


ANÁLISE ESTRUTURAL POR PARTIÇÃO DE DOMÍNIO  
EM AMBIENTE DE COMPUTAÇÃO PARALELA

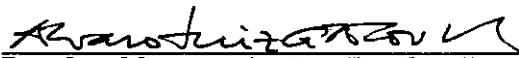
Marcos Queija de Siqueira

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA CIVIL.

Aprovada por:

  
\_\_\_\_\_  
Prof. Edson Castro Prates de Lima, D. Sc.  
(Presidente)

  
\_\_\_\_\_  
Prof. Nelson Francisco Favilla Ebecken, D.Sc.

  
\_\_\_\_\_  
Prof. Alvaro Luiz G. A. Coutinho, D.Sc.

  
\_\_\_\_\_  
Alvaro Maia da Costa, D. Sc.

RIO DE JANEIRO, R.J. - BRASIL

JUNHO DE 1989

SIQUEIRA, MARCOS QUEIJA

Análise Estrutural por Partição de Domínio em  
Ambiente de Computação Paralela. [RIO DE JANEIRO]  
1989.

VI, 99 p. 29,7 cm (COPPE/UFRJ, M. Sc., Engenharia  
Civil, 1989)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Subestruturas

2. Computação Paralela

I. COPPE/UFRJ

II. TITULO (série)

À minha família e, em especial, a meu pai pelo apoio e incentivo nos momentos mais difíceis.

## AGRADECIMENTOS

Ao professor Edison Castro Prates de Lima pelo apoio e orientação.

Ao professor Nelson Favilla Ebecken pelo interesse e motivação oferecidas.

Aos amigos Luís Volnei Sudatti Sagrilo, Mário Ribeiro da Silva Filho e Eduardo Lúcio Mendes Garcia pelo incentivo e pelo companheirismo em todos os instantes.

Ao C.B.P.F. (Centro Brasileiro para Pesquisas Físicas) por permitir o acesso aos seus equipamentos e pelo apoio e colaboração dos seus pesquisadores.

Aos colegas da COPPE/UFRJ, presentes em todos os momentos.

Resumo da tese apresentada à COPPE/UFRJ como parte dos requisitos necessário para obtenção do grau de Mestre em Ciências (M. Sc.).

ANÁLISE ESTRUTURAL POR PARTIÇÃO DE DOMÍNIO  
EM AMBIENTE DE COMPUTAÇÃO PARALELA.

MARCOS QUEIJA DE SIQUEIRA

JUNHO - 1989

Orientador: Edson Castro Prates de Lima  
Programa: Engenharia Civil

O objetivo deste trabalho é estudar a adequação de uma análise estrutural por partição de domínio em computadores de arquitetura paralela. Para a partição automática do domínio é apresentado um algoritmo que opera como um pré-processador. A análise estrutural por subestruras foi implementada no sistema multiprocessador A.C.P. (Advanced Computer Program), e através de métodos de avaliação de performance analisou-se o desempenho do sistema submetido a situações diversas.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M. Sc.)

STRUCTURAL ANALYSIS THROUGH DOMAIN DECOMPOSITION  
IN PARALLEL COMPUTERS

MARCOS QUEIJA DE SIQUEIRA

June - 1989

Thesis Supervisor: Edson Castro Prates de Lima

Department: Civil Engineering

In this work, the suitability of a domain decomposition technique for structural analysis in parallel computers is studied. An algorithm for automatic domain decomposition is presented, and it is embedded as a preprocessor in a computer program for finite element analysis. The whole solution procedure was implemented in the A.C.P. (Advanced Computer Program) multiprocessor system, where the performance was evaluated solving a number of examples.

## ÍNDICE

	Pag.
CAPITULO I - INTRODUÇÃO.....	1
CAPITULO II - MÉTODOS DE ANÁLISE.....	4
II.1 - Análise estática linear de estruturas usando a técnica de subestruturas.....	4
II.2 - Características principais e aplicabilidades do método.....	11
CAPITULO III - EQUIPAMENTO E PERFORMANCE.....	16
III.1 - Computadores paralelos.....	16
III.2 - O sistema A.C.F.....	21
III.3 - Avaliação de performance.....	27
CAPITULO IV - ESTRUTURA DE PROGRAMAÇÃO.....	32
IV.1 - Considerações gerais.....	32
IV.2 - Partição automática do domínio.....	33
IV.3 - Subestruturacão.....	37
CAPITULO V - EXEMPLOS E APLICAÇÕES.....	46
V.1 - Exemplos de partição automática do domínio.....	50
V.2 - Exemplos de análise por subestruturas.....	55
CAPITULO VI - CONCLUSÕES.....	93
REFERÊNCIAS BIBLIOGRÁFICAS.....	97

## CAPÍTULO I

## INTRODUÇÃO

O surgimento de novas concepções em arquiteturas de computadores tem motivado o estudo de métodos de análise e técnicas de programação que melhor se adaptem a essa nova realidade. Já existem disponíveis ,hoje em dia, vários sistemas computacionais de múltiplos processadores. Torna-se, então, importante a pesquisa dos sistemas mais adequados para a resolução dos vários problemas de engenharia.

A aplicação do método dos elementos finitos na análise estrutural é feita através da discretização da estrutura em um mosaico de partições elementares de dimensões finitas. Estas partições podem ser constituídas por elementos finitos básicos do tipo triangular, quadrilátero, tetraedros, paralelepípedos, ou qualquer outro tipo de elemento adequado ao problema a ser estudado. A técnica de subestruturas permite a combinação destes elementos básicos para a formação de superelementos.

O superelemento se constitui num modelo de subestrutura. Ele pode ser formado por um conjunto qualquer de elementos finitos básicos e/ou de outros superelementos, previamente definidos. O termo superelemento é usado deliberadamente de forma a enfatizar que eles são tratados no modelo estrutural de programação da mesma forma que os elementos finitos básicos.

A formulação do método dos elementos finitos usando a técnica de subestruturas é bem antiga. Os trabalhos apresentados por KIRSCH [1], PRZEMIENIECKI [2], PETERSON [3] e AHMED [4] mostram de forma completa toda a teoria para uma análise estática linear.

Este trabalho tenta adaptar esta formulação visando a sua adequação a computação paralela. Os recentes trabalhos de FARHAT [5],[6],[7], OMID [8] e PARK [9] demonstram a potencialidade deste método e a necessidade de programas que operem eficientemente com qualquer número de processadores

O capítulo II tem como objetivo apresentar todo o desenvolvimento teórico necessário para a análise estática por subestruturas. Para ressaltar a versatilidade do método dedica-se, neste capítulo, um item para abordar as suas principais características e aplicabilidades.

O capítulo III apresenta o conceito de computação paralela. São objeto de análise: a nova tecnologia com as suas diversas configurações em "hardware", a classificação dos vários sistemas existentes e as formas modernas de avaliação de performance. Um dos itens deste capítulo descreve o sistema A.C.P. (Advanced Computer Program), como ele se encaixa dentro da classificação geral de sistemas multiprocessadores e algumas peculiaridades desta máquina.

O capítulo IV é destinado a estrutura de programação adotada para este trabalho. Foram desenvolvidos dois programas: um pré-processador e um processador propriamente dito. O pré-processador tem a função de

particionar o domínio, e o processador de realizar uma análise estática completa. Todo o trabalho de programação foi feito sempre tendo como objetivo explorar de forma eficiente as características do sistema A.C.P.. Este capítulo é subdividido em três itens: o primeiro estabelece justificativa para os vários caminhos adotados, o segundo e terceiro apresentam algoritmos e fluxogramas usados nos dois programas.

O capítulo V analisa os resultados obtidos pelos dois programas. São apresentadas tabelas e gráficos elucidativos, permitindo uma visão abrangente das diversas análises.

Baseando-se nos resultados dos exemplos analisados são apresentadas no capítulo VI as conclusões finais. Como forma de incentivar trabalhos futuros são dadas sugestões de possíveis caminhos para novas pesquisas. A computação paralela é uma área relativamente nova de pesquisa e, portanto, bastante inexplorada e fecunda de potencialidade.

## CAPÍTULO II

## MÉTODOS DE ANÁLISE

## II.1 - ANÁLISE ESTÁTICA LINEAR DE ESTRUTURAS USANDO A TÉCNICA DE SUBESTRUTURAS

Na discretização do domínio contínuo pelo método dos elementos finitos utilizando o modelo de deslocamentos, o sistema de equações de equilíbrio para uma análise estática linear, pode ser escrito na forma matricial por:

$$[K] \cdot \{U\} = \{F\} \quad (\text{II.1})$$

onde,

$[K]$  → matriz de rigidez global

$\{U\}$  → vetor de deslocamentos nodais

$\{F\}$  → vetor de cargas nodais

O vetor  $F$  e a matriz  $K$  podem ser obtidas mediante a soma de contribuições individuais de elementos finitos básicos ou de superelementos integrantes do sistema estrutural.

Os superelementos são formados a partir dos elementos finitos básicos ou de outros superelementos, utilizando-se a técnica de subestruturas.

Ao particionarmos o domínio surgem dois tipos distintos de deslocamentos ou incógnitas, os internos e os de interface (fig. II.1), sendo o deslocamento total da estrutura dado pela soma vetorial de ambos.

$\{ U_I \}$  → vetor de deslocamentos para nós internos

$\{ U_P \}$  → vetor de deslocamentos para nós de interface

$$\{ U \} = \{ U_I \} + \{ U_P \} \quad (II.2)$$

A separação entre nós internos e de interface leva a consideração dos vetores  $\{ F_I \}$  e  $\{ F_P \}$  .

$\{ F_I \}$  → vetor de cargas nodais internas

$\{ F_P \}$  → vetor de cargas nodais partilhadas

$$\{ F \} = \{ F_I \} + \{ F_P \} \quad (II.3)$$

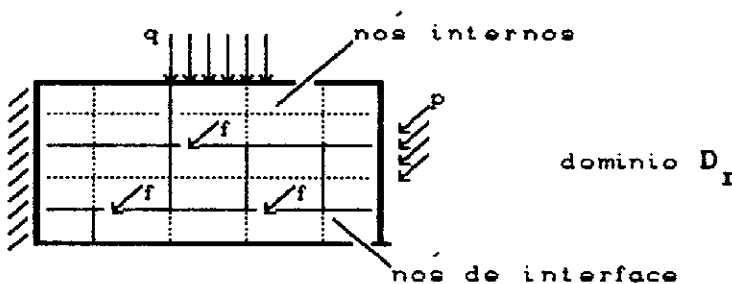


Figura II.1 - Placa mostrando nós internos e de interface depois da decomposição do domínio inicial

Se considerarmos o domínio completo como uma única subestrutura, o sistema de equações, neste caso, passaria a ser o da eq.(II.1), equivalente a uma análise convencional, não sendo necessário, portanto, a consideração de nós internos.

A partição do domínio em  $m$  regiões vai originar um sistema de equações local e distinto para cada região, o que possibilita uma análise em separado para cada subestrutura. O vetor de deslocamento e o de cargas seriam formados pelo conjunto de vetores de cada subestrutura, na forma seguinte:

$$\{U_I\} = \left\{ \left\{ U_I \right\}^1, \left\{ U_I \right\}^2, \left\{ U_I \right\}^3 \dots \dots \left\{ U_I \right\}^m \right\} \quad (II.4)$$

$$\{U_P\} = \left\{ \left\{ U_P \right\}^1, \left\{ U_P \right\}^2, \left\{ U_P \right\}^3 \dots \dots \left\{ U_P \right\}^m \right\} \quad (II.5)$$

$$\{F_I\} = \left\{ \left\{ F_I \right\}^1, \left\{ F_I \right\}^2, \left\{ F_I \right\}^3 \dots \dots \left\{ F_I \right\}^m \right\} \quad (II.6)$$

$$\{F_P\} = \left\{ \left\{ F_P \right\}^1, \left\{ F_P \right\}^2, \left\{ F_P \right\}^3 \dots \dots \left\{ F_P \right\}^m \right\} \quad (II.7)$$

O sistema de equações local para cada subestrutura pode ser escrito, quando ordenado convenientemente, de acordo com a expressão:

$$\begin{bmatrix} K_{II} & K_{IP} \\ K_{IP}^T & K_{PP} \end{bmatrix}^i \cdot \begin{Bmatrix} U_I \\ U_P \end{Bmatrix}^i = \begin{Bmatrix} F_I \\ F_P \end{Bmatrix}^i \quad (II.8)$$



$$\tilde{K}_{IP}^T \cdot U_I + \tilde{K}_{PP} \cdot U_P = \tilde{F}_P \quad (II.11)$$

isolando-se  $U_I$  na equação (II.10) e substituindo em (II.11), tem-se:

$$\left( \tilde{K}_{PP} - \tilde{K}_{IP}^T \cdot \tilde{K}_{II}^{-1} \cdot \tilde{K}_{IP} \right) \cdot \left( U_P \right) = \tilde{F}_P - \tilde{K}_{IP}^T \cdot \tilde{K}_{II}^{-1} \cdot \tilde{F}_I$$

$$(II.12)$$

fazendo,

$$\tilde{K}_G^i = \tilde{K}_{PP} - \tilde{K}_{IP}^T \cdot \tilde{K}_{II}^{-1} \cdot \tilde{K}_{IP} \quad (II.13)$$

$$\tilde{F}_G^i = \tilde{F}_P - \tilde{K}_{IP}^T \cdot \tilde{K}_{II}^{-1} \cdot \tilde{F}_I \quad (II.14)$$

onde,

$\tilde{K}_G^i$  → matriz de rigidez local condensada (matriz do superelemento ligado a subestrutura i)

$\tilde{F}_G^i$  → vetor de cargas local condensado (vetor de cargas do superelemento ligado a subestrutura i)

A geração da matriz  $\tilde{K}_G^i$  e do vetor  $\tilde{F}_G^i$  são obtidos, na prática, através da fatoração da matriz  $\tilde{K}_{II}$ , de acordo com:

$$\tilde{K}_{II} = L_I \cdot D_I \cdot L_I^T \quad (II.15)$$

onde ,

$\tilde{L}_I^T \rightarrow$  matriz triangular superior

$\tilde{D}_I \rightarrow$  matriz diagonal

$\tilde{L}_I \rightarrow$  matriz triangular inferior

substituindo a equação (II.15) em (II.13) e em (II.14), resulta:

$$\tilde{K}_G^i = \tilde{K}_{PP} - \tilde{K}_{IP}^T \cdot \tilde{L}_I^{-T} \cdot \tilde{D}_I^{-1} \cdot \tilde{L}_I^{-1} \cdot \tilde{K}_{IP} \quad (\text{II.16})$$

$$\tilde{F}_G^i = \tilde{F}_P - \tilde{K}_{IP}^T \cdot \tilde{L}_I^{-T} \cdot \tilde{D}_I^{-1} \cdot \tilde{L}_I^{-1} \cdot \tilde{F}_I \quad (\text{II.17})$$

fazendo-se  $\tilde{R}_{IP} = \tilde{L}_I^{-1} \cdot \tilde{K}_{IP}$  (II.18)

, e substituindo a equação acima em (II.16) e (II.17), chega-se finalmente a:

$$\tilde{K}_G^i = \tilde{K}_{PP} - \tilde{R}_{IP}^T \cdot \tilde{D}_I^{-1} \cdot \tilde{R}_{IP} \quad (\text{II.19})$$

$$\tilde{F}_G^i = \tilde{F}_P - \tilde{R}_{IP}^T \cdot \tilde{D}_I^{-1} \cdot \tilde{L}_I^{-1} \cdot \tilde{F}_I \quad (\text{II.20})$$

Supondo que o domínio inicial foi particionado em  $m$  subdomínios, a matriz de rigidez global condensada seria, então, formada pelo somatório conveniente das contribuições geradas por cada subestrutura na forma de seus superlementos:

$$\tilde{K}_S = \sum_{i=1}^m \tilde{K}_G^i \quad (\text{II.21})$$

$$\tilde{F}_S = \sum_{i=1}^m \tilde{F}_G^i \quad (\text{II.22})$$

Depois de formada a matriz de rigidez condensada  $K_s$ , e o vetor  $F_s$ , resolve-se o sistema :

$$\left[ K_s \right] \cdot \left\{ U_s \right\} = \left\{ F_s \right\} \quad (II.23)$$

onde,

$\left\{ U_s \right\}$  → vetor de deslocamentos para todos os nós de interface da estrutura

Resolvendo-se o sistema de equações (II.23) ,obtem-se os valores dos deslocamentos correspondentes aos graus de liberdade das interfaces. O cálculo dos deslocamentos nos nós internos a cada subestrutura, recuperação de resultados, é efetuado a partir do isolamento do vetor  $U_I$  na equação (II.10) com o auxílio da relação (II.18), resultando:

$$U_I^i = L^{-T} \cdot D^{-1} \cdot \left( L^{-1} \cdot F_I - R_{IP} \cdot U_P \right) \quad (II.24)$$

Uma vez obtidos os deslocamentos em todos os nós, determina-se a seguir as tensões a que estão submetidos os elementos básicos.

