parcial e envia o seu resultado para o processo Pai. Após a totalização o processo pais e envia o resultado de volta aos outros processos de modo que *todos* possam verificar o critério de convergência. Na figura a seguir representa-se essa operação:

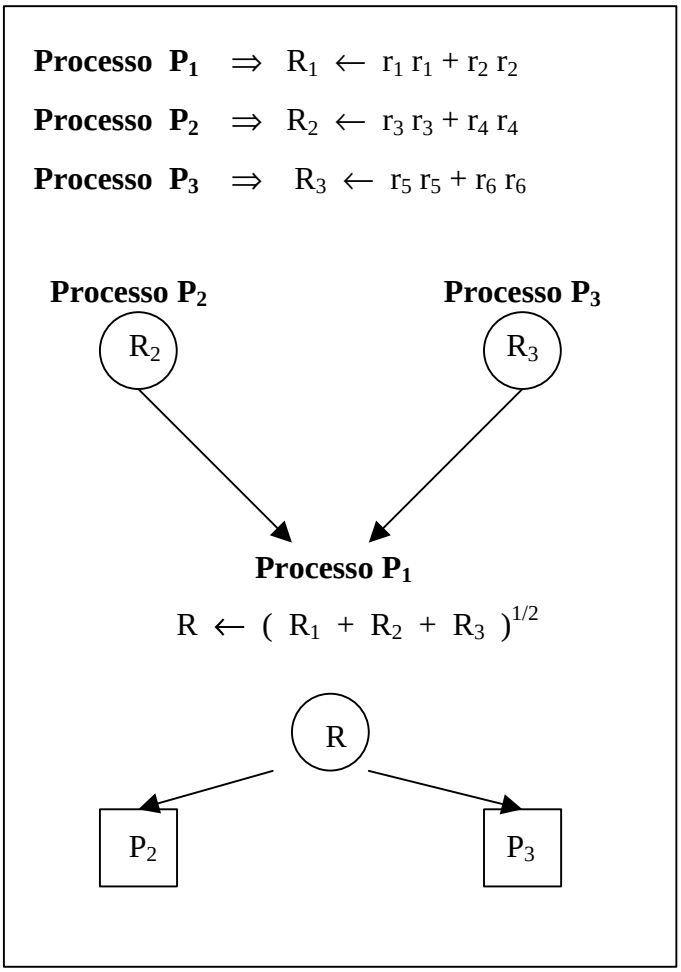


Figura 4.9 Procedimento para cálculo do produto escalar

Passo 5 Calcular parâmetro α_k

O cálculo do parâmetro α_k é dado pela relação:

$$\acute{a}_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A}\, \mathbf{p}_k} \qquad (4.11)$$

Neste caso, necessita-se calcular o produto $\mathbf{r}_k^T \mathbf{r}_k$, que pode ser compreendido com os comentários sobre o passo 4. A única observação adicional nesse caso é que deve-se armazenar esse produto em uma variável auxiliar (ρ) para que o mesmo possa ser usado posteriormente.

Na parcela dada no denominador, necessita-se fazer a multiplicação da matriz **A** pelo vetor $\mathbf{p}_k$. Essa multiplicação é processada à semelhança do passo 2, e também deve-se armazenar esse resultado em um vetor auxiliar ($\boldsymbol{\omega} = \mathbf{A}\,\mathbf{p}_k$). A execução em paralelo para $\mathbf{p}_k^T\,\boldsymbol{\omega}$ também já foi ilustrada no passo anterior.

Passo 6 Atualizar vetor **x**

Na atualização do vetor **x** utiliza-se a relação:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\ \mathbf{p}_k \tag{4.12}$$

Neste caso o parâmetro α_k já calculado no passo anterior, está disponível em *todos* os processos em execução. Uma vez que os vetores **x** e **p** estão parcialmente armazenados nestes processos, basta realizar a soma vetorial dada em (4.12), fazendo-se um laço apenas para a dimensão armazenada localmente. Na figura a seguir mostra-se esquematicamente essa operação:

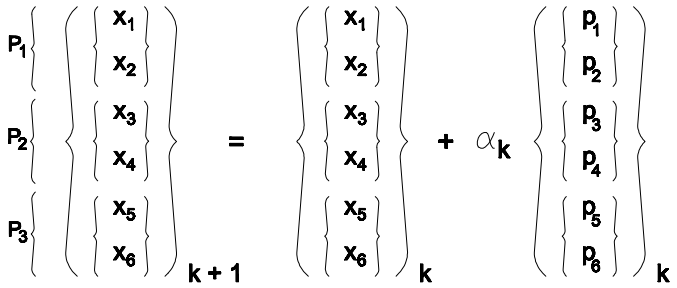


Figura 4.10 Atualização do vetor **x**.

Passo 7 Atualizar vetor **r**

Para se atualizar o vetor **r** usa-se a relação:

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k\,\mathbf{A}\,\mathbf{p}_k \tag{4.13}$$

$$= \mathbf{r}_k - \alpha_k\,\boldsymbol{\omega} \tag{4.14}$$

A operação vetorial neste caso é idêntica ao passo 6, sendo que o produto $\mathbf{A}\,\mathbf{p}_k$ foi armazenado na variável ω no passo 5.

Passo 8 Calcular β

O cálculo da variável β é dado pela relação:

$$\hat{a}_k = \frac{\mathbf{r}_{k+1}^T\ \mathbf{r}_{k+1}}{\mathbf{r}_k^T\ \mathbf{r}_k} \tag{4.15}$$

Assim, do passo 5 tem-se o produto $\mathbf{r}_k^T\,\mathbf{r}_k$ armazenado na variável ρ. Como já salientado no passo 4 esse produto (ρ) também está disponível em *todos* os processos. O produto $\mathbf{r}_{k+1}^T\,\mathbf{r}_{k+1}$ também é obtido à semelhança do passo 4 utilizando a atualização do vetor resíduo efetuada no passo 7. Obtendo-se a variável β pode-se passar ao cálculo do vetor **p** dado no passo 9.

Passo 9 Atualizar vetor **p**

A atualização do vetor p é dada pela relação:

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_{k+1}\,\mathbf{p}_k \tag{4.16}$$

Esta adição vetorial é semelhante ao procedimento do Passo 6, restando como diferença os parâmetros de cada relação.

O procedimento em paralelo para o CG necessita tanto do armazenamento parcial da matriz de coeficientes do sistema **A** quanto do vetor de parâmetros independentes **b**. No caso do MEF, o cálculo em paralelo e respectivo armazenamento parcial da matriz de coeficientes **A** já foi comentado no item 4.3. Para o vetor de parâmetros independentes, que no caso do MEF trata-se do vetor de forças nodais, esse procedimento em paralelo será comentado no próximo item. Pode-se agora passar ao paralelismo para análise não linear.

4.5 ANÁLISE NÃO-LINEAR EM PARALELO.

A análise não-linear desenvolvida neste trabalho utiliza como ferramenta de cálculo o MEF. Assim sendo, para se implementar o paralelismo na não linearidade, necessita-se que ele também seja implementado no **MEF**. Uma vez que esse procedimento em paralelo tenha sido instalado, pode-se passar ao conjunto da análise não linear. Neste item aborda-se essa implementação com comentários dos detalhes envolvidos no paralelismo.

O modelo de não linearidade utilizado neste trabalho foi apresentado no capítulo III. O procedimento numérico desta análise pode ser resumido nas etapas comentadas no item 3.5, que são dadas por:

1. Cálculo do incremento de carga.
2. Resolução do sistema de equações lineares.
3. Cálculo do vetor de forças internas.
4. Cálculo do resíduo.
5. Verificação da convergência.

Para uma maior compreensão deste procedimento, pode-se recorrer ao fluxograma da figura (4.11). Neste fluxograma, apresenta-se de modo completo todas as etapas que compõem a análise não-linear. Portanto, para se explorar com máximo proveito os recursos do paralelismo deve-se procurar paralelizar todas as etapas da não linearidade.