O cálculo de deslocamentos para nós internos a cada subestrutura e a determinação das tensões nos seus elementos básicos são operações que serão efetuadas ou não dependendo das necessidades do usuário. Nota-se, pela equação (II.24), que elas são desacopladas, ou seja, podem ser operadas separadamente em seus subdomínios dependendo somente de se conhecer os deslocamentos em suas

interfaces.

## II.2 - CARACTERÍSTICAS PRINCIPAIS E APLICABILIDADES DO MÉTODO

Nos parágrafos seguintes são citadas as principais características do método a fim de demonstrar a versatilidade e potencialidade desta formulação.

O método de subestruturas permite que se simplifique o trabalho de geração e verificação dos dados de entrada. Pode-se considerar de forma independente as diversas partes estruturais gerando para cada uma delas um superelemento correspondente. Os superelementos assim formados podem ser condensados para formar um nível mais alto de superelemento (fig II.2).

As partes estruturais repetitivas podem ser selecionadas como superelementos e descritas apenas uma vez. As propriedades de rigidez podem ser calculadas uma única vez e armazenadas numa biblioteca de superelementos para uso posterior. Obtém-se, desta forma, uma economia sensível de área de armazenamento e de esforço computacional. (fig. II.3)

Quando se trabalha a nível de projeto, as modificações necessárias somente afetariam os superelementos associados a esta determinada região. Bastaria, portanto, identificar os superelementos afetados e alterá-los em função da nova configuração. A escolha criteriosa dos graus de liberdade a serem condensados permitirá ao projetista uma maior versatilidade na análise, pois se as alterações somente afetarem um número

mínimo de subdomínios , então o tempo para uma reanálise será cada vez menor (fig. II.4).

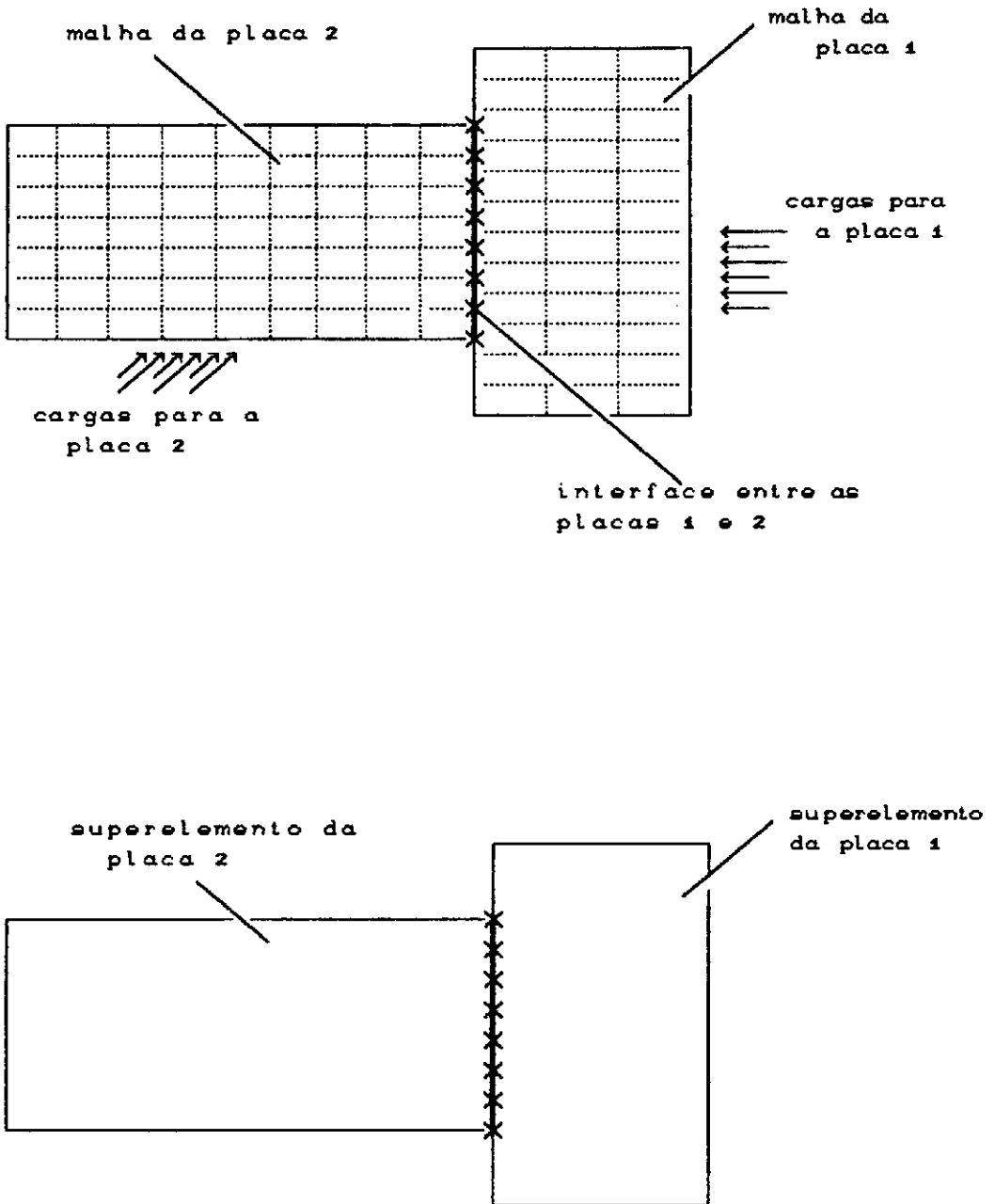
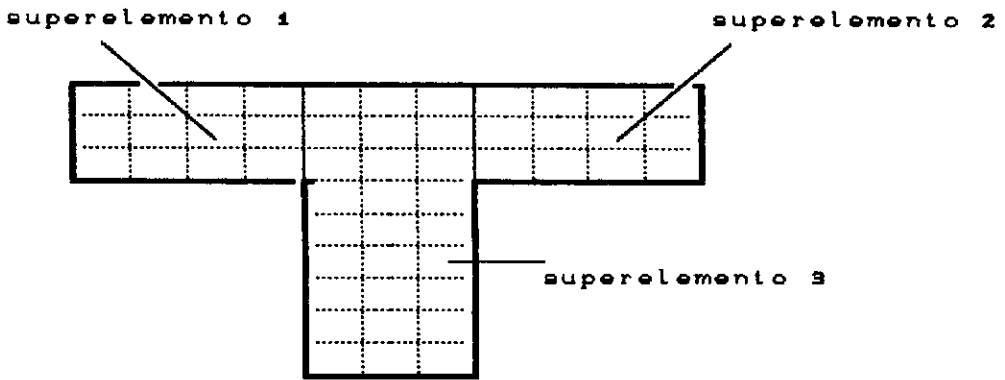


Figura II.2 - Indica a malha antes e depois da consideração de subestruturas



SEÇÃO COM OS SUPERELEMENTOS 1 E 2 SIMÉTRICOS

Figura II.3 - Simetria aplicada a técnica de subestruturas

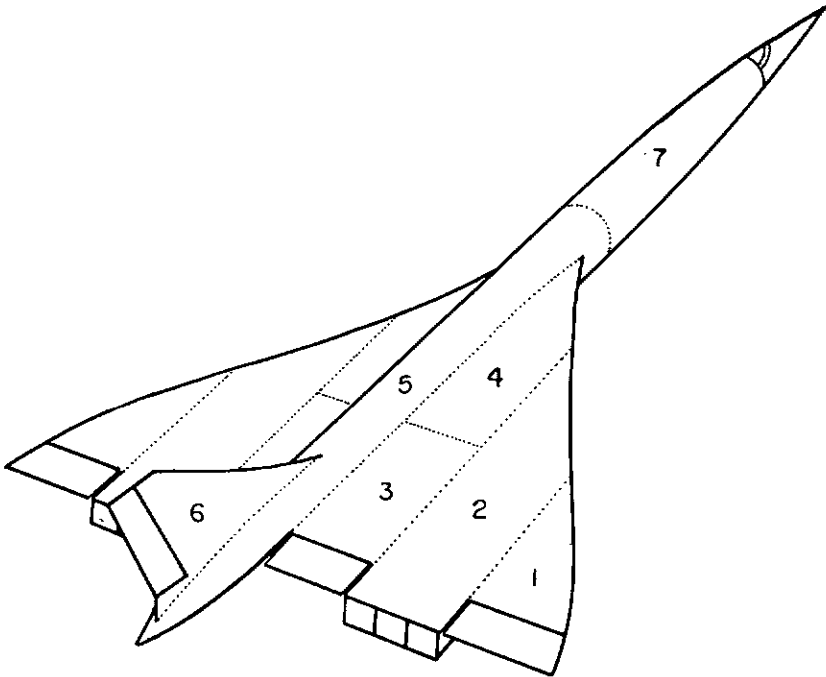


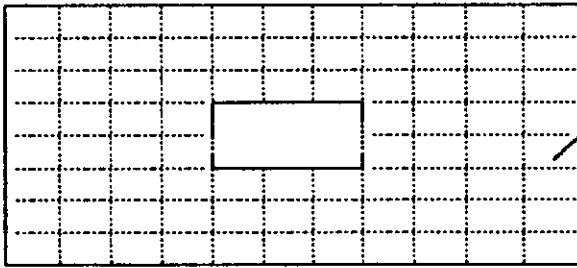
Figura II.4 - Exemplo de análise por divisão em subdomínios usando vários tipos de elementos estruturais

Desta forma é possível a abordagem de sistemas estruturais complexos, com um número muito grande de graus de liberdade. Problemas muito complexos podem ser sensivelmente simplificados, já que a resolução da equação (II.23) envolve um número muito menor de termos do que a

equação (II.1). O número de termos na equação (II.1) corresponde a todos os graus de liberdade da estrutura, enquanto que estão presentes nas equações (II.23) somente os graus de liberdade partilhados por duas ou mais subestruturas. Uma análise por subestruturas permite também que se opere com menos área disponível de memória real, podendo-se manter as variáveis mais constantemente na memória principal, com correspondente diminuição de tempo de I/O, e com isso aumentar a rapidez no acesso a estes valores (fig II.5).

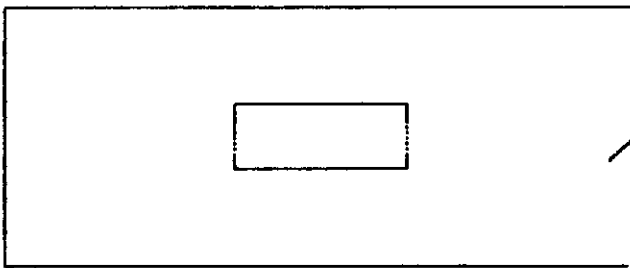
Uma das principais vantagens deste método é a possibilidade de adequação à programação em computadores de arquitetura paralela. A geração das matrizes de rigidez associadas a cada subestrutura, a condensação dos graus de liberdade internos e a posterior determinação das tensões nos elementos básicos, poderiam todas serem feitas de forma concorrente já que estão desacopladas e não necessitam de comunicação entre estas várias tarefas. A resolução do sistema de equações globais (II.23) também pode ser paralelizável, maiores detalhes em OMID [8] e FARHAT [6].

placa no estado plano com  
furo central



análise  
convencional  
(elementos finitos  
básicos)

número de equações = 104



análise com  
subestruturas  
(super elementos)

número de equações = 06

Figura II.5 - Influência da técnica de subestruturas  
na dimensão do sistema global de equações

## CAPÍTULO III

## EQUIPAMENTO E PERFORMANCE

## III.1 - COMPUTADORES PARALELOS

A idéia básica da computação paralela é a de fazer com que tarefas repetitivas sejam feitas de forma concorrente, isto é, fazer com que cada tarefa tenha um respectivo processador. Supondo que o tempo teórico de resolução fosse igual a  $t$  ( $t=t_1+t_2+t_3+\dots+t_n$ ), aplicando o paralelismo ele seria reduzido para  $t/n$ , onde  $n$  é o número de processadores. O alcance deste "speed-up" perfeito está condicionado ao grau de paralelismo do programa e a máquina que se utilizará, ver ORTEGA [10].

Uma diferença importante entre sistemas paralelos é o de como os processadores são programados, quais as suas capacidades de realizar tarefas diferentes simultaneamente.

Nos sistemas da classe SIMD (Single Instructions Multiple Data) todos os processadores paralelos são programados da mesma forma e, portanto, realizam ao mesmo tempo sempre as mesmas instruções. Cada processador opera uma corrente de instruções simples sobre uma corrente massa de dados a ele designada. Os primeiros sistemas concebidos pertenciam a este classe, e os computadores de processamento vetorial podem ser incluídos também se considerarmos que a vetorização acontece sob controle de "hardware".

Entretanto a maior parte dos computadores paralelos são da classe MIMD (Multiple Instructions Multiple Data).

Nesta classe, os processadores rodam programas próprios, o que proporciona ao programador uma maior flexibilidade na designação de tarefas. Tarefas diferentes requerem tempos diferentes de processamento, assim, enquanto que na classe SIMD o problema de sincronismo não existe a nível de programação, já na classe MIMD ele passa a ser fator de fundamental importância para a melhoria na eficiência do sistema.

Uma outra diferença importante é quanto à localização da memória. Existem dois tipos de computadores, os de memória localizada e os de memória compartilhada. Os processadores classificados como sendo de memória localizada (local memory) possuem associada a cada CPU uma capacidade de memória de uso exclusivo, a comunicação entre processadores é feita por envio e recebimento de mensagens através de "message passing". Já os de memória compartilhada (shared memory) têm a sua memória dividida em local e comum. A memória local seria usada para armazenar valores e informações desnecessárias a outros processadores, e a memória comum para locação e leitura de mensagens, ou seja, a comunicação entre processadores é feita por acesso não simultâneo a áreas comuns, ou de acesso comum a dois ou mais processadores. O tempo em que um processador espera para que um outro deixe livre certa área de acesso comum é chamado de tempo de contenção (contention time). O tempo de contenção cresce a medida que o número de processadores aumenta.

A forma como é feita a comunicação entre processadores é um dos aspectos mais importantes no estudo

dos computadores paralelos. Um computador paralelo se torna mais eficiente a medida em que o tempo gasto na comunicação for diminuindo.

Existem várias configurações para disposição dos processadores e também vários esquemas de comunicação associados a estas configurações, podemos citar:

a) Conexão completa (completely connected)- todos os processadores têm acesso a memória única, e por isso todos tem comunicação entre si. (fig III.1)

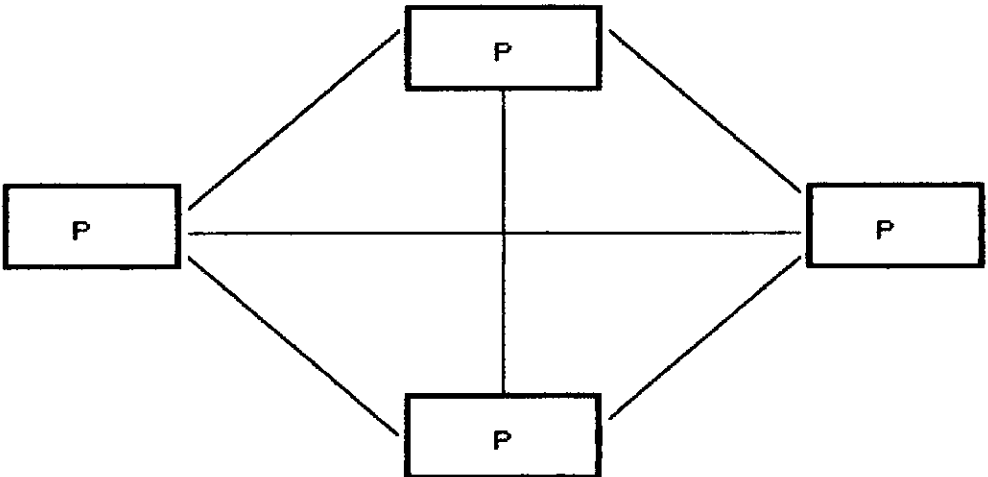


Figura III.1 - Configuração com conexão completa

b) Cruzamento de linhas (crossbar switch) - cada processador pode acessar a memória de um outro através das linhas que fazem a conexão (fig III.2).