No paralelismo para máquinas com memória distribuída, cada processo executa de modo independente as tarefas designadas ao mesmo. Durante a execução do programa, caso haja necessidade, o processo pode enviar e receber informações de outro processo. Essa troca de informações é realizada por meio de uma biblioteca de troca de mensagens.

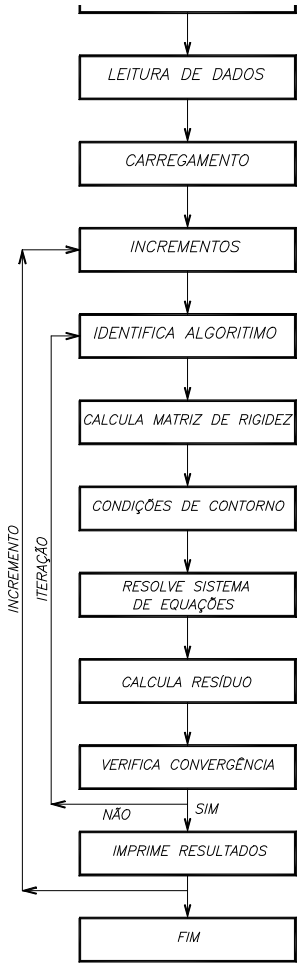


Figura 4.11 Análise não-linear em modo sequencial.

A aplicação do paralelismo à análise não-linear está mostrada na figura (4.12). Neste fluxograma está esquematizada a execução do programa em modo paralelo. Cada processo executa suas tarefas de modo independente e simultaneamente aos outros processos. Pelo fluxograma estão identificadas as rotinas que enviam e recebem dados de outros processos. Os dados trocados, e também a necessidade dos mesmos, serão comentados posteriormente. Comentam-se agora os detalhes específicos do paralelismo em todas as rotinas.

Quando se inicia um processamento em paralelo em máquinas de memória distribuída, vários processos podem ser disparados, como mostra a figura (4.12), dependendo da necessidade do programador. Ao se disparar um processo, o mesmo pode permanecer à espera de dados ou em execução.

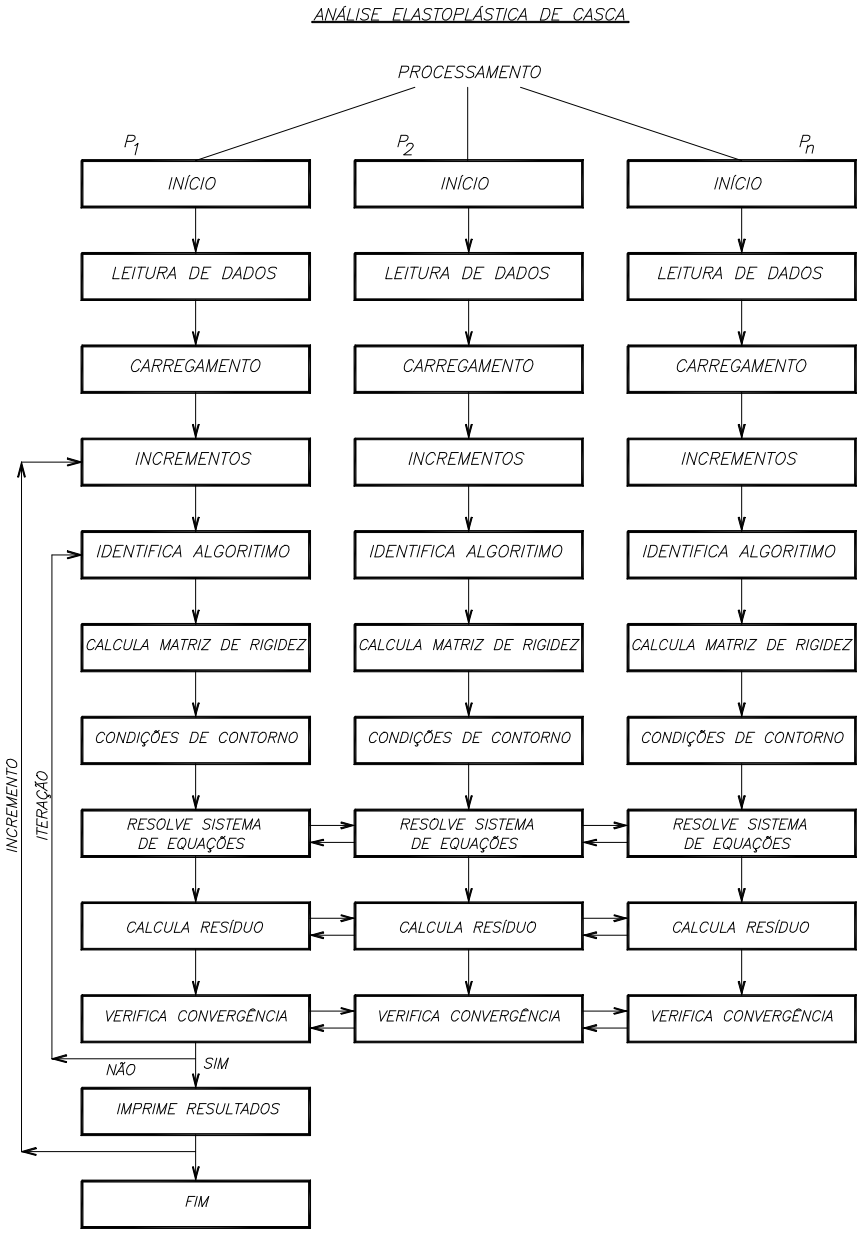


Figura 4.12 Análise não-linear em paralelo.

Leitura de Dados

Neste programa, os dados da estrutura devem estar armazenados em todos os processo. Neste caso, pode-se fazer a leitura do arquivo de dados por um processo e o mesmo remete para os outros processo ou, todos fazem a leitura no arquivo. Neste trabalho optou-se pela leitura autônoma onde cada processo acessa o arquivo de dados e faz a leitura completa.

Após a leitura dos dados, uma importante etapa é realizada simultaneamente, a divisão dos intervalos de pontos nodais. No item sobre o cálculo da matriz de rigidez ficou claro que o algoritmo utilizado realiza esse processo em um *loop* sobre os pontos nodais da estrutura, inserindo em cada linha da matriz os respectivos coeficientes de rigidez. Assim o paralelismo é obtido quando cada processo calcula um determinado número de linhas desta matriz. Para tanto é necessário que o mesmo identifique as linhas sobre as quais ele deverá efetuar o cálculo. Essa identificação é obtida dividindo-se o número de pontos nodais entre os processos. A repartição é obtida com a utilização de uma rotina executada por todos os processos onde *cada processo identifica o seu intervalo*. Esta rotina é mostrada no algoritmo 4.4.

```
ALGORITMO 4.4 DIVISÃO DE INTERVALOS NODAIS

N             NÚMERO DE PONTOS NODAIS
P             NÚMERO DE PROCESSOS
DELTA         INTEIRO
RANGES(1,I)   PONTO NODAL INICIAL DO PROCESSO I
RANGES(2,I)   INTERVALO DE PONTOS NODAIS DO PROC. I
RANGES(3,I)   PONTO NODAL FINAL DO PROCESSO I


DELTA = N/P
REM  = RESTO DO QUOCIENTE  DE N/P
SE (REM = 0) ENTÃO
    S = 0
SENÃO
    S = 1
FIM SE
R = 1
PARA I = 1 ATÉ  NÚMERO DE PROCESSOS P FAÇA
    RANGES (1,I) = R
    D = DELTA + S
    RANGES(2,I) = D
    R = R + D
    RANGES(3,I) = R - 1
    SE  (REM = 1)  ENTÃO
      S = 0
```

```
        SENÃO
          REM = REM - 1
        FIM SE
      FIM FAÇA
```

Na rotina ilustrada no algoritmo (4.4), a variável RANGES armazena os seguintes dados:

RANGES (1,I) → PONTO NODAL INICIAL DO PROCESSO **I**.
RANGES (2,I) → INTERVALO DE PONTOS NODAIS DO PROCESSO **I**.
RANGES (3,I) → PONTO NODAL FINAL DO PROCESSO **I**.

Assim, quando se efetuar o cálculo da matriz de rigidez, cada processo *I* o fará com seu respectivo intervalo – RANGES(1,I) a RANGES(3,I). Além de identificar o seu intervalo todos os processos identificam também o intervalo dos outros processos.

Além de dividir o número de pontos nodais, essa rotina divide também o número de elementos finitos entre os processos. Essa repartição será utilizada na rotina de cálculo do resíduo.

Cálculo do Carregamento

O vetor de cargas nodais aplicado na estrutura também deve ser dividido entre os processos disponíveis. Como já descrito no método dos Gradientes Conjugados, para se resolver o sistema cada processo armazena apenas a respectiva parcela do vetor de cargas nodais. Para se realizar esse cálculo, utiliza-se o procedimento da abordagem nodal já comentada no cálculo da matriz de rigidez. Esse procedimento é representado no algoritmo (4.5) para elementos com dois graus de liberdade por nó.

ALGORITMO 4.5 CÁLCULO DO VETOR DE FORÇAS EXTERNAS

ELIC	ELEMNTO FINITO INICIAL DO CARREGAMENTO
ELFC	ELEMENTO FINITO FINAL DO CARREGAMENTO
FD1 a FD6	CARGAS EXTERNAS NOS ELEMENTOS FINITOS
RANGES(1,I)	PONTO NODAL INICIAL NO PROCESSO I
RANGES(3,I)	PONTO NODAL FINAL NO PROCESSO I.
NCARD	NÚMERO DE CARGAS DISTRIBUÍDAS.
NVARN	NÚMERO DE VARIÁVEIS POR NÓ

```
1     C... ESTRUTURA DE DADOS AUXILIAR
2     PARA I=1 ATÉ NPOIN, FAÇA
3        NCONC(I)=0
4     FIM FAÇA
5     PARA I=1 ATÉ NELEM FAÇA
6        PARA J=1 ATÉ 3 FAÇA
7           P=NP(I,J)
8           NCONC(P)=NCONC(P)+1
9           ELEM_CONC(P,NCONC(P))=I
10          NO_CORRESP(P,NCONC(P))=J
11       FIM FAÇA
12    FIM FAÇA
13    C....CÁLCULO DO VETOR DE FORÇAS NODAIS
14    PARA I=1 ATÉ NCARD FAÇA
15       LER ELIC,ELFC,FD1,FD2,FD3,FD4,FD5,FD6
16       PARA J=RANGES(1,I) ATÉ RANGES(3,I) FAÇA
17          NN_CONC=NCONC(J)
18          PARA K=1 ATÉ NN_CONC FAÇA
19             ELEM=ELEM_CONC(J,K)
20             SE (ELEM. ≥.ELIC .e. ELEM.≤..ELFC) ENTÃO
21                CALCULAR ÁREA DO ELEMENTO ELEM
22                N1=NVARN*(J-ALOCAL+1)-(NVARN-1)
23                N2=N1+1
24                F(N1)=F(N1)+AREA(ELEM)*FD1/3.
25                F(N2)=F(N2)+AREA(ELEM)*FD2/3.
26             FIM SE
27          FIM FAÇA
28       FIM FAÇA
29    FIM FAÇA
```

À semelhança do cálculo da matriz de rigidez, para cada ponto nodal o algoritmo identifica os elementos finitos que concorrem a esse ponto e calcula a contribuição de cada elemento finito à respectiva posição no vetor de forças externas. Para melhor entendimento recorre-se ao exemplo da figura (4.2). Neste caso, supõe-se a existência de dois processos. Para o processo 1 estão alocados os pontos 1 a 3 e para o processo 2 estão alocados os pontos 4 a 6. Assim quando o processo 2 chama o ponto 4, o algoritmo identifica os elementos finitos 1, 2 e 3 e calcula a respectiva contribuição de cada elemento ao vetor de forças externas. Portanto, cada processo calcula somente sua parcela do vetor de forças externas.

No algoritmo 4.4 está evidenciado o procedimento para cálculo do vetor de forças nodais para carregamento distribuído. Quando se tem carregamento concentrado, na linha 14 desse algoritmo muda-se a variável NCARD para NCARC, e na linha 15 deve-se ler apenas o nó e as cargas externas.

Cálculo dos Incrementos de Cargas

O incremento de carga a ser aplicado na estrutura é uma fração do carregamento total. Uma vez que a carga total se encontra armazenada em cada processo, deve-se somente efetuar o cálculo dessa fração em cada processo, tratando-se de uma simples operação.

Identificação do Algoritmo

Nesse passo identifica-se o tipo de algoritmo adotado para a análise não-linear. Neste trabalho tem-se a opção de adotar algoritmo com matriz secante (considerando a rigidez inicial) ou matriz tangente (considerando a rigidez atualizada), conforme relação (3.67). Note-se que todos o processos efetuam essa identificação.

Cálculo da Matriz de Rigidez

O algoritmo que efetua o cálculo da matriz de rigidez da estrutura está comentado no item 4.3, sendo que os intervalos de cada processo são calculados com o algoritmo (4.4). Aproveitando-se da sua simetria, armazena-se a matriz em colunas,

como na figura (4.13), para que a mesma possa ser usado no método dos Gradientes Conjugados na etapa de multiplicação da matriz pelo vetor.

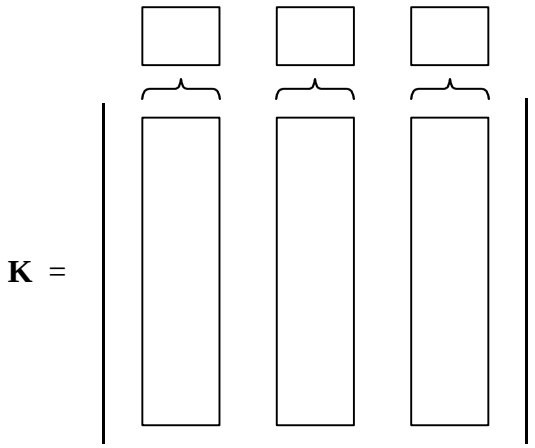


Figura 4.13 Armazenamento da matriz de rigidez.

Imposição das Condições de Contorno

A imposição das condições de contorno é efetuada utilizando a técnica de assumir valores " zero e um " nos coeficientes da matriz de rigidez e no vetor de forças nodais com condições de contorno a impor, ou seja:

$$K_{ij} = \begin{cases} 1 & p/ \quad i = j \\ 0.0 & p/ \quad i \neq j \end{cases} \tag{4.17}$$

$$F_i = 0.0 \tag{4.18}$$

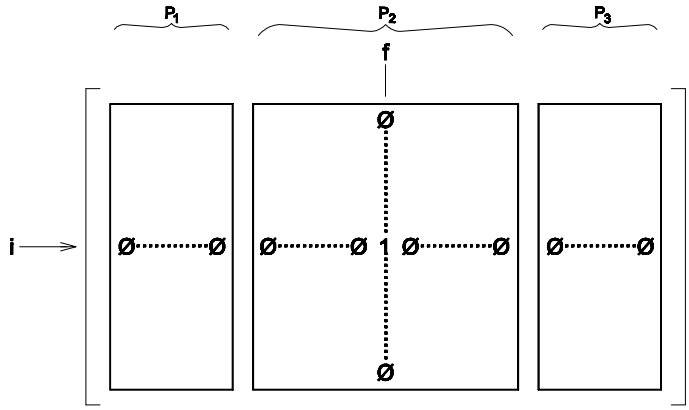


Figura 4.14 Matriz com condição de contorno.