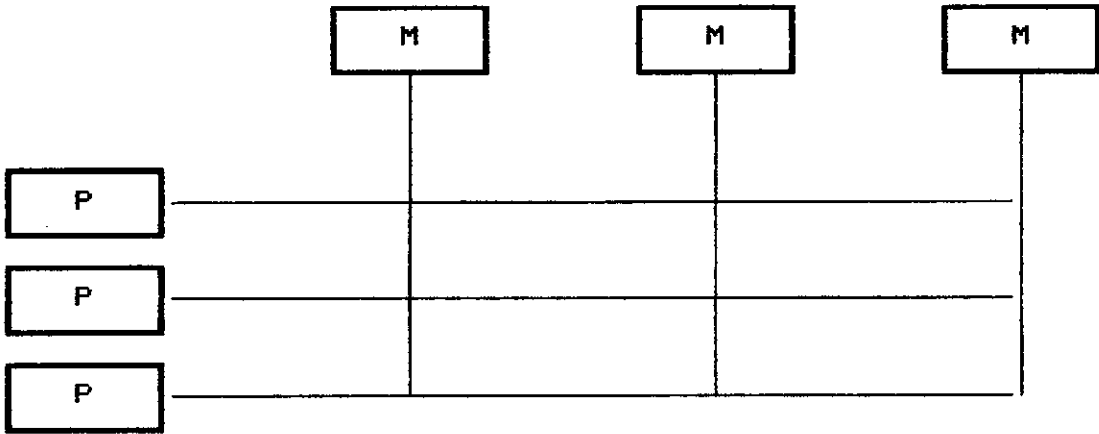


Figura III.2 - Configuração com cruzamento de linhas

c) "Bus" e "Ring" - todos os processadores se comunicam através de um "bus" de alta velocidade. (figuras III.3 e III.4)

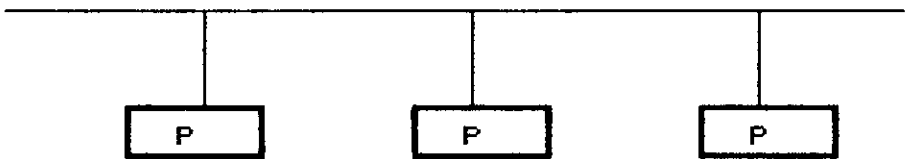


Figura III.3 - Configuração com utilização do "bus"

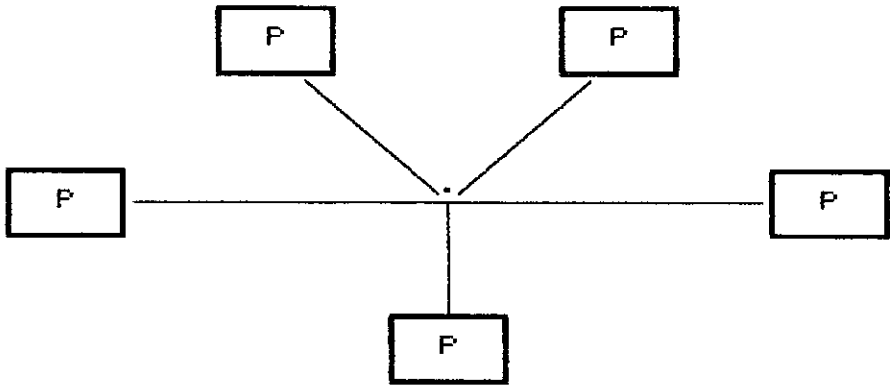


Figura III.4 - Configuração com utilização do "bus"

d) Conexão por malha (mesh connection) - (figura III.5)

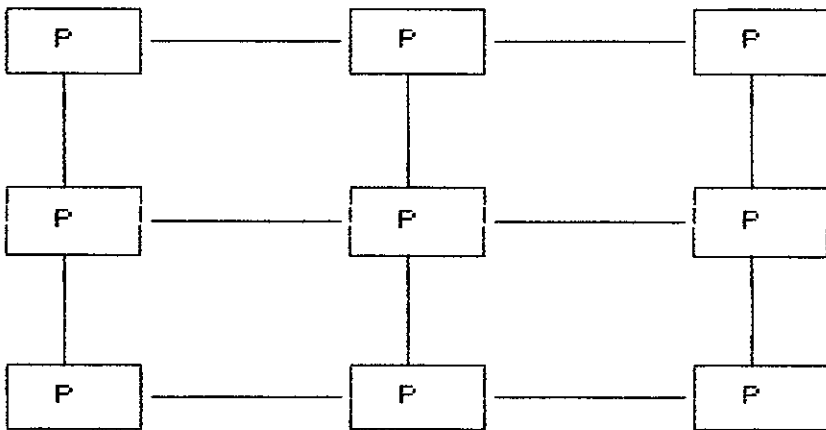


Figura III.5 - Configuração com conexão por malha

e) Hipercúbico (hipercube) - é uma extensão da conexão local para maiores dimensões. Cada processador é um vértice de um cubo e tem comunicação com  $k$  outros processadores. O valor de  $k$  depende do número de

processadores disponíveis, e é calculado através da relação:

$$n^{\circ} \text{ de processadores} = 2^k$$

A configuração de um hipercubo composto de oito processadores é mostrada na figura III.6

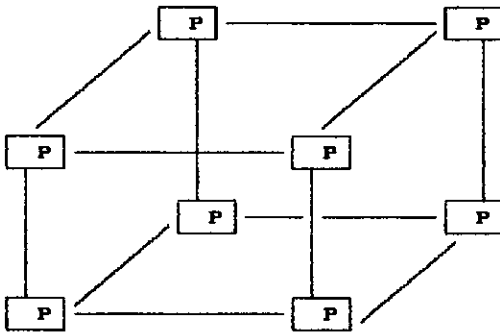


Figura III.6 - Hipercubo com 8 processadores

### III.2 - O SISTEMA A.C.P (ADVANCED COMPUTER PROGRAM)

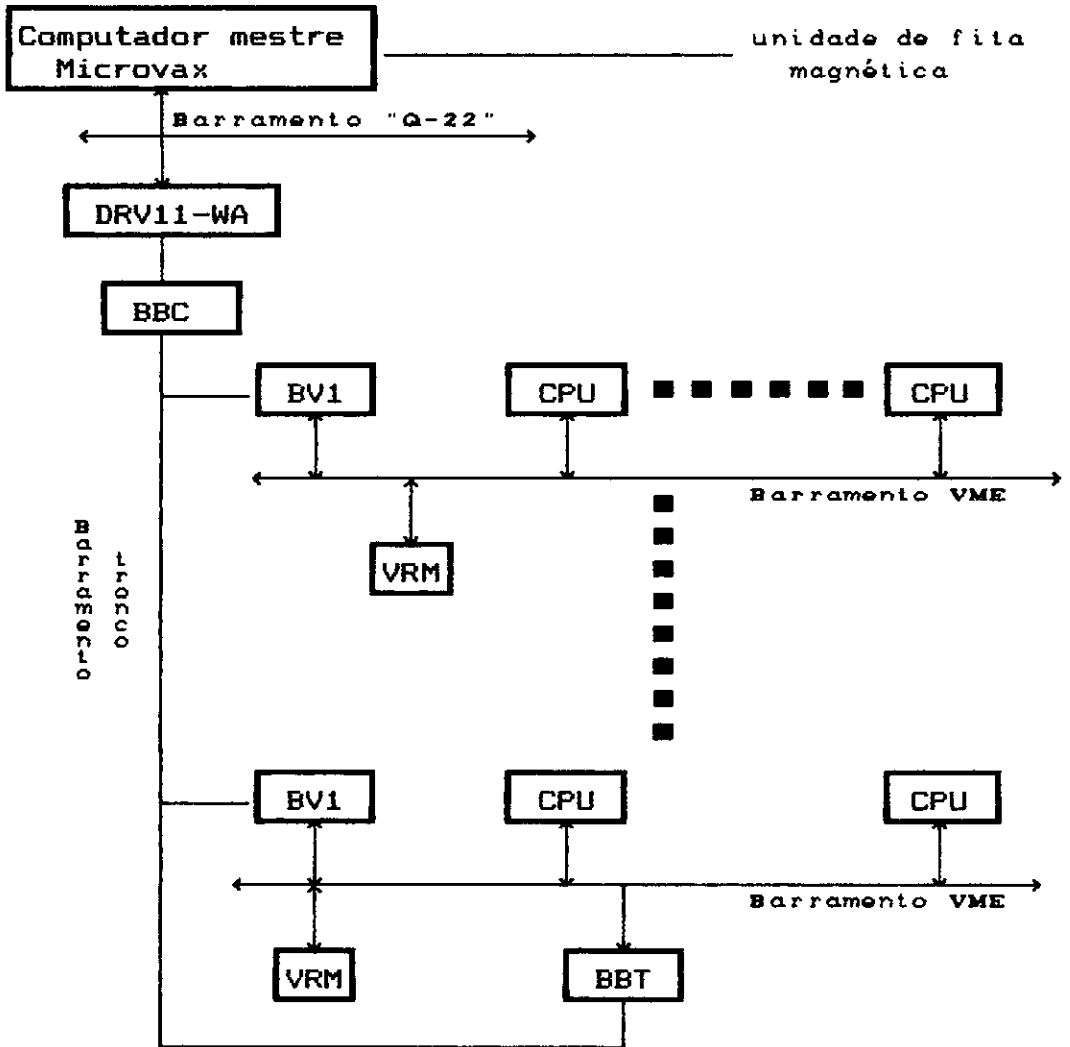
O sistema A.C.P (Advanced Computer Program) foi desenvolvido pela FERMILAB (Fermi National Accelerator Laboratory) para atender a necessidade de resolução de problemas com paralelismo explícito ou de estrutura claramente paralela.

Este tipo de problema também chamado de evento orientado, deve necessitar de muito tempo de CPU dos processadores, de poucas operações de entrada e saída de dados, e de não precisar de nenhuma ou quase nenhuma comunicação entre os processadores.

A primeira geração do ACP pode ser classificada como

sendo da classe MIMD (Multiple-Data Multiple-Instruction), de memória localizada e de paralelização explícita. A configuração do sistema está representada na figura III.7 retirada de BARROS [11].

O sistema A.C.P. , quando encarado a nível de "hardware" pode se encaixar na configuração do tipo "bus". Entretanto isto não impede que ele possa ser configurado em outras formas, seja logicamente dentro do programa ou até se alterando a disposição física de seus processadores.



#### LEGENDA

- BBC → Interface controladora do barramento tronco  
 BV1 → Interface do barramento tronco VME  
 DRV11-WA → Unidade de DMA  
 CPU → Nó de processamento  
 BBT → Terminal do barramento tronco  
 VRM → Árbitro do barramento VME

Figura III.7 - Configuração do sistema A.C.P.

O sistema é do tipo mestre/escravo. É função do mestre ou hospedeiro gerenciar todas as operações de I/O (Input/Output, Entrada/Saída) dos processadores, e também realizar qualquer outro tipo de operação consistente com sistema operacional a que ele está submetido. O hospedeiro pode ser um MicroVAX ou um VAX 11/780.

O escravo ou nó tem como função única a tarefa de processar os eventos a ele designados. Por ser tipo MIMD, o sistema aceita que nós diferentes operem programas diferentes. O nó pode assumir separadamente quatro estados: READY, RUNNING, DONE e DEAD. (fig. III.8)

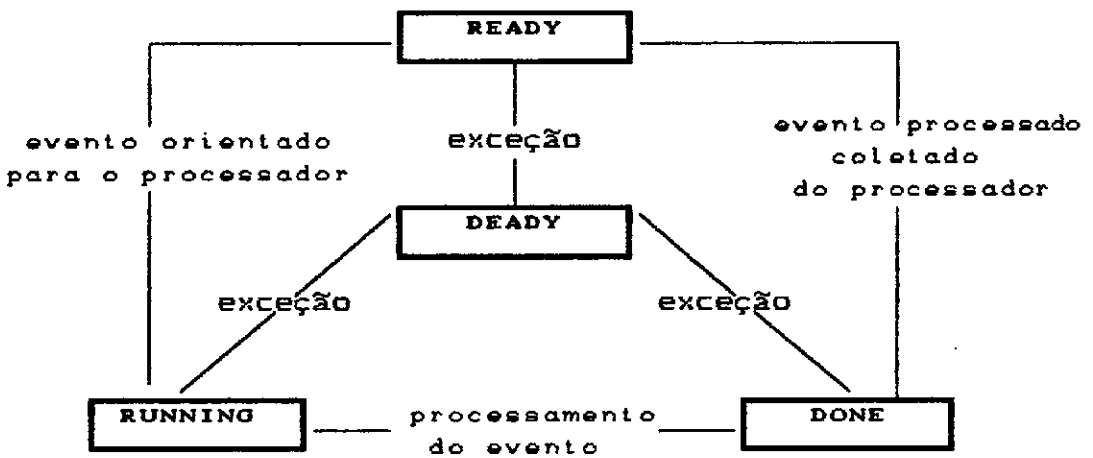


Figura III.8 - Sequências para análise de eventos pelos processadores

Cabe ao sistema operacional "LUNI" (Little Unix) a sincronização destes estados. Primeiramente o hospedeiro carrega o nó com o LUNI e o programa a ser executado por aquele nó. Como acabou de ser inicializado o LUNI o coloca

no estado READY, pronto para receber o evento a ser processado. Ao receber o evento, o processador passa para o estado RUNNING, executando o programa designado. Ao fim da execução, no estado DONE, os resultados estão prontos para serem coletados; completada a operação de envio de resultados o nó ou processador passa novamente para o estado READY. Caso ocorra algum erro ou exceção em qualquer destes estágios, o processador assume o estado DEAD, é retirado do sistema e espera uma nova reinicialização.

Cada processador é um modelo de microprocessador comercial de 32 bits/16MHz , MC 68020 ou um AT 32100 com seu respectivo coprocessador de ponto flutuante. A sua capacidade disponível de memória RAM (Random Access Memory) é de 2 Mbytes. A completa disposição dos componentes é dada na figura III.9 e encontrada em BARROS [11].

Um trabalho detalhado sobre as especificações do sistema A.C.P. pode ser encontrado em FISCHLER [12] e [13].

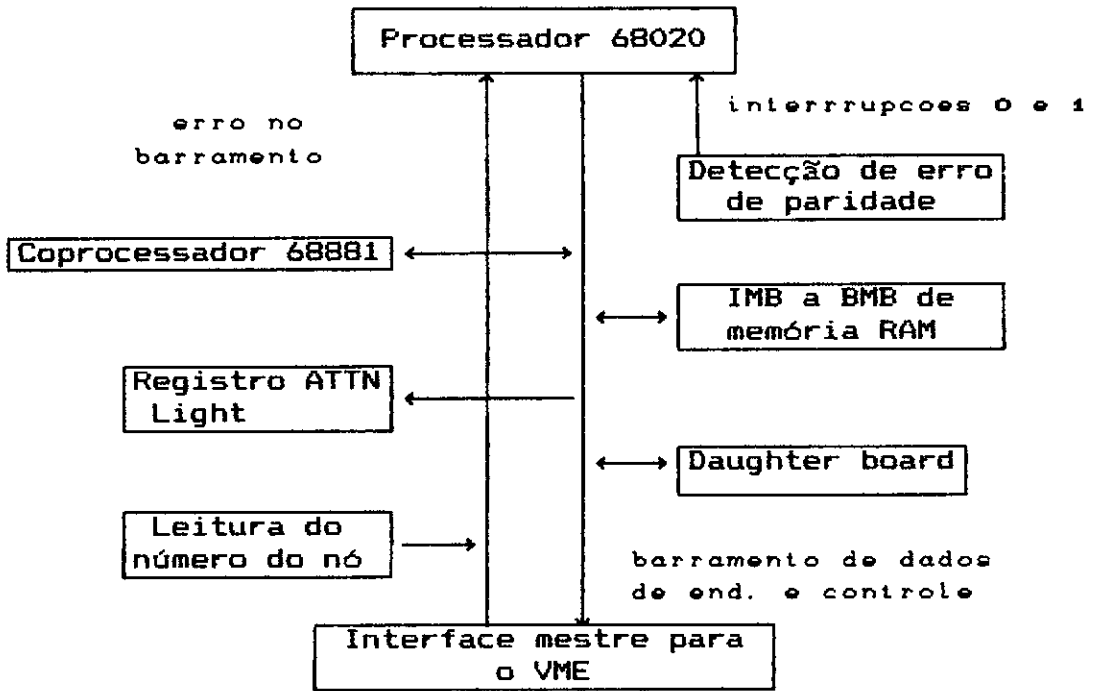


Figura III.9 - Disposição dos componentes dos processadores

A quantidade de informações que podem ser enviadas do hospedeiro para os nós ou vice-versa, não deve superar 200 Kbytes.

Acompanhando a máquina existe um suporte em "software" para auxiliar na programação de problemas do tipo evento orientado. O programador deve elaborar programas distintos, o que é executado pelo hospedeiro e o que é executado pelos nós. As rotinas de execução, seja para os nós seja para o hospedeiro, devem ser elaboradas em FORTRAN. A rotina de ligação, que se encarrega da transmissão de informações, é elaborada explorando todas as facilidades que o "software" de suporte coloca à disposição dos usuários. De modo a facilitar a depuração

de programas, existe um simulador de paralelismo que pode ser carregado em máquinas da DIGITAL de ambiente VMS (Virtual Memory System).

O C.B.P.F. (Centro Brasileiro de Pesquisas Físicas) têm instalado no laboratório de física experimental de altas energias um sistema A.C.F. de primeira geração composto de 21 processadores. O hospedeiro para o sistema é um MicroVAX operando num ambiente VMS.

### III.3 - AVALIAÇÃO DE PERFORMANCE

Um importante aspecto que deve ser analisado ao falarmos em computação paralela é a questão da performance. A correta adequação entre o programa e a máquina utilizada, assim como a avaliação de rendimento da máquina devem ser constantemente medidos para que se tenha sempre uma visão abrangente de todo o trabalho de programação. É necessário, então, obter formas de se medir esta performance, e para isto deve-se conhecer alguns conceitos básicos, ver ORTEGA [10].