Uma vez que a matriz se encontra armazenada nos processos em execução, todos devem impor as restrições quando se fizer necessário. Na figura (4.14) mostra-se o exemplo de uma condição de contorno onde os processos 1 e 3 devem assumir valores 0.0 na matriz de rigidez para a *linha i*, e o processo 2 deve assumir valores 0.0 para a *linha i* e *coluna j*, exceto quando o coeficiente $i = j$, onde $K_{ij} = 1.0$.

Nesse caso deve-se ter um algoritmo em cada processo que identifique que valor assumir para o grau de liberdade com restrição. No algoritmo a seguir os processos efetuam essa etapa da análise não-linear segundo o critério já exposto.

ALGORITMO 4.6 IMPOSIÇÃO DE CONDIÇÕES DE CONTORNO

NUMNP.....	NÚMERO DE PONTOS NODAIS DA ESTRUTURA
NVARN.....	NÚMERO VARIÁVEIS POR NÓ DO ELEMENTO FINITO.
NVFIX.....	NÚMERO DE PONTOS NODAIS COM CONDIÇÃO DE CONTORNO A IMPOR
RANGES (1,P).....	PONTO NODAL INICIAL DO PROCESSO P.
RANGES (3,P).....	PONTO NODAL FINAL DO PROCESSO P.
M(I).....	PONTO NODAL COM CONDIÇÃO DE CONTORNO A IMPOR.
L1(M(I),K).....	K-ÉSIMO GRAU DE LIBERDADE DO PONTO NODAL M(I) COM CONDIÇÃO DE CONTORNO A IMPOR.
F(I).....	VETOR DE FORÇAS NODAIS
KE(I,J).....	MATRIZ DE RIGIDEZ.

```
1     IMPOSICAO DAS CONDICOES DE CONTORNO EM CADA PROCESSO.
2       NEQ=NUMNP*NVARN
3       NEQLOCAL=(RANGES(3,P)-RANGES(1,P)+1)*NVARN
4       PARA I=1 ATÉ NVFIX FAÇA
5         SE (M(I).≥.RANGES(1,P).E. M(I).≤.RANGES(3,P)) ENTÃO
6           PARA K=1 ATÉ NVARN FAÇA
7           SE (L1(M(I),K).>.0) ENTÃO
8             M1=NVARN*(M(I)-RANGES(1,P))+K
9             M2=NVARN*(M(I)-1)+K
10            PARA J=1 ATÉ NEQ FAÇA
11              KE(J,M1)=0.0
12            FIM FAÇA
13            PARA J=1 ATÉ NEQLOCAL FAÇA
14              KE(M2,J)=0.0
15            FIM FAÇA
16            KE(M2,M1)=1.0
17            F(M1)=0.0
18          FIM SE
19        FIM FAÇA
20        SENÃO
21          PARA K=1 ATÉ NVARN FAÇA
22            SE (L1(M(I),K).>.0) ENTÃO
23              M3=NVARN*(M(I)-1)+K
24              PARA J=1 ATÉ NEQLOCAL FAÇA
25                KE(M3,J)=0.0
26              FIM FAÇA
27            FIM SE
28          FIM FAÇA
```

29	**FIM SE**
30	**FIM FAÇA**

Nota-se na linha 5 do algoritmo (4.6) que o mesmo identifica se o grau de liberdade com condição de contorno a impor pertence ao processo em questão – processo P2 na figura (4.14). Se não for o caso, deve impor somente a condição de nulificar os coeficientes da matriz de rigidez – processos P1 e P3 da figura (4.14).

Resolução do Sistema de Equações.

A etapa de resolução do sistema de equações foi comentada no item 4.4. Contudo, é importante salientar que a matriz de rigidez está armazenada em forma de colunas como na figura (4.13), onde cada processo é responsável por sua parcela no cálculo da matriz. Esse modo de armazenamento da matriz é uma necessidade neste método de resolução de sistemas lineares em paralelo.

As condições de contorno também foram impostas de modo paralelo, como já exposto na seção anterior.

Cálculo do Resíduo.

Após o cálculo dos deslocamentos nodais na etapa de resolução do sistema de equações, pode-se passar à etapa de cálculo do resíduo. Para se calcular o resíduo em uma iteração dentro de um incremento de carga, necessita-se calcular o vetor de forças internas, que resulta da integração do estado de tensões atual da estrutura (conforme equações 3.78a e 3.78b). Observe-se que o vetor de forças internas é calculado no domínio do elemento sendo originalmente uma integral de volume.

Em modo seqüencial o cálculo do vetor de forças internas é feito em um *loop* sobre o número de elementos finitos da malha, calculando-se o vetor de forças internas para cada elemento conforme o modelo de análise não-linear descrito no item 3.5. Após calculados os vetores de forças internas de todos os elementos, pode-se passar ao cálculo do vetor de forças internas nodais, que será resultado da contribuição das forças internas de todos os elementos finitos que concorrem a um determinado nó. Pode-se visualizar melhor esse procedimento com o auxílio da

figura (4.15). Neste caso o vetor de forças internas nodais para o nó 5 será resultado da contribuição das forças internas dos elementos finitos 1, 2, 3, 6, 7 e 8.

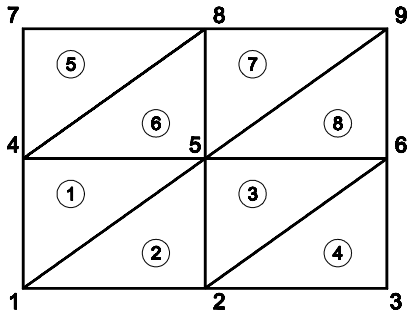


Figura 4.15 Malha para cálculo do vetor de forças nodais.

Para se efetuar esse procedimento em modo paralelo, pode-se dividir o número de elementos finitos entre o número de processos em execução. Cada processo efetuaria o cálculo do vetor de forças internas para os elementos destinados ao mesmo. No caso do exemplo da figura (4.12), supondo-se a existência de dois processos em execução, pode-se alocar para o processo 1 os elementos 1 a 4 e para o processo 2 os elementos 5 a 8. Cada processo efetuaria de modo independente o cálculo do vetor de forças internas para cada elemento.

Resta neste caso o cálculo das contribuições dos elementos para o vetor de forças internas nos respectivos nós. Uma dificuldade que surge pode ser ilustrada com auxílio do exemplo anterior. Como já descrito, para se calcular o vetor de forças internas para o nó 5, necessita-se da contribuição dos elementos 1, 2, 3 ,6, 7 e 8. No caso da execução em paralelo, os elementos 1, 2 e 3 serão calculados no processo P1 e os elementos 6, 7 e 8 serão calculados no processo P2. Para se resolver esse problema adotou-se a estratégia de, após efetuados os respectivos cálculos, cada processo enviar seus resultados a todos os outros processos bem como receber dos mesmos seus respectivos resultados.

Assim, quando se fizer necessário adiciona-se para o nó 5 os resultados dos elementos 1, 2, 3, 6, 7 e 8, o processo que for responsável por essa tarefa (P2) poderá efetuá-la sem nenhuma dificuldade, pois o vetor de forças nodais destes elementos se encontra armazenado no mesmo.

Para se adicionar a contribuição de todos os elementos finitos que se encontram na vizinhança de determinado ponto nodal, necessita-se da identificação destes elementos. Essa identificação pode ser feita com o auxílio do algoritmo para abordagem nodal (linhas 3 a 13 do algoritmo 4.7), apresentado no procedimento de cálculo paralelo da matriz de rigidez. Essa contribuição é realizada pelas linhas 15 a 32 do algoritmo (4.7)

Finalizada a contribuição dos elementos para o vetor de forças internas nodais, pode-se completar o paralelismo dentro dessa etapa, passando-se ao cálculo do vetor resíduo. O vetor resíduo é dado pela equação (3.79), sendo realizado para cada grau de liberdade de cada ponto nodal da estrutura, ou seja:

$$\boldsymbol{\emptyset} = \begin{Bmatrix} \emptyset_1 \\ . \\ . \\ . \\ \emptyset_i \end{Bmatrix} = \begin{Bmatrix} F_{ext,1} \\ . \\ . \\ . \\ F_{ext,n} \end{Bmatrix} - \begin{Bmatrix} F_{int,1} \\ . \\ . \\ . \\ F_{int,n} \end{Bmatrix} \qquad (4.19)$$

Uma vez que o número de pontos nodais da estrutura se encontra dividido entre o número de processos em execução, a realização da soma algébrica da equação (4.19) pode ser efetuada de modo direto, tendo-se o cuidado de fazê-la em um *loop* somente para os graus de liberdade de responsabilidade do processo em questão. No algoritmo (4.7) essa soma é representada pelas linhas 33 a 36, supondo-se um elemento finito com dois graus de liberdade por nó.