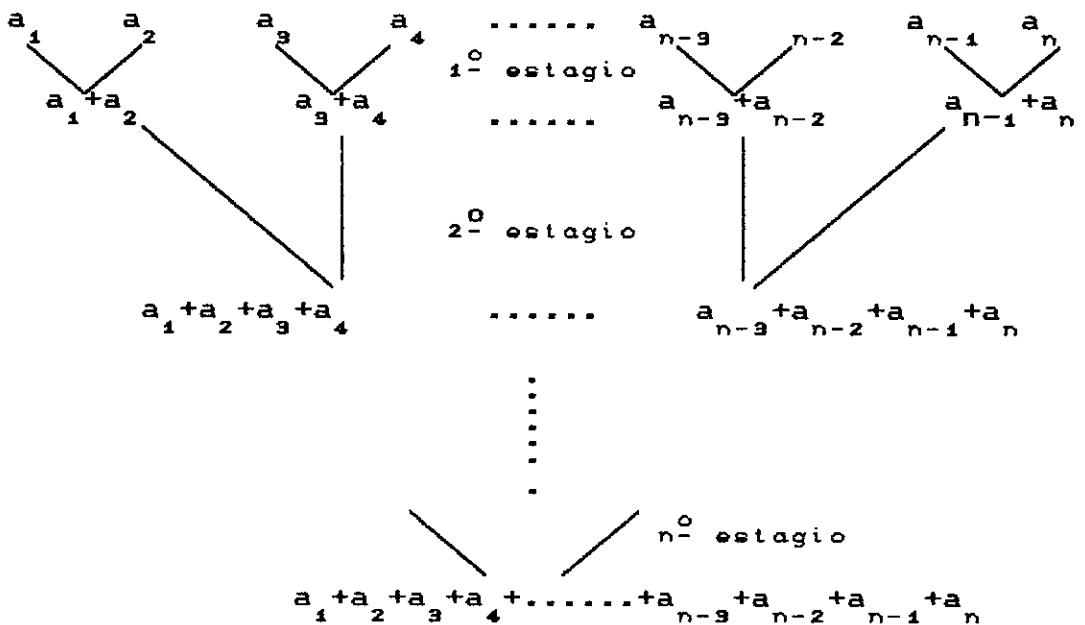
O primeiro conceito é o de grau de paralelismo. Ele é definido como sendo o número de operações que um algoritmo pode fazer em paralelo. Se, por exemplo, o nosso problema for a soma de dois vetores  $\underline{A}$  e  $\underline{B}$  de dimensão  $n$ , então o grau de paralelismo é  $n$ . Deve-se observar que o grau de paralelismo independe do número de processadores.

Grau médio de paralelismo de um algoritmo é o número de operações dividido pelo número de estágios necessários para realizá-las. O exemplo citado anteriormente tem o

grau médio igual ao grau normal, já que o número de estágios é 1. Se, no entanto, o nosso problema fosse somar  $n$  números  $(a_1, a_2, a_3, \dots, a_n)$  com  $n = 2^q$  números, então o número máximo de estágios seria  $q$ . O grau médio de paralelismo para este problema seria igual a soma dos graus de paralelismo de cada estágio dividido por  $q$ .

O número de somas que podem ser efetuadas no primeiro estágio é igual a  $n/2$ , no segundo é  $n/4$ , e assim por diante, até atingir o último estágio. Logo, temos:

- grau de paralelismo no 1<sup>o</sup> estágio =  $n/2$
- grau de paralelismo no 2<sup>o</sup> estágio =  $n/4$
- .....
- grau de paralelismo no  $n^o$  estágio = 1



$$\text{grau médio de paralelismo} = \frac{1}{q} ( n/2 + n/4 + \dots + 1 )$$

Relacionado ao conceito de grau de paralelismo existe o de granulometria. Alta granulometria é quando um número grande de tarefas podem ser feitas de forma independente, e baixa granulometria quando o número de tarefas é pequeno. O exemplo dado, de soma de dois vetores, é um caso de baixa granulometria, já que o número de operações se resume a soma de dois escalares.

A medida mais comumente usada quando queremos avaliar uma performance é a de "speedup". Existem dois tipos de "speedup".

O primeiro é definido como sendo o tempo de execução usando um único processador dividido pelo tempo de execução usando  $p$  processadores (equação III.1).

$$S_p = \frac{\text{tempo de execução com 1 processador}}{\text{tempo de execução com } p \text{ processadores}} \quad (\text{III.1})$$

O segundo "speedup" diz respeito a eficiência do algoritmo paralelo, e é tirado da relação entre o tempo de execução usando um único processador operando sob um algoritmo sequencial dividido pelo tempo de execução do algoritmo paralelo usando  $p$  processadores (equação III.2).

$$S'_p = \frac{\text{tempo de exec. com um proc. usando algor. sequencial}}{\text{tempo de exec. com } p \text{ proc. usando algor. paralelo}}$$

(III.2)

A relação  $S'_p$  é para muitos problemas difícil de ser obtida, pois nem todo problema paralelizável existe na forma sequencial, e vice-versa. Certos problemas exigem um alto custo na sua elaboração e por isto torna-se inviável investir em se programar adotando-se duas filosofias

diferentes de programação.

A "eficiência" do algoritmo paralelo sobre ele mesmo é definido pela equação (III.3).

$$E_p = \frac{S_p}{P} \quad (\text{III.3})$$

A "eficiência" do algoritmo paralelo sobre o sequencial é dada pela equação (III.4).

$$E'_p = \frac{S'_p}{P} \quad (\text{III.4})$$

Muitos fatores podem influir no tempo de execução de um programa em paralelo e, portanto, degenerar um "speedup" perfeito. A falta de um perfeito grau de paralelismo no algoritmo e o desbalanceamento nas cargas de trabalho de cada processador seriam fatores ligados a formulação do problema. O tempo gasto na comunicação de dados, na contenção e na sincronização estariam mais relacionados ao tipo de máquina em se opera.

"Data ready time" é o atraso causado pela comunicação, contenção e sincronização dos dados.

Um modelo formal de "speedup" é o que leva em conta todos estes fatores, sendo definido pela equação (III.5), retirada de ORTEGA [10].

$$S_p = \frac{T_1}{(\alpha_1 + \alpha_2/k + \alpha_3/p) \cdot T_1 + t_d} \quad (\text{III.5})$$

onde,

$T_1$  → tempo de execução usando 1 processador

$\alpha_1$  → fração de operações realizadas com um processador

$\alpha_2$  → fração de operações realizadas com grau médio de paralelismo  $k$

$\alpha_3$  → fração de operações realizadas com grau de paralelismo  $p$

$t_d$  → "data ready time"

O caso mais importante a ser discutido para a equação (III.5) é quando temos  $\alpha_2 = 0$ ,  $t_d = 0$ ,  $\alpha_1 = \alpha$ ,  $\alpha_3 = 1 - \alpha$ . A equação (III.5) toma, então, a forma,

$$S_p = \frac{1}{\alpha + (1 - \alpha) / p} \quad (\text{III.6})$$

conhecida como lei de "Amdahl".

Deve-se observar na equação (III.6) que mais importante que o número de processadores  $p$ , é o número de operações do programa que podem ser feitas em paralelo. Uma boa formulação em paralelo é obtida com o valor de  $\alpha$  mais próximo possível de 0 (zero), daí por diante basta crescer o número de processadores. Por outro lado, se o valor de  $\alpha$  for alto, de nada adianta crescer com o valor de  $p$ . Supondo que  $\alpha = 50\% = 1/2$ , então a relação (III.6) fica,

$$S_p = \frac{2}{(1 + 1/p)} < 2$$

ou seja, mesmo crescendo com  $p$ , o "speedup" jamais passaria de 2.

## CAPITULO IV

## ESTRUTURA DE PROGRAMAÇÃO

## IV.1 - CONSIDERAÇÕES GERAIS

Uma das maiores dificuldades encontradas para se operar com grandes estruturas está na complexidade da massa de dados. Existe uma quantidade muito grande de informações a serem fornecidas, tais como: coordenadas nodais, tipos de materiais, incidências nos elementos, cargas nodais, etc.

Um dos objetivos do programa desenvolvido para partição automática do domínio é o de tentar reduzir ao mínimo a interferência do usuário, ou seja, tentar automatizar todo trabalho de manipulação destes dados.

Um outro objetivo muito importante é o de se tentar balancear as cargas de trabalho para cada processador. Neste sentido, toma-se a carga de trabalho (número de operações) como proporcional ao número de elementos ligados àquele subdomínio e ao número de graus de liberdade de interface.

Sabendo-se que quanto maior o número de elementos, maior se torna o tempo necessário para a formação da matriz de rigidez completa da subestrutura (equação II.8). O programa adota, na medida do possível, uma quantidade de elementos igual para todos os subdomínios.

O número de operações necessárias para a condensação envolvidas nas equações II.13 e II.14 vai depender do

número de graus de liberdade internos e de interface. Neste sentido, deve-se sempre particionar a estrutura de forma que todos os subdomínios tenham aproximadamente o mesmo número de incógnitas, sejam elas internas ou de interface.

Uma das principais causas de degradação dos programas implementados em computadores paralelos é a necessidade de muita transmissão de informações. Para que se diminua o número de informações transmitidas pode-se aumentar o número de subdomínios para que cada qual tenha um tamanho menor e/ou diminuir o número de graus de liberdade de interface. O pré-processador tenta fazer com que estas condições sejam atingidas.

Ambos os programas, pré-processador e processador foram feitos usando uma estrutura modular de programação, o que lhes dá grande versatilidade.

#### IV.2 - PARTIÇÃO AUTOMÁTICA DO DOMÍNIO

O programa pré-processador, responsável pela partição automática do domínio em subdomínios, foi feito baseando-se no trabalho apresentado por FARHAT [11], e é resumido a seguir:

## Estrutura de dados

MN,MP → o vetor MN contém a conectividade entre elementos, é gerado na forma de pilha tendo como apontador o vetor MP.

NE,NP → o vetor NE armazena a incidência nó-elemento, ou seja, quais elementos estão ligados a determinado nó. Ele é gerado na forma de pilha e o seu apontador é o vetor NP.

NW → vetor contendo o peso nodal, ou seja, número de elementos incidentes em cada nó. É formado simultaneamente com os vetores NE e NP.

ME → vetor que armazenará os elementos pertencentes a cada subdomínio, gerados pelo algoritmo.

NINT,NINF → vetores que irão conter, respectivamente, os nós internos e de interface de cada subdomínio.

NNP → número de subdomínios desejado

L → número de elementos em cada subdomínio.

$$L = (\text{Número de elementos}) / (\text{número de subdomínios})$$

## ALGORITMO

- faça  $i = 1, NNP$

(a) localiza-se o nó "n" com peso mínimo associado.

(b) determinam-se todos os elementos ligados a este nó. Eles iniciam a lista dos elementos pertencentes ao subdomínio i.

(c) acrescentem-se de forma recursiva nesta lista, os elementos vizinhos aos previamente determinados. Vão sendo adicionados elementos até que se complete os L desejados.

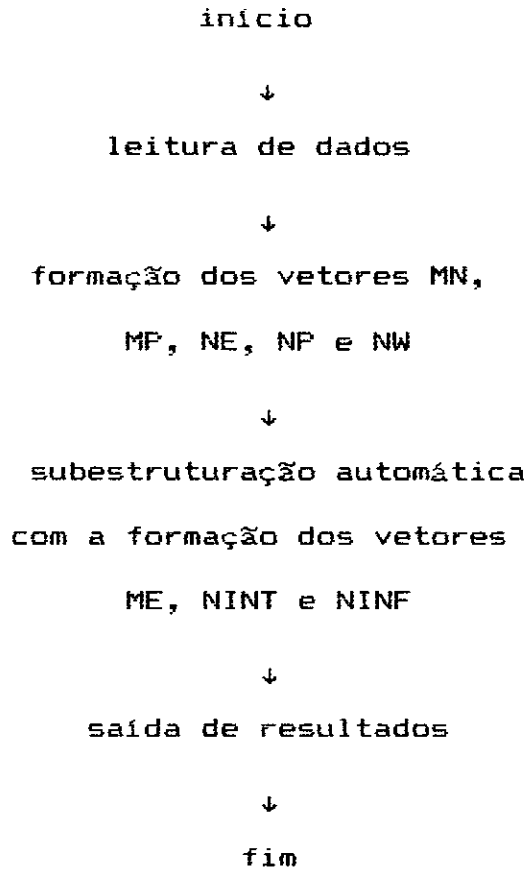
(d) cada vez que um elemento for considerado como pertencente ao subdomínio, reduzem-se os pesos dos nós incidentes neste elemento.

(e) removem-se todos os elementos considerados pertencentes ao i-ésimo subdomínio do sistema.

(f) identificam-se, então, os nós internos e de interface associados ao i-ésimo subdomínio. Os nós pertencentes ao subdomínio são denominados nós de interface quando possuem peso nodal diferente de zero, e nós internos quando possuem peso nodal igual a zero.

## FLUXOGRAMA

(Partição automática do domínio)



### IV.3 - SUBESTRUTURAÇÃO

O programa desenvolvido para análise por subestruturas foi feito obedecendo ao fluxograma básico que é dado a seguir :

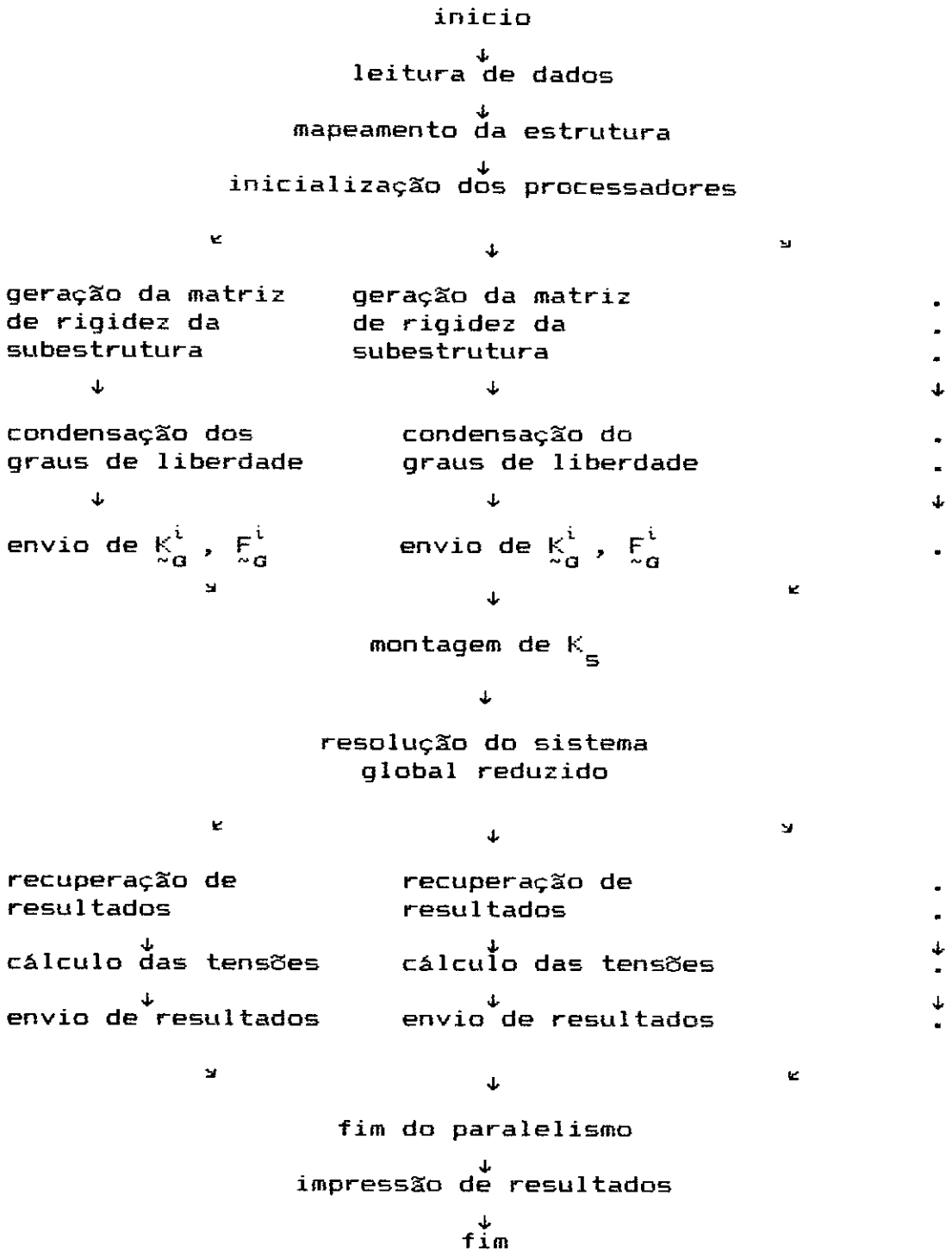


É importante observar no fluxograma básico que o cálculo das tensões nos elementos e o dos deslocamentos nos nós internos é feito no hospedeiro, de forma sequencial.

Esta forma de agir faz com que seja necessário a transmissão do "node" para o "host" da matriz de rigidez completa da subestrutura e não somente da matriz de rigidez do superelemento. Os dados a serem enviados são a matriz de rigidez interna decomposta (equação II.15), a matriz  $R_{IP}$  (equação II.18), o vetor resultante da operação  $L_I^{-1} \cdot F_I$  que ocupa a mesma área de memória do vetor  $F_I$ , e a matriz condensada (equações II.19 e II.20).

A justificativa para esta atitude está no fato de que os processadores (nós) não possuem acesso a unidades de disco e fita. Para que se possa calcular as tensões de forma paralela é necessário armazenar os valores das matrizes nos nós de modo que eles possam ser acessados a qualquer instante. O fluxograma seria, então, alterado e ficaria na forma seguinte :

## FLUXOGRAMA II



Uma alternativa para se guardar estes valores seria tornar estas variáveis como do tipo compartilhadas. Uma variável do tipo local é de uso restrito ao evento. Já a do tipo compartilhada, armazenada em áreas de memória real (RAM) interna ao processador, podem ser acessadas por aquele ou outro processador a qualquer momento, desde que não seja simultaneamente.

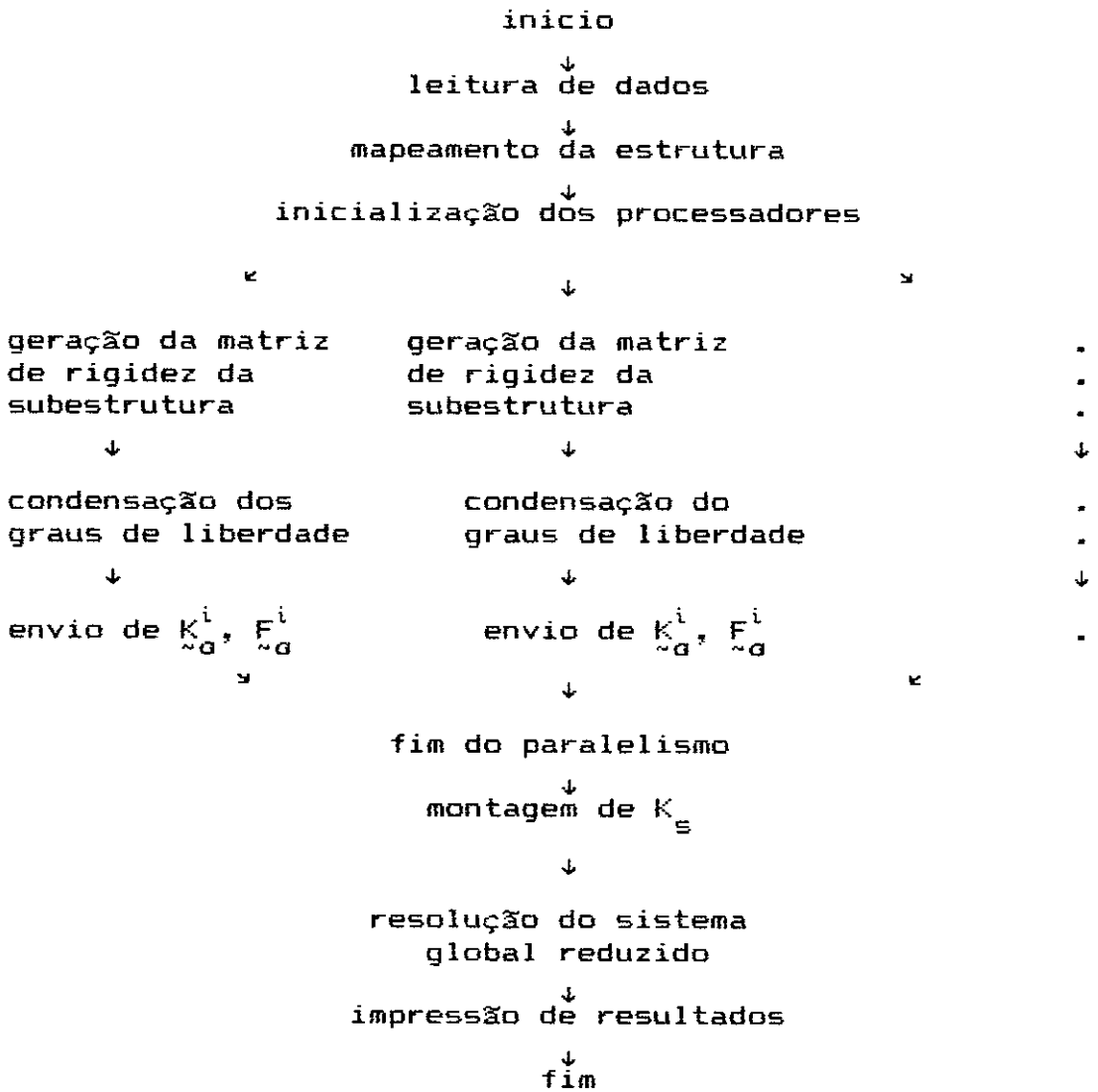
Já foi citado anteriormente, capítulo III, que a memória RAM disponível em cada processador é de 2 Mbytes. Se o fluxograma II fosse utilizado como opção, cada processador, ao operar mais de um evento e tendo de reservar obrigatoriamente parte da memória disponível para cada um destes eventos, eventualmente a memória seria insuficiente. O fato de um processador analisar mais de um evento, ocorre constantemente quando tem-se por objetivo do programa a pesquisa, e deseja-se determinar valores de "speedup" e eficiência. Também ocorrerá quando o número de eventos for maior que o número de processadores, ou seja, quando o problema for suficientemente grande para que aconteça de mesmo escolhendo o menor número possível de subdomínios este número ainda seja maior que o número de processadores disponíveis.

Os subdomínios devem ter um tamanho conveniente para que não esgotem a capacidade de memória do processador ao ser operado. Num mesmo problema sempre que diminuirmos o número de subdomínios estaremos aumentando o tamanho de cada um deles e, portanto, aumentando as suas necessidades em área de memória.

Com o objetivo de verificar o que ocorre ao transmitir

as equações II.19 e II.20, adaptou-se o programa para que fossem calculados somente os deslocamentos nos nós de interface. No fluxograma III, portanto, não são determinados deslocamentos nos nós internos e nem as tensões a que estão submetidos os elementos básicos. O fluxograma teria a forma:

## FLUXOGRAMA III



A técnica utilizada para introduzir as condições de contorno ao problema foi a de 1 e 0. O armazenamento da matriz de rigidez de cada subestrutura foi feito por alturas efetivas de coluna. As matrizes dos superelementos representadas pelas equações II.19 e II.20 foram tratadas como elementos básicos comuns, a nível de programação. Portanto, na montagem do sistema de equações global (eq. II.21 e II.22) são armazenadas como blocos (fig. IV.1).

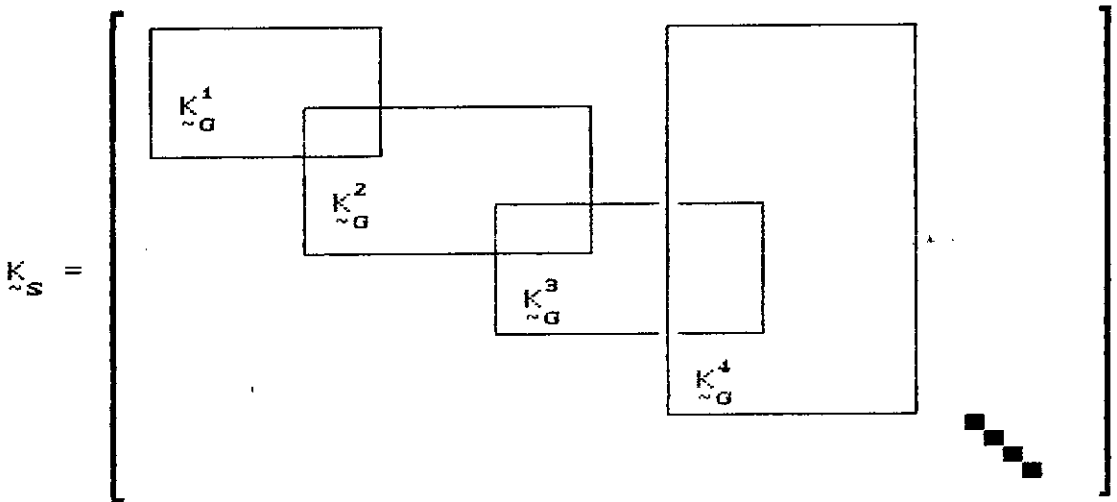


Figura IV.1 - Matrizes dos superelementos operando como blocos

A resolução do sistema de equações global (eq. II.23) foi feito memória central, segundo o procedimento abaixo:

$$\tilde{K}_S \cdot \tilde{U}_S = \tilde{F}_S \quad (\text{IV.1})$$

fatorando-se  $\tilde{K}_S$ , tem-se

$$\tilde{K}_S = \tilde{L} \cdot \tilde{D} \cdot \tilde{L}^T \quad (\text{IV.2})$$

onde ,

$\tilde{L}^T \rightarrow$  matriz triangular superior

$\tilde{D} \rightarrow$  matriz diagonal

$\tilde{L} \rightarrow$  matriz triangular inferior

faz-se,

$$\tilde{V} = \tilde{L}^{-1} \cdot \tilde{F}_S \quad (\text{IV.3})$$

e a partir daí,

$$\tilde{U}_S = \tilde{L}^{-T} \tilde{D} \cdot \tilde{V} \quad (\text{IV.4})$$

A análise de grandes estruturas requer uma maior precisão para o cálculo dos deslocamentos, pois o número de operações envolvidas na condensação pode causar uma instabilidade numérica no método ocasionada pela aritmética finita dos computadores. Dotou-se o programa de subestruturação da opção de operar em precisão simples ou dupla, conforme escolha do usuário. Vale resaltar que , no caso de se operar em precisão dupla, a área necessária para armazenar variáveis do tipo real é dobrada. Como consequência temos que será aumentada, num mesmo problema e usando um mesmo algoritmo, a quantidade de informações a serem transmitidas e recebidas pelos nós. Além disso, como a capacidade de memória RAM dos processadores é limitada a 2 Mbytes, o tamanho da subestrutura terá de ser diminuído em função, principalmente, da área ocupada pela matriz de rigidez da mesma.

## CAPÍTULO V

## APLICAÇÕES E RESULTADOS

A fim de demonstrar a adequação do algoritmo proposto para decomposição automática de domínio a qualquer tipo de elemento estrutural, são apresentados 2 exemplos. Eles são respectivamente compostos por elementos quadriláteros de 4 nós usados na análise de problemas de estado plano de tensões pelo método dos elementos finitos, e elementos reticulados de pórtico plano. Todos os resultados encontrados estão expostos no item V.1 deste capítulo.

O item V.2 é reservado a apresentação dos resultados encontrados pela aplicação dos fluxogramas I e III a vários exemplos. A composição dos dois fluxogramas foi apresentada no capítulo IV.

O fluxograma I foi aplicado a 5 estruturas e utilizou-se para estes exemplos toda a capacidade de transmitir e receber informações dos nós que é de 200 Kbytes, conforme citado no capítulo III.

O cálculo do número de dados enviados a cada processador foi feito através do controle do comprimento do vetor que faz a ligação entre o hospedeiro e os nós. O vetor que promove a ligação é tratado pelo programa como um dado do tipo "COMMON" por imposição da máquina, não sendo possível, portanto, variar a sua dimensão de um evento para outro. Conseqüentemente, se um determinado evento necessitasse de 50 Kbytes de dados para ser carregado, e o vetor de ligação tivesse 200 Kbytes, seriam

enviados 150 Kbytes de zeros. Isto ressalta a necessidade de se ter eventos (subdomínios) balanceados de forma que o comprimento do vetor não seja muito alterado.

Para aplicar o fluxograma III foram escolhidos exemplos que utilizassem os 2 Mbytes de memória disponível de cada processador, elevando, desta forma, ao máximo o tempo de processamento nos nós. Como o fluxograma III pode operar em precisão simples e dupla, foram escolhidos 5 exemplos que para serem executados nas duas versões. Para completar a lista de exemplos, o fluxograma III foi executado em um problema dotado de 6 subdomínios. Este sexto exemplo só pôde ser executado em precisão dupla. Como forma de simplificar a descrição dos vários exemplos são apresentadas tabelas que resumem as informações sobre toda a estrutura. Nestas tabelas são definidos parâmetros cujos significados são indicados a seguir:

número de processadores .....	np
tempo p/ leitura e impressão dos dados de entrada .....	t1
tempo p/ cálculo dos ponteiros .....	t2
tempo p/ inicialização dos processadores .....	t3
tempo de montagem da matriz de rigidez $K_s$ .....	t4
tempo de resolução do sistema .....	t5
tempo para cálculo das tensões nos elementos básicos e impre- ssão de resultados .....	t6
tempo total .....	tt
<u>tempo t4 com n processadores</u> tempo t4 com 1 processador .....	sp-1

$\frac{\text{tempo tt com n processadores}}{\text{tempo tt com 1 processador}}$	.....	sp-2
$\frac{\text{sp-1}}{\text{np}}$	.....	ef-1
$\frac{\text{sp-2}}{\text{np}}$	.....	ef-2
altura da estrutura	.....	h
largura da estrutura	.....	b
largura entre dois vãos consecutivos na base	.....	vao b
largura entre dois vãos consecutivos na altura	.....	vao h
número de vãos na altura	.....	nh
número de vãos na base	.....	nb

A unidade de medida de tempo adotada foi a de segundos. Todos os tempos medidos são do tipo "elapsed time", a não ser alguns casos que serão referenciados de maneira explícita. Tomou-se o "elapsed time" como medida por entender-se ser este o tempo "real" de resolução do problema. A performance de um sistema depende do rendimento de todas as peças que o compõe, e não somente do tempo de C.P.U., por exemplo. É claro que sendo o sistema VAX multi-usuário, o tempo do programa vai

dependem do número de usuários no sistema. Assim, toda vez que se quis obter resultados livres de qualquer interferência operou-se sozinho, ou seja, a máquina toda a disposição de um único usuário.

O tempo classificado como de inicialização dos processadores é o tempo que leva o sistema para carregar os nós com o programa a ser executado e com o sistema operacional "LUNI". Este tempo varia muito pouco se temos 1 ou 6 processadores. Os exemplos dados no item V.2 demonstram este fato.

Todos os exemplos analisados pelo fluxograma I e III tem a sua forma geral semelhante a figura V.1. O que se tentou fazer é trabalhar com modelos em que se pudesse controlar mais facilmente o número de elementos em cada subdomínio e, através disso, cada vez que se quisesse aumentar o problema para um número maior de subdomínios bastaria que aumentássemos o número de vãos na altura de forma que o número de elementos total da estrutura fosse um múltiplo do número de elementos desejado para cada subdomínio.

Observa-se também, que neste caso, se for somente aumentada a altura do modelo, cada subdomínio terá no máximo dois vizinhos. O número de nós de interface aumenta proporcionalmente com o número de subdomínios, e assim, consegue-se aumentar o número de nós internos e elementos sem alterar o de nós de interface. A importância destas facilidades é mostrada a medida em que os exemplos dados no item V.2 vão sendo apresentados e discutidos.

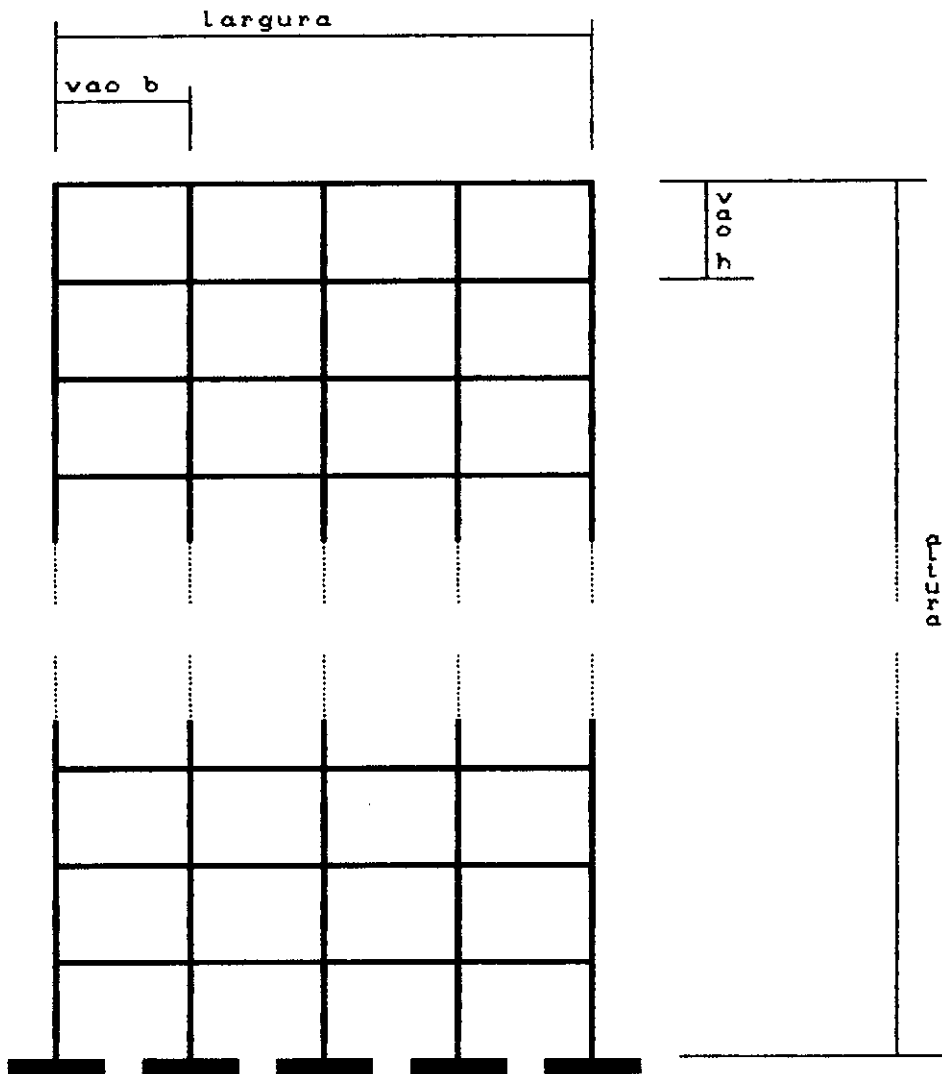


Figura V.1 - Configuração geral adotada para todos os exemplos

#### V.1 - EXEMPLOS DE PARTIÇÃO AUTOMÁTICA DO DOMÍNIO

O primeiro exemplo de decomposição de domínio é o de uma placa submetida a um estado plano de tensões e discretizada com 64 elementos finitos quadriláteros de 4 nós. O número total de nós para este modelo é de 81 nós

dispostos conforme a figura V.2 . A numeração dos elementos é mostrada na figura V.3 .