ALGORITMO 4.7 CÁLCULO DO RESÍDUO

F_LOAD(I)..... VETOR DE FORÇAS INTERNAS NODAIS
ELOAD(I,J)..... VETOR DE FORÇAS INTERNAS NO ELEMENTO I.
FT_LOAD(I)..... VETOR DE FORÇAS EXTERNAS DADO PELO INCREMENTO DE CARGA.

```
1          C   CÁLCULO DA CONTRIBUIÇÃO DOS ELEMENTOS PARA O VETOR
2     DE
3          C   FOR ÇAS INTERNAS NODAIS.
4          PARA I=1 ATÉ NPOIN, FAÇA
5             NCONC(I)=0
```

```
6          FIM FAÇA
7          PARA I=1 ATÉ NELEM FAÇA
8             PARA J=1 ATÉ 3 FAÇA
9                P=NP(I,J)
10               NCONC(P)=NCONC(P)+1
11               ELEM_CONC(P,NCONC(P))=I
12               NO_CORRESP(P,NCONC(P))=J
13            FIM FAÇA
14         FIM FAÇA
15         NEQLOCAL=(RANGES(3,P)-RANGES(1,P)+1)*NVARN
16         PARA NI=1 ATÉ NEQLOCAL FAÇA
17            F_LOAD(I)=0.0
18            F1_LOAD(I)=0.0
19         FIM FAÇA
20         PARA I=RANGES(1,P) ATÉ RANGES(3,P) FAÇA
21            NELCON=NCONC(I)
22            PARA J=1 ATÉ NELCON FAÇA
23               ELEM=ELEM_CONC(I,J)
24               NO=NO_CORRESP(I,J)
25               N=(I-RANGES(1,P)+1)
26               N1=N*NVARN-(NVARN-1)
27               N2=N1+1
28               N11=NVARN*NO-(NVARN-1)
29               N22=N11+1
30               F_LOAD(N1)=F_LOAD(N1)+ELOAD(ELEM,N11)
31               F_LOAD(N2)=F_LOAD(N2)+ELOAD(ELEM,N22)
32            FIM FAÇA
33         FIM FAÇA
34         C   CALCULO DO RESIDUO ( Ψ = F_EXT  -  F_INT)
35         PARA I=1 ATÉ NEQLOCAL FAÇA
36            F_LOAD(I)=FT_LOAD(I)-F_LOAD(I)
           FIM FAÇA
```

Verificação da Convergência

A verificação da convergência é feita usando-se a relação (3.80), ou seja:

$$\frac{\| \Ä \mathbf{U}^r \|}{\| \mathbf{U} \|} \text{x} 100 \leq \text{Tolerância} \qquad (4.20)$$

Neste caso, para se efetuar essa verificação em paralelo adota-se o seguinte procedimento:

. Cada processo calcula sua norma parcial (tanto do deslocamento na iteração $\| \Delta \mathbf{U}^r \|$ quanto do deslocamento total $\| \mathbf{U} \|$), envia seu resultado a todos os outros processos e recebe os resultados parciais dos outros processos.

. Todos os processos totalizam a norma e verificam a convergência. Se não se verificar a convergência, passa-se a outra iteração; caso contrário, imprime os resultados parciais e passa-se a outro incremento de carga.

O passo 1 da verificação da convergência é efetuado de modo direto, uma vez que, a partir da resolução do sistema de equações, se encontra armazenado em cada processo os respectivos valores dos deslocamentos na iteração e deslocamentos totais. Neste caso é necessário apenas instalar um *loop* efetuando-se o produto dos deslocamentos, considerando-se somente os graus de liberdade armazenados no processo, que são dados por:

(Nº de graus de liberdade por nó) * (RANGES (2,I)) (4.21)

Se a convergência atingiu a tolerância desejada, o processo pai imprime os resultados parciais e passa-se a um novo incremento de carga. Caso contrário, volta-se a uma nova iteração, onde o vetor de cargas aplicadas será o resíduo obtido na iteração anterior.

CAPÍTULO V

EXEMPLOS

5.1 INTRODUÇÃO

Neste capítulo mostram-se os resultados dos exemplos elastoplásticos executados em processamento paralelo os quais encontram-se em bibliografia conhecida. Além dos resultados para o comportamento elastoplástico, mostra-se o resultado obtido com a implementação em paralelo do modelo não linear. Apresentam-se também comparações com resultados seqüenciais.

5.2 PLACA APOIADA ESTRATIFICADA

Como ilustração do modelo elastoplástico utilizado, apresenta-se o exemplo da placa apoiada com comportamento elastoplástico perfeito. Neste caso a comparação é realizada com os resultados apresentados em OWEN e HINTON (1980) para placa estratificada. Para tanto, se faz uso dos resultados do elemento de placa heterosis com nove pontos nodais. Para o elemento de placa utilizado nesta comparação adotou-se quatro pontos de integração no domínio do elemento e nove pontos de Gauss ao longo da espessura. Foram utilizados os seguintes parâmetros para simulação da placa:

Módulo de Young **E = 10.92**

Coeficiente de Poisson ν **= 0.3**

Espessura **t = 0.01**

Largura **L = 1.0**

Parâmetro de Endurecimento **H = 0.0**

Limite Elástico σ_Y **= 1600**

Carregamento **q = 1.0**

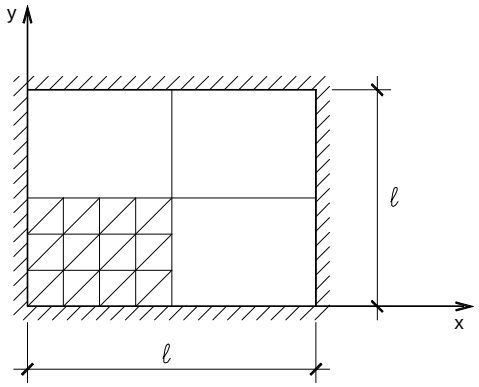


Figura 5.1 Placa quadrada apoiada nos quatro lados.

O carregamento aplicado é distribuído em cinqüenta incrementos iguais de carga, e considera-se critério de convergência em deslocamentos com tolerância de 0.01 % . A malha utilizada na obtenção dos resultados é de 100 pontos nodais, totalizando 600 graus de liberdade. Os resultados obtidos para o comportamento elastoplástico estão apresentados na figura 5.2, onde se mostra o deslocamento vertical (w) no ponto central da placa. Neste caso, o elemento DKT que incorpora os efeitos de flexão, apresenta comportamento próximo ao apresentado na referência utilizada.