1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18
19	20	21	22	23	24	25	26	27
64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81

Figura V.2 - Numeração dos nós na placa indicada

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
							32
							40
							48
							56
57	58	59	60	61	62	63	64

Figura V.3 - Numeração dos elementos na placa indicada

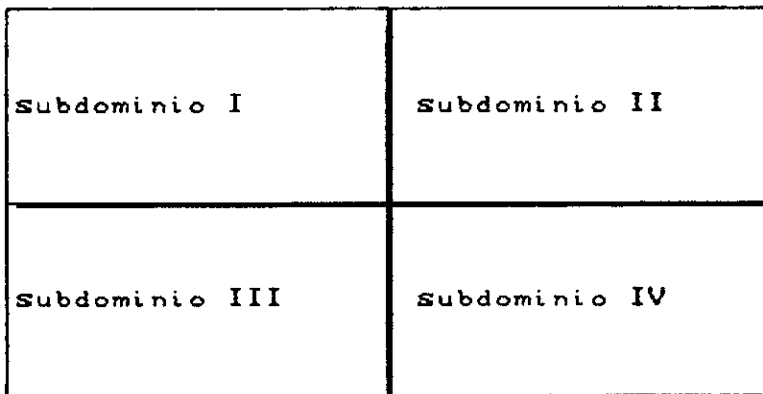
A aplicação do algoritmo de subestruturação automática neste exemplo foi feito variando o número de subdomínios de 2 a 5. Os resultados encontram-se na tabela V.1.

NUMERO DE SUBDOMINIOS	NUMERO DE NÓS DE INTERFACE
2	13
3	19
4	17
5	23

Tabela V.1 - Número de subdomínios para as diversas decomposições

É importante notar através deste exemplo que o aumento no número de subdomínios não significa necessariamente um aumento do número de nós de interface.

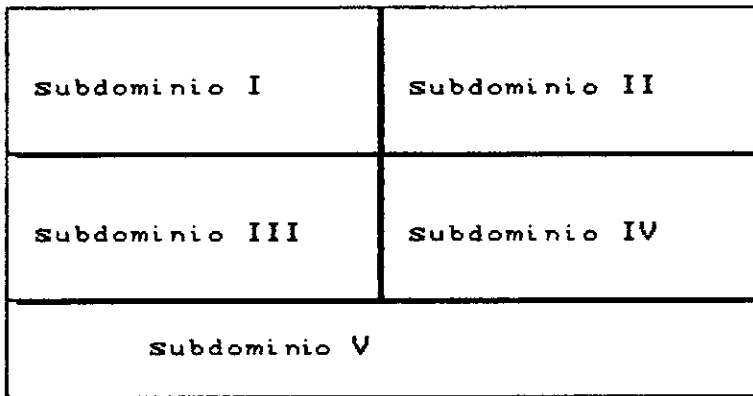
As figuras V.4 e V.5 apresentam as interfaces obtidas com a decomposição com 4 e 5 subdomínios.



—— interface

—— contorno

Figura V.4 - Decomposição em 4 subdomínios



—— interface      —— contorno

Figura V.5 - Decomposição em 5 subdomínios

O segundo caso estudado corresponde a decomposição da torre da figura V.3, composta de 39 nós e 60 elementos de pórtico plano.

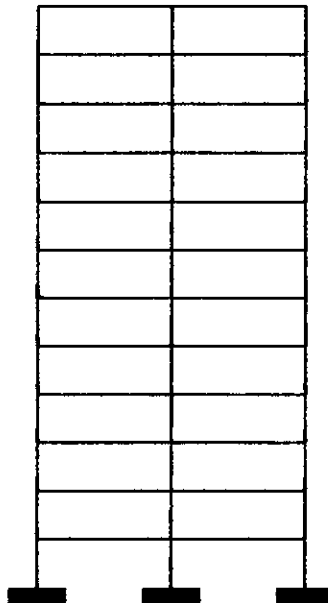


Figura V.6 - Estrutura composta de elementos de pórtico

A tabela V.2 indica o número de nós de interface para as diversas decomposições efetuadas com o algoritmo

proposto.

NUMERO DE SUBDOMINIOS	NUMERO DE NÓS DE INTERFACE
2	3
3	6
4	9
5	12
6	13
7	16
8	16
9	18
10	21

Tabela V.2

As figuras V.7 e V.8 indicam os elementos e a disposição de cada subestrutura encontradas pela decomposição em 3 e 5 subdomínios, respectivamente.

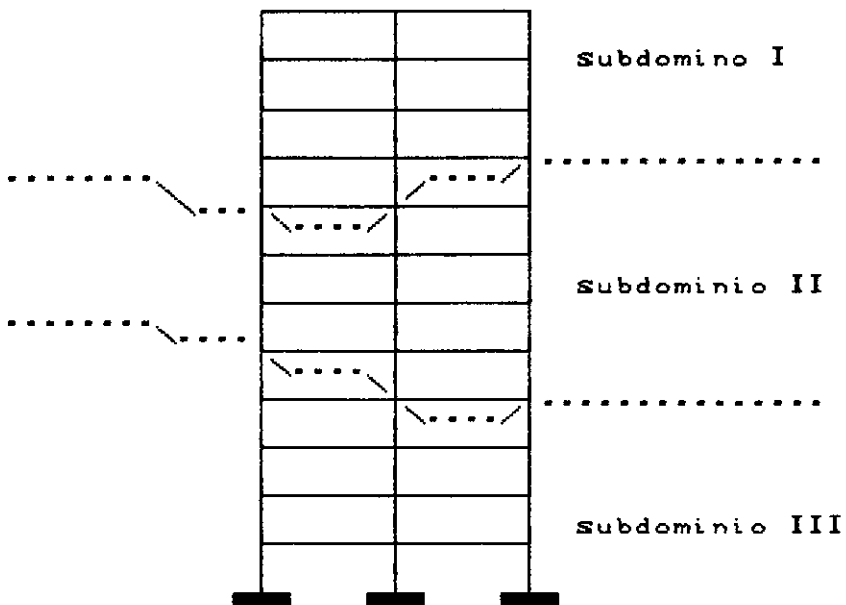


Figura V.7 - Decomposição em 3 subdomínios

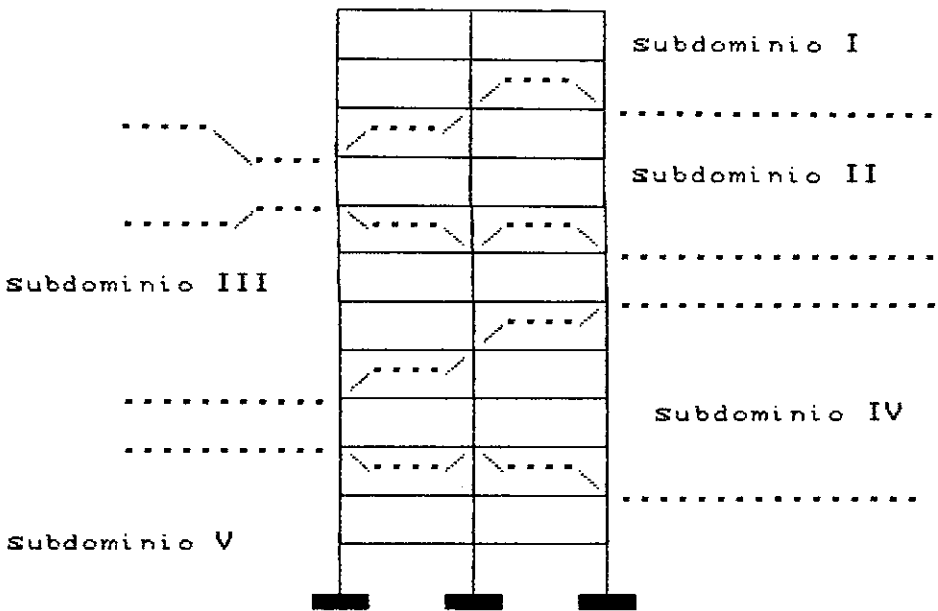


Figura V.8 – Decomposição em 5 subdomínios

Este segundo exemplo reafirma a tendência manifestada no primeiro. Ao mesmo tempo, ele permite afirmar que o algoritmo é melhor aplicável a estruturas que possuem uma dimensão preponderante sobre as demais. Estes tipos de modelos têm, pela sua própria geometria, a capacidade de antecipar os possíveis resultados da subestruturação, e assim propiciar um maior controle da divisão automática por parte do usuário.

## V.2 - EXEMPLOS DE ANÁLISE POR SUBESTRUTURAS

Os exemplos resolvidos através da aplicação do fluxograma I envolvem grande quantidade de informações a serem transmitidas do hospedeiro para os nós (processadores) e vice-versa. O tempo de comunicação, para estes exemplos, levou a uma degradação do índice de speedup e fez com que não se conseguisse em nenhum dos

exemplos operar com mais de 2 processadores.

O primeiro exemplo é de uma torre composta de 245 elementos e 144 nós. A configuração geral é mostrada pela figura V.1 tendo-se como parâmetros os valores descritos pela tabela V.3. O carregamento atuante é originário do vento e atua em toda a face lateral esquerda. Todas as cargas são consideradas aplicadas nos nós laterais, e tem valor igual a unidade.

número de nós .....	144
número de elementos .....	245
número de graus de liberdade do nó .....	3
número de nós do elemento .....	2
descrição = "elemento reticulado de pórtico plano"	
número de casos de carregamento .....	1
número de graus de liberdade da estrutura .....	420
número de subdomínios .....	2
altura (h).....	70 m
largura (b).....	9 m
vão b .....	3 m
vão h .....	2 m
nh .....	35
nb .....	3

Tabela V.3 - Descrição do modelo

Após a aplicação do algoritmo de decomposição de domínio para gerar 2 subdomínios, os resultados indicaram 4 nós de interface para todo o modelo. A dimensão do sistema de equações globais representado na equação (II.21) é igual a 12. Os valores encontrados na decomposição estão na indicados tabela V.4 .

Subdomínio	n <sup>o</sup> de nós internos	n <sup>o</sup> de nós de interface	n <sup>o</sup> de elementos
1	71	4	122
2	69	4	123

Tabela V.4 - Resultados da divisão automática

A análise desta estrutura foi feita segundo o fluxograma I , operando em precisão simples. Como forma de se medir a performance , o número de processadores usados variou de 1 a 2 . Os valores encontrados estão na tabela V.5 .

A resolução do exemplo utilizando somente um processador equivale a resolução de forma sequencial. O tempo encontrado de forma sequencial dividido pelo tempo de execução com n processadores vai fornecer o valor de speedup, conforme a relação III.1. A eficiência foi obtida a partir da relação III.3.

np	1	2
t1	5.10	5.09
t2	0.03	0.02
t3	15.15	14.56
t4	40.06	33.82
t5	0.02	0.02
t6	6.95	6.98
tt	67.31	60.50
sp-1	---	1.18
sp-2	---	1.11
ef-1	---	0.59
ef-2	---	0.56

Tabela V.5 - Análise por subestruturas com até 2 processadores

No segundo exemplo, mantendo aproximadamente o mesmo número de elementos em cada subdomínio, aumentou-se o número de subdomínios de 2 para 3. O carregamento teve a mesma origem que o do primeiro exemplo, ou seja, cargas de vento aplicadas diretamente nos nós laterais e de valor unitário. A descrição deste exemplo é dado na tabela V.6.

número de nós .....	212
número de elementos .....	364
número de graus de liberdade do nó .....	3
número de nós do elemento .....	2
descrição = "elemento reticulado de pórtico plano"	
número de casos de carregamento .....	1
número de graus de liberdade da estrutura .....	624
número de subdomínios .....	3
altura (h).....	104 m
largura (b).....	9 m
vão b .....	3 m
vão h .....	2 m
nh .....	52
nb .....	3

Tabela V.6 - Descrição do modelo

A partição do domínio inicial em três subdomínios ficou como relatado na tabela V.7 .

Subdomínio	n <sup>o</sup> de nós internos	n <sup>o</sup> de nós de interface	n <sup>o</sup> de elementos
1	70	4	121
2	65	8	121
3	69	4	122

Tabela V.7 - Resultados da divisão automática

O que se pode notar facilmente é o aumento dos nós de interface do modelo, o seu valor passou de 4 para 8. Assim sendo, o sistema de equações global também terá a sua dimensão dobrada, passando de 12 para 24 . Para este caso existem três subdomínios e por isso o número de processadores variou de 1 a 3. Os resultados encontrados na análise completa da estrutura pelo fluxograma I

encontram-se reunidos na tabela V.8 .

np	1	2	3
t1	6.84	6.92	6.87
t2	0.04	0.05	0.04
t3	14.72	14.65	14.51
t4	61.04	51.71	51.00
t5	0.05	0.05	0.05
t6	11.45	10.49	10.75
tt	94.13	83.88	83.22
sp-1	---	1.18	1.20
sp-2	---	1.12	1.13
ef-1	---	0.59	0.40
ef-2	---	0.56	0.38

Tabela V.8 - Análise por subestruturas com até 3 processadores

Uma primeira observação que deve ser feita baseada numa comparação entre os resultados obtidos entre o primeiro exemplo e o segundo é de que o "speedup" (sp-1), considerando somente a parte do problema executada em paralelo, se manteve constante ao usarmos dois processadores, logo, a eficiência também foi a mesma. Deve-se notar também que o fato de usarmos três processadores para a análise do segundo exemplo não representou quase nenhum ganho de tempo quando comparado ao tempo que foi gasto para mesma análise só que utilizando dois processadores.

O número de nós de interface de cada subestrutura, devido a geometria da estrutura, é igual a 4 ou 8. Se a

subestrutura possuir dois vizinhos, ou seja, se ela se encontrar entre as extremidades da torre, o número de nós de interface será 8. Se por outro lado, o número de nós for igual a 4, então ela é uma subestrutura que só tem um vizinho e, portanto, envolve uma das duas extremidades.

O terceiro exemplo buscou aumentar o número de subdomínios para 4. Tendo a sua forma regular disposta como o indicado na figura V.1, este exemplo apresenta 288 nós e 497 elementos. A descrição do exemplo encontra-se na tabela V.9 .

número de nós .....	288
número de elementos .....	497
número de graus de liberdade do nó .....	3
número de nós do elemento .....	2
descrição = "elemento reticulado de pórtico plano"	
número de casos de carregamento .....	1
número de graus de liberdade da estrutura .....	852
número de subdomínios .....	4
altura (h).....	142 m
largura (b).....	9 m
vão b .....	3 m
vão h .....	2 m
nh .....	71
nb .....	3

Tabela V.9 - Descrição do modelo

A aplicação do algoritmo de subestruturação automática forneceu os resultados dados pela tabela V.10.

Subdomínio	n ° de nós internos	n ° de nós de interface	n ° de elementos
1	72	4	124
2	67	8	124
3	70	4	124
4	67	8	125

Tabela V.10 - Resultados da divisão automática

O fluxograma I foi novamente aplicado a este modelo, variando o número de processadores de 1 até 4. Todos os processadores são carregados em sequência, ou seja, um após o outro as informações são recebidas e a partir daí eles são colocados no estado "running", conforme explicado no capítulo III ,item III.2 . O mesmo procedimento é repetido para quando o hospedeiro recebe os resultados do processamento. Os resultados encontrados com o quarto exemplo estão na tabela V.11.

np	1	2	3	4
t1	8.86	9.18	8.87	8.62
t2	0.04	0.04	0.04	0.05
t3	14.77	14.81	14.20	14.52
t4	83.55	69.45	70.28	69.20
t5	0.24	0.09	0.10	0.09
t6	14.82	15.22	15.19	14.68
tt	122.28	108.80	108.68	107.16
sp-1	---	1.20	1.19	1.21
sp-2	---	1.12	1.13	1.14
ef-1	---	0.60	0.40	0.30
ef-2		0.56	0.38	0.29

Tabela V.11 - Análise por subestruturas com até 4 processadores

Um aspecto importante a ser observado neste exemplo, aparece quando nota-se que partes do programa que operam de forma sequencial, e que obviamente independentem do número de processadores, apresentam tempos diferentes ao se variar o número de processadores. Por exemplo, o tempo gasto na leitura e impressão dos dados de entrada do problema variou de 8.62 até 9.18, portanto uma diferença de 0.5 segundos. A explicação está na instabilidade do equipamento ocasionada por operações de manutenção de suas funções vitais. Esta diferença tem que ser levada em conta para resultados muito próximos, como os tempos de resolução usando 2, 3 ou 4 processadores.