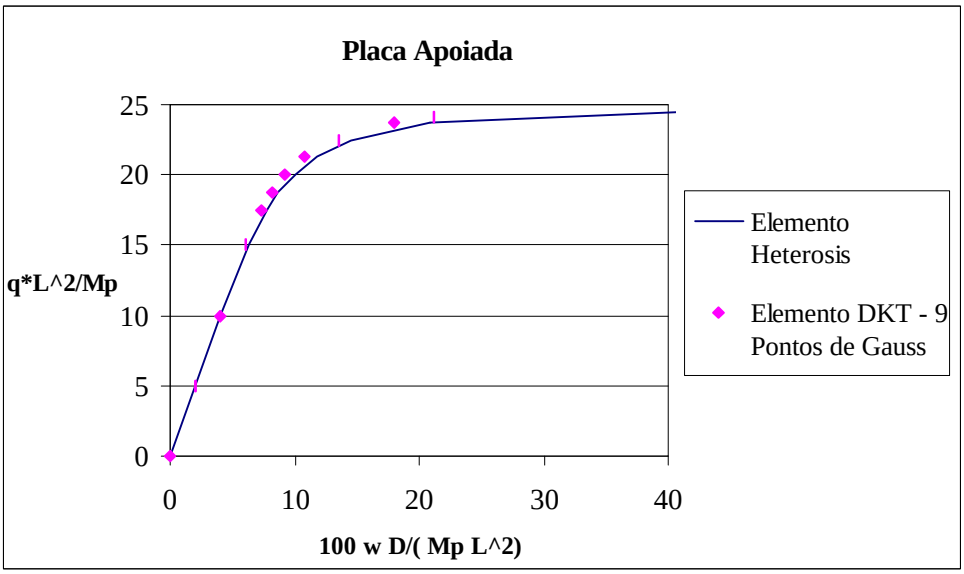


Figura 5.2 Curva Carga x Deslocamento do ponto central da placa.

Os resultados da implementação em paralelo do exemplo da placa apoiada estão apresentados em seguida. Na tabela 5.1 mostra-se o tempo total de execução do programa para 1, 2 e 3 processadores. Na figura 5.3 está apresentado o speed-up para o exemplo da placa apoiada e na figura 5.4 mostra-se a eficiência obtida com a implementação em paralelo.

Tabela 5.1 Tempo total de processamento para placa apoiada.

Número de Processadores	Tempo Total de Processamento (seg)
01	3352.126
02	2020.983
03	1558.982

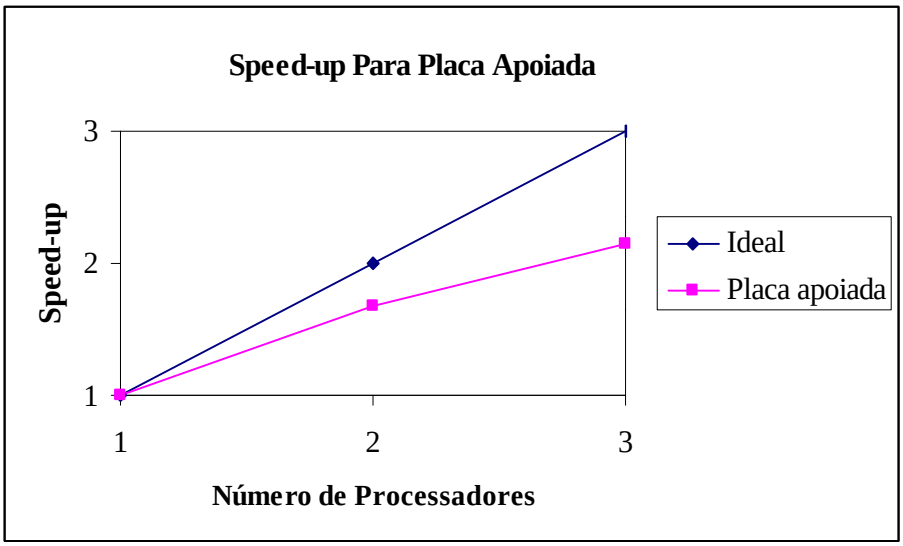


Figura 5.3 Speed-up para placa apoiada

Neste exemplo obteve-se eficiência de 83 % para dois processadores e 71 % para três processadores. Como se pode notar com os resultados obtidos, a implementação em paralelo apresentou-se vantajosa pela redução do tempo total de processamento. A eficiência obtida neste caso apresenta-se elevada, restando a comprovação para um número maior de processadores.

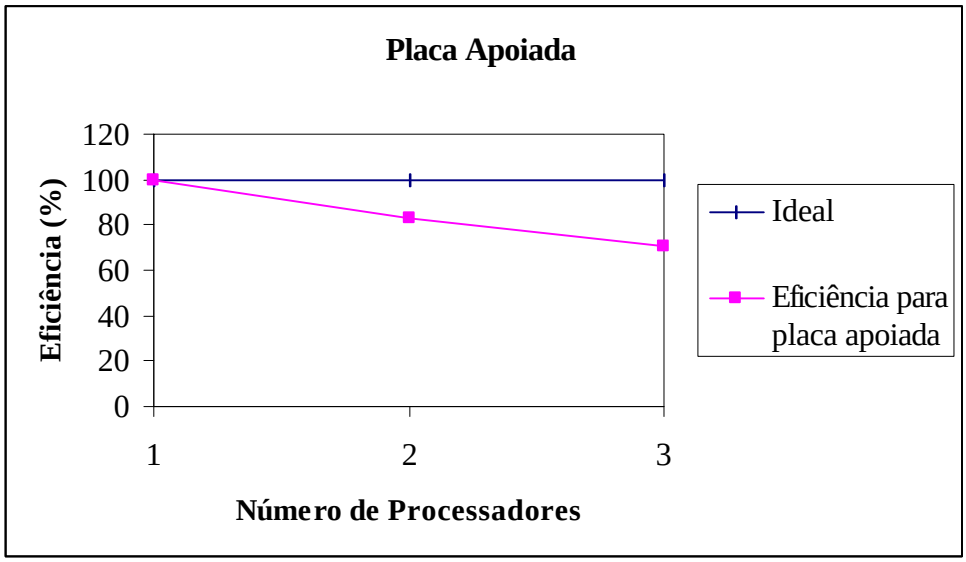


Figura 5.4 Eficiência para placa apoiada.

5.3 CASCA CILÍNDRICA

Além do exemplo anterior, apresenta-se também um exemplo da implementação em paralelo para uma casca cilíndrica, conforme figura 5.5. A casca indicada neste exemplo é referência clássica na análise estrutural. A comparação realizada neste texto é obtida a partir dos resultados para o elemento TSL6, conforme MESQUITA (1998), para o qual utilizou-se uma malha totalizando 1283 graus de liberdade. Esta casca foi simulada com os seguintes parâmetros:

Módulo de Young **$E = 2.1 \times 10^{10}$ N/m^2**

Coeficiente de Poisson **$\nu = 0.0$**

Espessura **$t = 0.076$ m**

Largura **$L = 15.2$ m**

Parâmetro de Endurecimento **$H = 0.0$**

Limite Elástico **$\sigma_Y = 4.1 \times 10^6$ N/m^2**

Carregamento **$q = 3.0 \times 10^3$ N/m^2**

Ângulo de Abertura **$\phi = 40^0$**

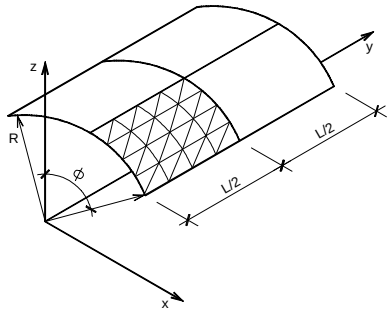


Figura 5.5 Casca cilíndrica apoiada em paredes rígidas.

Para o elemento DKTFF utilizou-se um total de 117 pontos nodais, 192 elementos com divisão de 8x12x2 elementos e nove pontos de Gauss ao longo da espessura. A análise elastoplástica foi dividida em cinqüenta incrementos iguais de cargas e admitida tolerância em deslocamento de 0.1 %. Os resultados obtidos referem-se ao deslocamento vertical do ponto da borda livre, intermediário entre os

apoios da casca. O resultado para a análise elastoplástica está apresentado na figura 5.6.

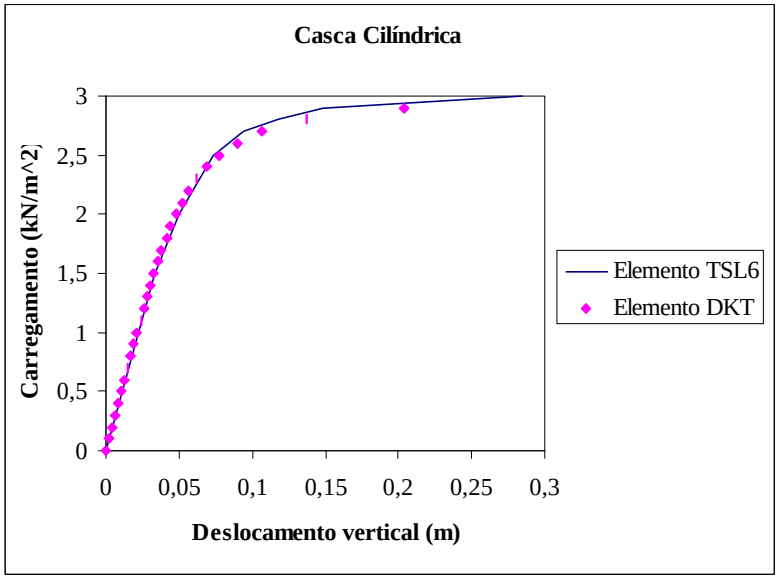


Figura 5.6 Deslocamento transversal do ponto intermediário da borda livre

Os resultados para a execução em paralelo deste exemplo estão apresentados nas figuras 5.7, 5.8 e na tabela 5.2. Para este exemplo a eficiência obtida com dois processadores e tres processadores foi de 83.5 % e 76.7% respectivamente. O tempo total de execução com três processadores reduziu-se para menos da metade do tempo para um processador, que mostra uma clara vantagem em se usar o paralelismo neste caso.

Tabela 5.2 Tempo total de processamento para casca cilíndrica.

Número de Processadores	Tempo Total de Processamento (seg)
01	14800.848
02	8881.531
03	6518.990

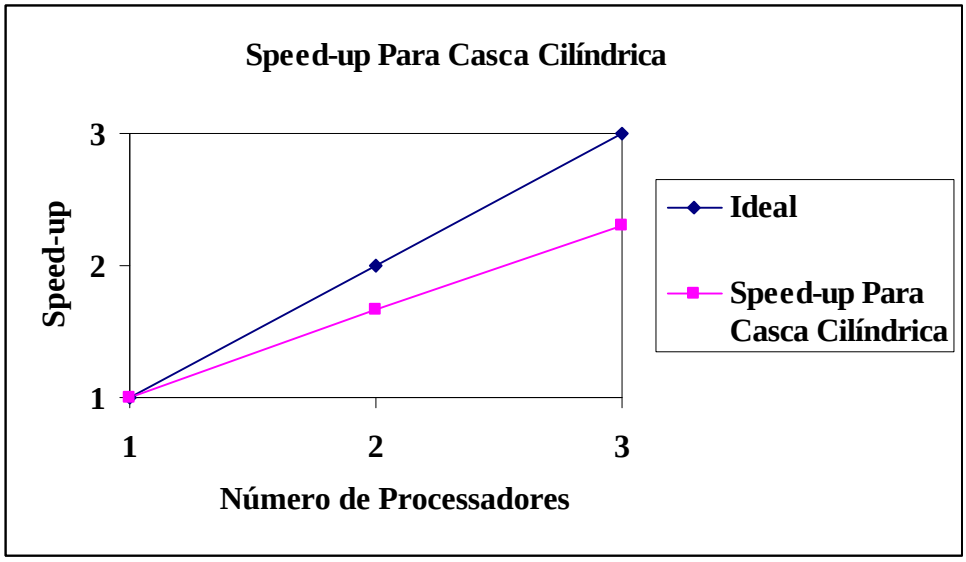


Figura 5.7 Speed-up para casca cilíndrica.

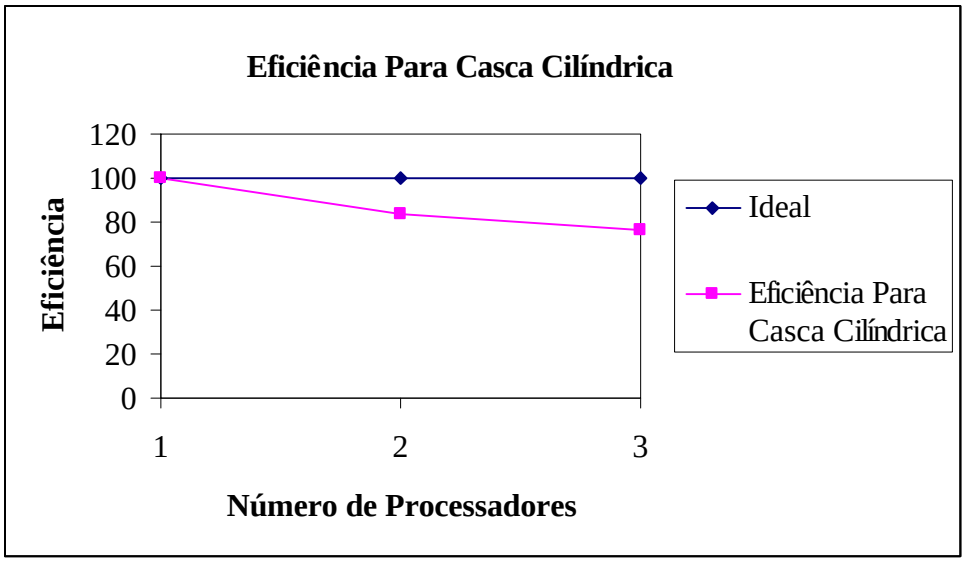


Figura 5.8 Eficiência para casca cilíndrica.

Os resultados obtidos para o exemplo da casca cilíndrica apresentam-se satisfatórios considerando-se a redução do tempo de processamento. A eficiência obtida com o número de processadores disponível também apresenta-se satisfatória.

Entretanto deve-se fazer a comprovação desses resultados para um número maior de processadores.

CAPÍTULO VII

BIBLIOGRAFIA

ADELI, H.; KAMAL, O. (1992). Concurrent analysis of large structures-I. Algoritms. *Computers and Structures* v. 42, n. 3, p. 413-424.

ADELI, H.; KAMAL, O. (1992). Concurrent analysis of large structures-II. Algoritms. *Computers and Structures* v. 42, n. 3, p. 413-424.

ADELI, H., KUMAR, S. (1995) Concurrent structural optimization on massively parallel supercomputer. *Journal of Structural Engineering*, Novembro, p1588-1597

AJIZ, M. A. (1990). Application of parallel data structures to the finite element solution of circular flat plates. *Computers and Structures* v. 37, n. 5, p. 647-662.

ALMEIDA, V. A. F.; ÁRABE, J. N. C. (1991). *Introdução à supercomputação*. São Paulo LTC

ALMEIDA, V. S. (1999). *Uma adaptação do MEF para análise em multicomputadores: Aplicações em alguns modelos estruturais*. 126p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo.

AMDAHL, G.(1967). *Validity of the single processor aproach to achieving large scale computing capabilities.* AFIPS Proceeding of the sprint joint computer conference, n. 30, p. 438-485.

BADDOURAH, M. A. ; NGUYEN, D. T.(1994). Parallel-vector computations for geometrically nonlinear finite element analysis. *Computers and Structures* v. 51, n. 6, p. 785-789.

BARRAGY, E.; CAREY, G. F. (1988). A parallel element-by-element solution scheme. *International Journal For Numerical Methods In Engineering* v. 26, p. 2367-2382.

BELYTSCHKO, T.; GILBERTSEN, N. D. (1992). Implementation of mixed time integration techinique on a vectorized computer with shared memory. *International Journal For Numerical Methods In Engineering* v.35, n. 9, Nov. 30, p. 1803-1828.

CARTER JR, W. T.; SHAW, T. L. ; LAW, H. K. (1989). A parallel finite element method and its prototype implementation on a hypercube. *Computers and structures* , v.31, n.6, p921-934.