Sabendo-se que foi mantido o tempo de processamento

nos vários processadores, e que ao mesmo tempo manteve-se o número de informações enviadas e recebidas, para todos os exemplos, notam-se dois fatos: o primeiro é que o tempo gasto para a montagem da matriz de rigidez evoluiu de forma proporcional de exemplo para exemplo, e segundo que não houve ganho neste mesmo tempo quando se aumentou o número de processadores de 2 para 4.

O tempo de formação da matriz de rigidez de uma subestrutura e geração do superelemento correspondente, levando-se em conta que todos os subdomínios possuem aproximadamente o mesmo tamanho, pode ser calculado através do tempo global de geração ( $t_4$ ) dividido pelo número de subdomínios, ao se operar o problema de forma sequencial, ou seja, com um processador. Para o primeiro exemplo, temos:

$$t_4 = 40.60 \quad ;$$

$$\text{logo, } t_4 / 2 \approx 20 \text{ segundos}$$

no segundo exemplo, temos:

$$t_4 = 61,04 \quad ;$$

$$\text{logo, } t_4 / 3 \approx 20 \text{ segundos}$$

no terceiro exemplo, temos:

$$t_4 = 83.55 \quad ;$$

$$\text{logo, } t_4 / 4 \approx 21 \text{ segundos}$$

Estes resultados demonstram que houve balanceamento para as cargas de todos os processadores. Todos os subdomínios foram processados com o mesmo tempo de C.P.U nos nós. Vale salientar que todos exemplos que rodaram com

um processador foram executados pela mesma unidade de processamento, ou seja, utilizaram a mesma C.P.U.. É fundamental se esclarecer este ponto, porque a disposição física de uma C.P.U no sistema, se ela está mais "distante" ou não do hospedeiro, influencia no tempo de comunicação. Além disso, C.P.U.s diferentes podem vir a apresentar diferentes velocidades de processamento.

Em seguida, é dado o quarto exemplo. Ele tem sua forma dada pela figura V.1, tendo-se como parâmetros todos valores dados pela tabela V.12 . O número de nós é 356 e a torre possui 616 elementos de pórtico plano.

número de nós .....	356
número de elementos .....	616
número de graus de liberdade do nó .....	3
número de nós do elemento .....	2
descrição = "elemento reticulado de pórtico plano"	
número de casos de carregamento .....	1
número de graus de liberdade da estrutura .....	1056
número de subdomínios .....	5
altura (h).....	176 m
largura (b).....	9 m
vão b .....	3 m
vão h .....	2 m
nh .....	88
nb .....	3

Tabela V.12 - Descrição do modelo

A divisão do domínio inicial em 5 subdomínios ficou como mostra a tabela V.13.

Subdomínio	n <sup>o</sup> de nós internos	n <sup>o</sup> de nós de interface	n <sup>o</sup> de elementos
1	71	4	123
2	66	8	123
3	67	8	123
4	66	8	123
5	70	4	124

Tabela V.13 - Resultados da divisão automática

A resolução do quarto exemplo, ainda pelo fluxograma I, forneceu os valores dados na tabela V.14.

np	1	2	3	4	5
t1	10.77	10.76	10.73	10.66	10.77
t2	0.05	0.07	0.05	0.05	0.05
t3	14.31	14.74	14.30	14.61	15.11
t4	107.75	87.89	88.07	87.27	86.48
t5	0.14	0.11	0.14	0.10	0.14
t6	17.44	17.41	17.38	17.47	17.37
tt	150.46	130.98	130.68	130.16	129.91
sp-1	---	1.22	1.22	1.23	1.25
sp-2	---	1.15	1.15	1.16	1.16
ef-1	---	0.61	0.41	0.31	0.25
ef-2	---	0.57	0.38	0.25	0.23

Tabela V.14 - Análise por subestruturas com até 5 processadores

Antes de entrarmos no quinto exemplo, deve-se esclarecer alguns dos resultados já encontrados e certas tendências que estão se apresentando.

O quarto exemplo comprova a hipótese de que todos os subdomínios estão sendo processados com o mesmo tempo, aproximadamente 20 segundos de "elapsed time". O tempo de

20 segundos inclui o tempo de se enviar dados aos processadores, o tempo de processamento propriamente dito, e mais o tempo de envio de resultados e armazenamento em disco. Isto nos leva a concluir que todos os processadores estão com a mesma carga de trabalho.

Foi feito uma medição do tempo de C.P.U. dos nós, os resultados encontrados foram de que o tempo que o processador leva para gerar a matriz de rigidez e posteriormente condensá-la é de 3 segundos. Ora, se o tempo total foi de 20 e o de processamento real foi de 3, existe uma diferença de 17 segundos.

Os 17 segundos, então, foram gastos em comunicação e em acesso a unidade de disco. O acesso a unidades de disco no hospedeiro, Microvax, vai ter a sua velocidade afetada a medida que o número de usuários do sistema for alterado. Contudo, em todos os exemplos, para se evitar a flutuação no tempo de disco, o sistema inteiro sempre operou com somente um usuário. Os resultados enviados dos nós para o hospedeiro, conforme explicado no capítulo IV, ítem IV.3, incluem toda a matriz da subestrutura e mais o vetor de cargas. A resolução do sistema de equações para cálculo dos deslocamentos nos nós de interface somente envolve termos associados aos superlelementos, logo, aqueles valores que não são necessários são guardados em unidades de disco para posterior leitura.

Na prática, o tempo de comunicação deve somar o tempo de acesso a disco e o tempo de transmissão de informações, pois é este o tempo real em que se manipulou os dados fornecidos pelos processadores paralelos. Assim sendo,

toda vez que falarmos em tempo de comunicação, deve-se lembrar de que ele é uma soma de dois tempos.

Como demonstram os vários valores de speedup e eficiência já encontrados, o aumento no número de subdomínios não foi acompanhada pelos valores de speedup. Mesmo crescendo com o número de eventos e de processadores, o speedup não se alterou, e evidentemente a eficiência caiu. Neste quarto exemplo vemos que se usamos 2 ou 5 processadores obtemos o mesmo tempo  $t_4$ , ou seja, apesar de aumentarmos o número de processadores disponíveis, o programa, na prática, comporta-se como se só houvessem sempre 2 processadores.

A explicação para esta performance é de que o tempo de comunicação está predominando na execução.

Supondo que tivéssemos 5 eventos e cinco processadores. Os processadores são carregados em sequência, daí, carregamos o primeiro nó com o primeiro evento. Partimos para o segundo que também será carregado só que com o segundo evento. O que se faz agora é carregar o terceiro evento no processador de número três, contudo o primeiro processador já está pronto para enviar as suas respostas. Assim sendo, na verdade não se precisa de mais de dois processadores.

O último exemplo a ser testado trata-se de uma torre com 424 nós e 735 elementos. A sua forma geral é dada pela figura V.1 tendo como parâmetros os valores dados pela tabela V.15 .

número de nós .....	424
número de elementos .....	735
número de graus de liberdade do nó .....	3
número de nós do elemento .....	2
descrição = "elemento reticulado de pórtico plano"	
número de casos de carregamento .....	1
número de graus de liberdade da estrutura .....	1260
número de subdomínios .....	6
altura (h).....	210 m
largura (b).....	9 m
vão b .....	3 m
vão h .....	2 m
nh .....	105
nb .....	3

Tabela V.15 - Descrição do modelo

O número de graus de liberdade do sistema, incluindo nós internos e de interface, totalizam 1260. Por uma análise convencional pelo método dos elementos finitos seria esta a dimensão do sistema de equações.

Como se deseja repartir este domínio por 6, usando o algoritmo de partição automática, temos os resultados na tabela V.16.

Subdomínio	n <sup>o</sup> de nós internos	n <sup>o</sup> de nós de interface	n <sup>o</sup> de elementos
1	71	4	122
2	69	4	122
3	66	8	122
4	65	8	122
5	66	8	122
6	67	8	125

Tabela V.16 - Resultados da divisão automática.

O número de nós de interface encontrada nesta subestruturação foi de 20. Com isso, o sistema de equações, representado pelo somatório das contribuições das várias subestruturas através dos seus superelementos, terá dimensão 60. A diferença, portanto, entre uma análise convencional e o método de subestruturas, neste exemplo, foi de 1200 equações.

Usando até 6 processadores para analisar este quinto exemplo, encontramos a performance dada na tabela V.17.

np	1	2	3	4	5	6
t1	12.46	12.73	12.48	12.42	12.52	12.59
t2	0.06	0.05	0.08	0.05	0.06	0.05
t3	15.75	14.63	14.72	14.52	14.84	14.47
t4	129.34	101.94	100.46	101.53	100.39	99.86
t5	0.17	0.17	0.18	0.18	0.17	0.18
t6	20.55	19.41	19.07	19.35	19.87	19.59
tt	178.33	148.94	146.98	148.05	147.84	146.75
sp-1	---	1.27	1.29	1.27	1.29	1.30
sp-2	---	1.19	1.21	1.20	1.20	1.22
ef-1	---	0.63	0.43	0.32	0.25	0.22
ef-2	---	0.60	0.40	0.30	0.24	0.20

Tabela V.17 - Análise por subestruturas com até 6 processadores

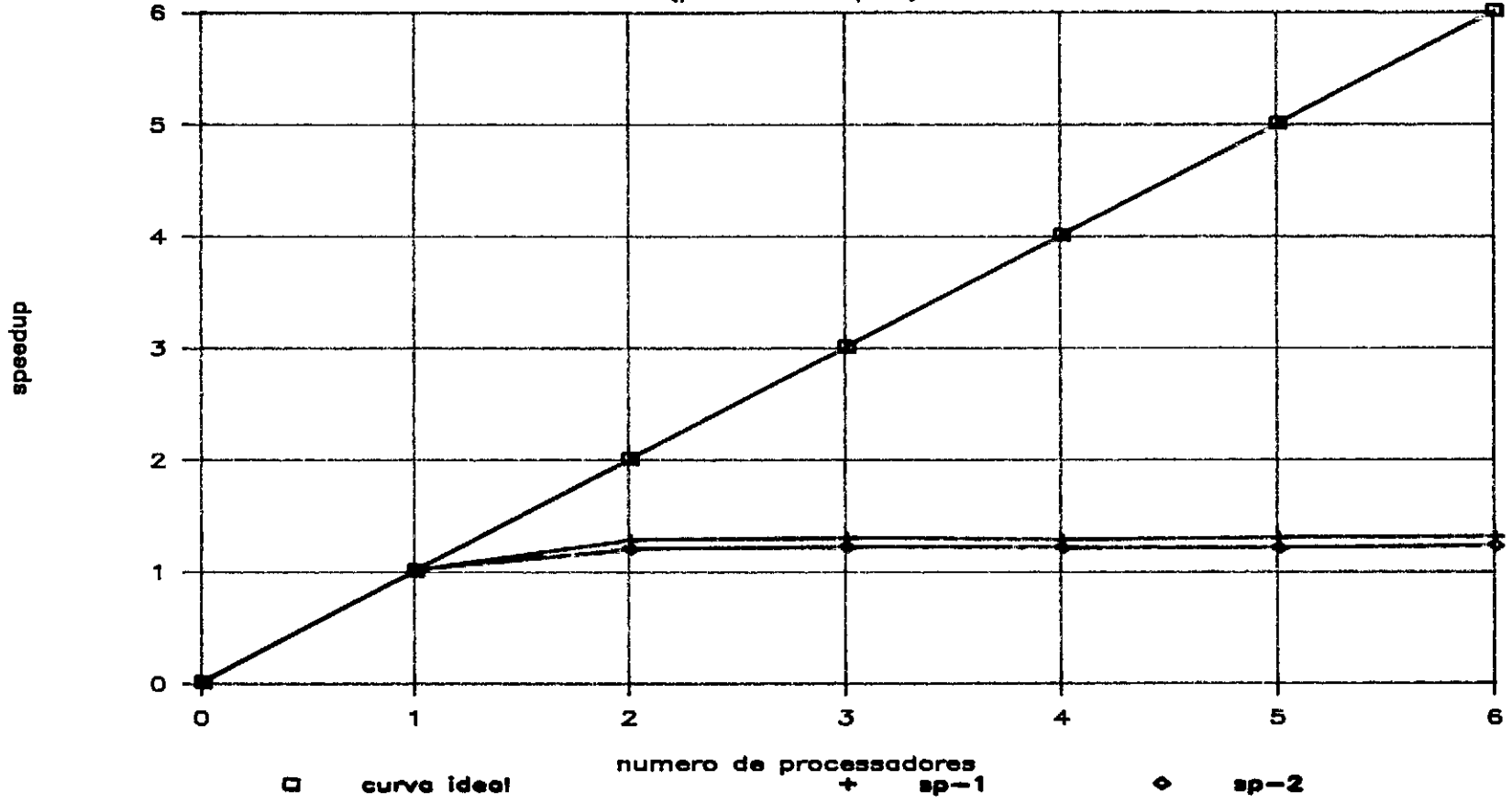
O gráfico V.1 traça o comportamento do valor de speedup a medida em que aumentamos o número de processadores. A curva ideal é aquela curva que representa um funcionamento ideal de programa e máquina. Neste caso,

teríamos que cada vez que incluíssemos mais um processador a execução se tornaria mais rápida na mesma proporção. Com 2 processadores o programa é 2 vezes mais rápido do que com 1, com 3 processadores 3 vezes mais rápido, e assim por diante.

As curvas  $sp-1$  e  $sp-2$  representam, respectivamente, os valores de speedup somente na fase paralela do programa e os valores de speedup considerando o programa como um todo.

# FLUXOGRAMA I

(precisão simples)



Como se pode notar pelas curvas do gráfico V.1, não se conseguiu trabalhar com mais de 2 processadores em nenhum dos exemplos dados. Quando a quantidade de informação a ser transmitida é muito grande e o tempo de C.P.U. dos nós não consegue compensar o tempo de comunicação, o sistema degenera. De nada adianta se acrescentar processadores pois, na verdade, os processadores estão ociosos e a eficiência só pode ser medida com 2 processadores.

Em alguns casos, o tempo  $t_4$  com mais processadores chegou até a piorar o speedup. Isto ocorreu porque o incremento de processadores leva ao aumento do tempo em que um processador espera para descarregar os seus resultados. O estrangulamento no recebimento de informações aumenta o número de operações de controle do sistema, e por isso a queda de rendimento.

Os valores de speedup pelo fluxograma I foram tão baixos que não justificam um paralelismo com este tipo de problema aplicado a esta máquina. Deve-se, então, diminuir o tempo de comunicação e elevar o de C.P.U. nos nós. Os próximos exemplos, que utilizam o fluxograma III, tentam atingir estes objetivos.

A diferença básica entre o fluxograma I e III está na quantidade de informações a serem transmitidas. No fluxograma I, a transmissão é da matriz de rigidez da subestrutura, enquanto que no fluxograma III é só da matriz do superelemento.

O fluxograma III pode operar em precisão simples e dupla. Os resultados usando dupla precisão são indicados

entre parênteses.

O fluxograma III em precisão simples enviou mensagens para os processadores com 14 Kbytes e recebeu mensagens de 8 Kbytes. Em precisão dupla estes valores foram dobrados, passando para 28 Kbytes e 8 Kbytes, respectivamente.

A escolha dos 5 primeiros exemplos foi feita de modo eles pudessem ser executados em precisão simples e dupla, e assim, comparar as duas performances. Os exemplos ocuparam os 2 Mbytes de memória disponível de cada processador para serem executados.

O primeiro exemplo a ser executado com o fluxograma III foi de uma torre com 292 nós e 504 elementos. O número de graus de liberdade total, introduzidas as condições de contorno, é 864. A descrição é dada na tabela V.18.

número de nós .....	292
número de elementos .....	504
número de graus de liberdade do nó .....	3
número de nós do elemento .....	2
descrição = "elemento reticulado de pórtico plano"	
número de casos de carregamento .....	1
número de graus de liberdade da estrutura .....	864
número de subdomínios .....	2
altura (h).....	144 m
largura (b).....	9 m
vão b .....	3 m
vão h .....	2 m
nh .....	72
nb .....	3

Tabela V.18 - Descrição do modelo

A decomposição ficou como o indicado na tabela V.19.