CHIEN, L. S.; SUN, C. T. (1989). Parallel processing techiniques for finite element analysis of nonlinear large truss structures. *Computers and Structures*, v. 31, n. 6, p. 1023-1029.

CHIANG, K. N.; FULTON, R. E. (1990). Concepts and implementation of parallel finite element analysis. *Computers and Structures* v. 36, n. 6, p. 1039-1046.

CIMMERMAN, M. (1996). *Resolução de sistemas lineares via métodos iterativos com pré-condicionadores – Aplicação em problemas de engenharia de estruturas*. São Paulo. 111p. Dissetação (Mestrado) – Escola Politécnica da Universidade de São Paulo.

CHEN, W. F. (1982) *Plasticity in reinforced concrete*. New York, McGraw Hill.

CHEN, H. C.; BYREDDY, V. (1997). Solving plate bending problems using finite strips on networked workstations. *Computer and Structures* v. 62, n. 2, p. 227-236.

CODENOTTI, B.; LEONCINI, M.(1991). *Parallel complexity of linear system solution.* Singapore, World Scientific Publishing.

De-CEGAMA, A. L. (1989). *Parallel processing architetures and VLSI hardware.* EUA, Prentice Hall.

DEKKER, T. J.; HOFFMANN, W; POTMA, K.(1994). Parallel algorithms for solving large linear systems. *Journal of Computacional and Applied Mathematics* v. 50, p. 221-232

FREEMAN, L.; PHILIPS, C. (1992). *Parallel Numerical Algorithms.* Inglaterra, Prentice Hall International.

FLYNN, M. (1972). *Some Computer Organizations And Their Effectiveness.* IEEE Transactions, v. 21.

EL-SAYED, M. E. M.; HSIUNG, C. K. (1990). Parallel finite element computation with separate substructures. *Computers and Structure* v. 36, n. 2, p. 261-265.

EL-SAYED, M. E. M.; HSIUNG, C. K. (1994). Comparison between two decomposition approaches for parallel computation of structural optimization. *Computers and Structure* v. 52, n. 4, p. 719-722.

FARHAT, C.; WILSON, E. (1988). A parallel active column equation solver. *Computers and Structures*, v. 28, n. 2, p. 289-304.

FARHAT, C.; CRIVELLI, L. (1989). A general aproach to nonlinear fe computations on shared-memory multiprocessors. *Computers Methods In Applied Mechanics and Engineering* v. 72, p. 153-171.

FERIANI, A.; FRANCHI, A.; GENNA, F. (1996) An incremental elastic-plastic finite element solver in a workstation cluster enviroment. Part I: Formulations and parallel processing. *Computer Methods In Applied Mechanics And Engineering.* 130, p 289-298.

FERIANI, A.; FRANCHI, A.; GENNA, F. (1996) An incremental elastic-plastic finite element solver in a workstation cluster enviroment. Part II: Performance of a first implementation. *Computer Methods In Applied Mechanics And Engineering.* 130, p 299-318.

GOEHLICH, D.; KOMZSIK, L.; FULTON, R. E. (1989). Application of a parallel equation solver to static fem problems. *Computers and Structures* v. 31, n. 2, p. 121-129.

GOLUB, G. H. ; LOAN, Van C. F. (1984). *Matrix computation*. Johns Hopkins Press, Baltimore.

HILL, R. (1950), *The matemathical theory of plasticity*. Oxford, Claredon Press.

JOHNSSON, S. L.; MATHUR, K. K. (1990). Data structures and algorithms for the finite element method on a data parallel supercomputer. *International Journal For Numerical Methods In Engineering* v. 29, p. 881-908.

JAQUES, M. W. S.; ROSS, C. T. F. ; STRICKLAND, P. (1996). Exploiting inherent parallelism in non-linear finite element analysis. *Computers and Structures* v. 58, N. 4, p 801-807.

LAKSHMIVARAHN, S.; DHALL, S. K. (1990). *Analysis and Design of Parallel Algorithms.* EUA, McGraw Hill.

LAW,H. K. (1986). A parallel finite element solution method. *Computers and Strutures*, v. 23, n. 6, p. 845-858 .

LAW, K. H.; MACKAY, D. R. (1993). Parallel row-oriented sparse solution method for finite element structural analysis. *International Journal for Numerical Methods in Engineering* v. 36, n. 17, Sep. 15, p. 2895-2919.

LIU, J.; WU, G. (1994). Parallel analysis of rotacional periodic structures. *Computers and Structures* v. 50, n. 6, Mar 17, p. 785-795.

OÑATE, E. (1992). *Cálculo de estruturas por el método de elementos finitos: Analisis elástico lineal*. Barcelona, Centro Internacional de Métodos Numéricos in Ingenieria.

OU, R.; FULTON, R. E. (1988). An investigation of parallel numerical integration methods for nonlinear dynamics. *Computers and Structures* , v. 30, n 1/2, p. 403-409.

OWEN, D. R. J.; HINTON. E. (1980). *Finite Element in Plasticity: Theory and Pratice.* U. K. Pineridge Press Limited.

QUINN, M. J.(1987). *Designing Efficient algorithms for Parallel Computers.* EUA McGraw Hill Book Company.

PAPADRAKAKIS, M.; DRACOPOULOS, M. C.(1991). Improving the efficiency of preconditioning for iterative methods. *Computers and Structures* v. 41, n.6 p. 1263-1272.

RAO, A. R. M., LOGANATHAN, K., RAMAN, N. V., (1993). Studies on two concurrent solvers for finite element analysis. *Advances in Engineering Softwares* v 18, p. 161-166.

REZENDE,M. N. (1995). *Processamento paralelo em análise estrutural.* Tese (Doutorado) - Escola de *Engenharia* de São Carlos, Universidade de São Paulo.

REZENDE, M. N.; PAIVA, J. B. (1994). *Algorithm for parallel assembling of the stiffness matrix of structural analysis.* Advances In Parallel And Vector Processing For Structural Mechanics p. 43-46

SAXENA, M.; PERUCHIO, R. (1992). Parallel fem algorithms based on recursive spatial decomposition - I. Automatic mesh generation. *Computers na Structures* v. 45, n. 5-6, Dec. 3, p. 817-831.

SAXENA, M.; PERUCHIO, R. (1993). Parallel fem algorithms based on recursive spatial decomposition - II. Automatic analysis via hierarchical substructuring. *Computers na Structures* v. 47, n. 1, Apr. 3, p. 143-154.

SCHMIT, L. A.; LAI, YE-CHEN; (1994). Structural optimization based on preconditioned conjugate gradient analysis methods. *International Journal For Numerical Methods In Engineering* v. 37, n. 6, Mar 30, p. 943-964.

SUN, C. T.; MAO, K. M.(1988). A global-local finite element method suitable for parallel computations. *Computer and Structures* v. 29, n. 2, p. 309-315.

SYNN, S. Y.; FULTON, R. E. (1995). Practical strategy for concurrent substructure analysis. *Computer and Structures* v. 54, n. 5, p. 939-944.

STONE, H. (1987). *High-Performance Computer Architeture.* EUA, Addison-Wesley

TOPPING, B. H. V.; KHAN, A. I. (1996). Subdomain generation for non-convex parallel finite element domains. *Advances in Engineering Software*. 25, p 253-266.

YALAMANCHILI, K. K.; ANAND, S.C.; WARNER, D. D. (1992). Three-dimensional finite element analysis on a hypercube computer. *Computers and structures* v. 42, n. 1, p. 11-20

YAGAWA, G.; SONEDA, N.; YOSHIMURA, S. (1991). Large scale finite element analysis using domain decomposition method on a parallel computer. *Computers and Structures* v. 38, n. 5-6, p. 615-625.

ZHENG, D., CHANG, T. Y. P. (1995). Parallel, Cholesky method on MIMD with shared memory. *Computers and Structures*, v. 56 N. 1 p. 25-38.

ZIENKIEWICZ, O. C., TAYLOR , R. L.(1991). *The Finite Element Method.* EUA, McGraw Hill.