Subdomínio	n <sup>o</sup> de nós internos	n <sup>o</sup> de nós de interface	n <sup>o</sup> de elementos
1	145	4	252
2	143	4	252

Tabela V.19 - Resultados da divisão automática

Nota-se o aumento sensível do número de elementos em cada subdomínio em relação aos exemplos executados pelo fluxograma I. A consequência direta é o aumento do tempo de C.P.U. nos nós. A análise por subestruturas ficou como o indicado na tabela V.20.

np	1	2
t1	9.45 (9.41)	9.59 (9.49)
t2	0.03 (0.02)	0.04 (0.02)
t3	15.96 (16.20)	15.10 (15.55)
t4	15.84 (17.34)	9.30 (9.94)
t5	0.02 (0.01)	0.01 (0.01)
t6	1.70 (1.64)	3.89 (1.66)
tt	43.00 (44.62)	37.93 (36.67)
sp-1	---	1.70 (1.74)
sp-2	---	1.13 (1.22)
ef-1	---	0.85 (0.87)
ef-2	---	0.57 (0.61)

Tabela V.20 - Análise por subestruturas com até 2 processadores

O tempo para cálculo das tensões nos elementos básicos e impressão de resultados (t6) foi alterado para somente impressão de resultados. Isto se deve a impossibilidade de calcular as tensões, já que, para este fluxograma, não são calculados os deslocamentos para os nós internos.

O fluxograma III só faz acesso a unidades de disco para entrada de dados e para saída de resultados. Toda a matriz do superelemento é somada ao sistema de equações e como não há necessidade de se calcular os deslocamentos

nos nós internos e as tensões nos elementos, a matriz completa da subestrutura não chega a ser transmitida pelos nós.

O segundo exemplo diz respeito a uma torre semelhante a da figura V.1 com 436 nós e 756 elementos. O carregamento repete o mesmo procedimento dos exemplos anteriores, ou seja, cargas de valor unitário aplicadas diretamente em todos os nós laterais da estrutura. Ele é descrito na tabela V.21.

número de nós .....	436
número de elementos .....	756
número de graus de liberdade do nó .....	3
número de nós do elemento .....	2
descrição = "elemento reticulado de pórtico plano"	
número de casos de carregamento .....	1
número de graus de liberdade da estrutura .....	1296
número de subdomínios .....	3
altura (h).....	216 m
largura (b).....	9 m
vão b .....	3 m
vão h .....	2 m
nh .....	108
nb .....	3

Tabela V.21 - Descrição do modelo

A divisão em 3 subdomínios ficou como o indicado na tabela V.22.

Subdomínio	n <sup>o</sup> de nós internos	n <sup>o</sup> de nós de interface	n <sup>o</sup> de elementos
1	145	4	252
2	140	8	252
3	143	4	252

Tabela V.22 - Resultados da divisão automática

Variando-se o número de processadores de 1 a 3, encontramos para este exemplo os resultados dados pela tabela V.23.

np	1	2	3
t1	13.04 (13.35)	13.04 (13.85)	12.81 (13.51)
t2	0.36 (0.30)	0.36 (0.29)	0.43 (0.36)
t3	16.51 (15.33)	14.97 (15.34)	15.15 (15.44)
t4	29.77 (31.84)	15.92 (17.50)	15.39 (16.45)
t5	0.05 (0.18)	0.05 (0.07)	0.05 (0.05)
t6	2.34 (2.32)	2.23 (2.30)	2.25 (2.23)
tt	62.06 (63.32)	46.57 (49.35)	46.09 (48.04)
sp-1	---	1.87 (1.82)	1.93 (1.94)
sp-2	---	1.33 (1.28)	1.35 (1.32)
ef-1	---	0.93 (0.91)	0.64 (0.65)
ef-2	---	0.67 (0.64)	0.45 (0.44)

Tabela V.23 - Análise por subestruturas com até 3 processadores

A tabela V.23 mostra que a eficiência ao se trabalhar com dois processadores foi muito alta, chegando a 93% em precisão simples e 91% em precisão dupla. Em comparação ao exemplo anterior onde se usou dois subdomínios e dois processadores, houve um ganho de 8% e 4%, respectivamente. Isto quando consideramos somente a parte paralela do programa, se considerarmos o programa total (sp-2), então a diferença diminui.

O terceiro exemplo escolhido para ser aplicado o fluxograma III é de uma torre regular com 576 nós e 1001 elementos. A descrição está na tabela V.24 .

número de nós .....	576
número de elementos .....	1001
número de graus de liberdade do nó .....	3
número de nós do elemento .....	2
descrição = "elemento reticulado de pórtico plano"	
número de casos de carregamento .....	1
número de graus de liberdade da estrutura .....	1716
número de subdomínios .....	4
altura (h).....	286 m
largura (b).....	9 m
vão b .....	3 m
vão h .....	2 m
nh .....	143
nb .....	3

Tabela V.24 - Descrição do modelo

Como se pode ver, este exemplo, assim como todos os outros, não representam casos reais. São simplesmente modelos teóricos ajustados a uma necessidade de pesquisa.

A divisão em 4 subdomínios é dada na tabela V.25 .

Subdomínio	n <sup>o</sup> de nós internos	n <sup>o</sup> de nós de interface	n <sup>o</sup> de elementos
1	144	4	250
2	139	8	250
3	142	4	250
4	139	8	250

Tabela V.25 - Resultados da divisão automática

A análise por subestruturas está na tabela V.26.

np	1	2	3	4
t1	17.02 (17.30)	17.04 (21.30)	17.07 (16.73)	18.59 (16.67)
t2	0.05 (0.02)	0.06 (0.02)	0.06 (0.02)	0.09 (0.02)
t3	14.89 (15.77)	14.91 (15.79)	14.68 (15.53)	17.27 (15.02)
t4	42.17 (46.14)	28.55 (31.16)	22.06 (23.77)	15.42 (17.27)
t5	0.09 (0.13)	0.09 (0.13)	0.09 (0.13)	0.09 (0.13)
t6	3.16 (3.36)	3.29 (3.34)	3.19 (3.16)	3.40 (3.14)
tt	77.38 (82.72)	63.94 (71.75)	57.15 (59.34)	54.86 (52.24)
sp-1	---	1.48 (1.48)	1.91 (1.94)	2.73 (2.67)
sp-2	---	1.21 (1.15)	1.35 (1.39)	1.41 (1.58)
ef-1	---	0.74 (0.74)	0.64 (0.65)	0.68 (0.67)
ef-2	---	0.61 (0.58)	0.45 (0.46)	0.35 (0.40)

Tabela V.26 - Análise por subestruturas com até 4 processadores

Um aspecto interessante que aparece neste e em outros exemplos é que se temos 4 eventos, teóricamente deveríamos ter o mesmo tempo  $t_4$  usando 2 ou 3 processadores. Isto ocorreria porque, supondo-se que o tempo de comunicação fosse zero, deveríamos ter para as duas situações sempre duas operações de carregar os nós. O que seria afetado é a eficiência do sistema, já que se utilizássemos três em lugar de dois processadores, na segunda operação de carga, teríamos dois processadores ociosos durante a resolução do quarto evento.

Contudo, o que se verifica na prática é que sempre que se aumenta o número de processadores há uma variação no tempo em que o programa opera em paralelo.

A explicação para este comportamento é a instabilidade do sistema. A instabilidade pode estar em fatores ligados a programação, como quantidade informações transmitidas, ou em fatores ligados ao rendimento da máquina, como velocidade de processamento da C.P.U.

O quarto exemplo envolve 724 nós e 1260 elementos de pórtico plano. A dimensão global do sistema de equações é de 2160 graus de liberdade.

número de nós .....	724
número de elementos .....	1260
número de graus de liberdade do nó .....	3
número de nós do elemento .....	2
descrição = "elemento reticulado de pórtico plano"	
número de casos de carregamento .....	1
número de graus de liberdade da estrutura .....	2160
número de subdomínios .....	5
altura (h).....	360 m
largura (b).....	9 m
vão b .....	3 m
vão h .....	2 m
nh .....	180
nb .....	3

Tabela V.27 - Descrição do modelo

A subestruturação automática se deu como o indicado na tabela V.28 .

Subdomínio	n <sup>o</sup> de nós internos	n <sup>o</sup> de nós de interface	n <sup>o</sup> de elementos
1	145	4	252
2	140	8	252
3	140	8	252
4	140	8	252
5	143	4	252

Tabela V.28 - Resultados da divisão automática

A performance do programa, variando-se o número de processadores de 1 até 5 está na tabela V.29 .

np	1	2	3	4	5
t1	21.29 (21.14)	21.27 (21.06)	21.37 (21.39)	22.13 (21.35)	21.24 (21.15)
t2	0.07 (0.03)	0.07 (0.03)	0.07 (0.03)	0.08 (0.03)	0.16 (0.03)
t3	15.48 (15.21)	15.09 (15.18)	15.27 (15.34)	15.96 (15.12)	16.24 (14.95)
t4	56.97 (61.64)	29.66 (32.18)	23.05 (25.57)	15.97 (17.55)	15.58 (17.63)
t5	0.13 (0.31)	0.13 (0.19)	0.13 (0.20)	0.14 (0.20)	0.13 (0.16)
t6	4.14 (4.14)	4.09 (4.25)	4.04 (4.22)	4.16 (4.39)	4.04 (4.41)
tt	98.08 (102.48)	70.32 (72.89)	63.92 (66.74)	58.44 (58.63)	57.39 (58.33)
sp-1	- ---	1.92 (1.92)	2.47 (2.41)	3.57 (3.51)	3.66 (3.50)
sp-2	- ---	1.39 (1.41)	1.53 (1.54)	1.68 (1.75)	1.71 (1.76)
ef-1	- ---	0.96 (0.96)	0.82 (0.80)	0.89 (0.88)	0.73 (0.70)
ef-2	- ---	0.70 (0.70)	0.51 (0.51)	0.42 (0.44)	0.34 (0.35)

Tabela V.29 - Análise por subestruturas com até 5 processadores

O exemplo dado acima mostra uma redução grande no número de equações a ser resolvida. Enquanto numa análise tradicional o número de equações chega a 2160, a divisão em subestruturas permite a redução para 48 equações.

A técnica de subestruturas permite que se distribua entre os vários processadores a necessidade de memória. É claro que quando utilizou-se os 5 processadores gastou-se 10 Mbytes, entretando a paralelização permitiu que esta necessidade fosse dividida por cinco, e ainda que o acesso a estas informações fossem diretas. O armazenamento dos diversos valores é feito por memória do tipo real (RAM).

Se partíssemos para uma análise convencional, a alternativa seria a blocagem do sistema de equações e armazenamento em disco, ou a utilização de toda a capacidade de memória principal e somando-se a ela as memórias periféricas e virtuais. Ambas as alternativas, a medida que o problema aumentasse de tamanho, poderiam se tornar inviáveis, pois o tempo de acesso as variáveis iria aumentar consideravelmente.

O gráfico V.2 traça o comportamento aproximado do sistema ao operar o fluxograma III em precisão simples. São considerados os valores de speedup na parte paralela do programa (sp-1), e no tempo total de execução (sp-2).

# FLUXOGRAMA III (precisão simples)

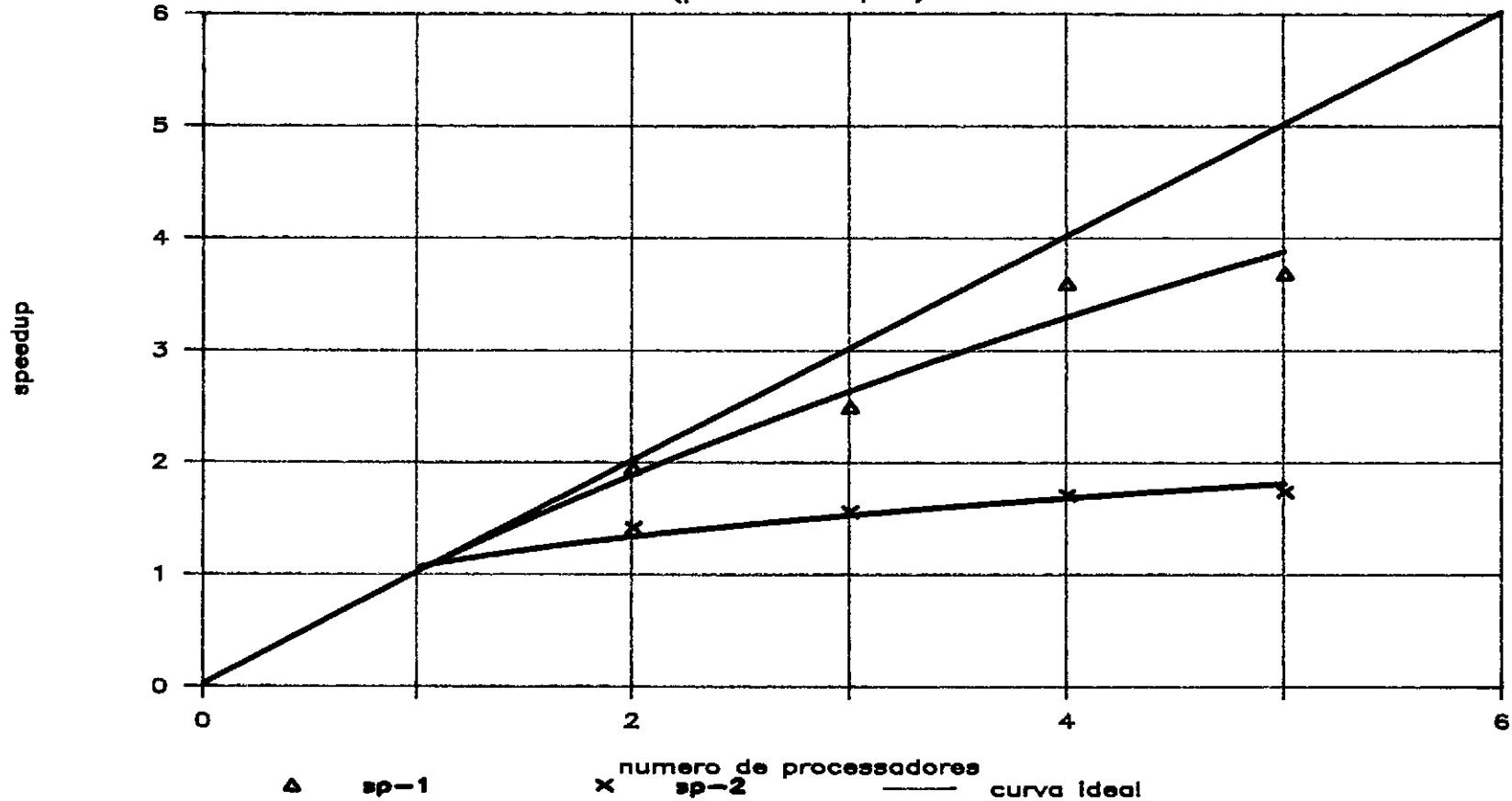


Gráfico V.2 - Evolução em speedup com o Fluxograma III

Os valores de speedup na fase paralela, para todos os exemplos executados pelo fluxograma III em precisão simples, cresceram a medida em que aumentamos o número de eventos. No exemplo acima atingiu-se 3.66, ou seja, o tempo gasto quando operou-se com o máximo de processadores possíveis foi 3,66 vezes mais rápido do que utilizando um processador.

A execução em precisão simples e dupla mostraram, para todos os exemplos, um comportamento dentro do esperado. A precisão dupla fez com que se aumentasse o tempo de comunicação, daí, o tempo total de processamento do subdomínio alcançar um valor maior.

Evidentemente que o crescimento do tempo de comunicação poderia supor que os valores de speedup seriam afetados. O que se verificou na prática, foi que este aumento foi pequeno e acabou por não pesar muito no índice, seja pelo fato de a quantidade de informações enviadas ser relativamente pequena, seja por ela ser compensada com o aumento no tempo de C.P.U. dos nós ao operarem em precisão dupla.

A escolha em se operar em precisão simples não se faz baseando-se somente em disposição do usuário, por vezes tem-se de usar a precisão dupla por uma necessidade.

O último exemplo a ser executado pelo fluxograma III é o descrito na tabela V.30. A resolução deste exemplo só pode ser feita utilizando-se precisão dupla nas variáveis reais, por isso, não podem ser comparadas a performance do programa em precisão simples e dupla.

O limite encontrado para se aplicar a precisão

simples foi de o número de graus de liberdade global estar abaixo de 2580.

Este último exemplo tem 864 nós e 1505 elementos. O carregamento é o mesmo dos exemplos anteriores e é aplicado diretamente nos nós laterais. A altura total do modelo é 430 metros e a largura é de 9 metros. O objetivo de estudar este exemplo é verificar o comportamento do sistema ao se carregar os 6 subdomínios nos processadores paralelos.

número de nós .....	864
número de elementos .....	1505
número de graus de liberdade do nó .....	3
número de nós do elemento .....	2
descrição = "elemento reticulado de pórtico plano"	
número de casos de carregamento .....	1
número de graus de liberdade da estrutura .....	2580
número de subdomínios .....	6
altura (h).....	430 m
largura (b).....	9 m
vão b .....	3 m
vão h .....	2 m
nh .....	215
nb .....	3

Tabela V.30 - Descrição do modelo

A decomposição em 6 subdomínios é indicada na tabela V.31 .

Subdomínio	n <sup>o</sup> de nós internos	n <sup>o</sup> de nós de interface	n <sup>o</sup> de elementos
1	144	4	250
2	139	8	250
3	142	4	250
4	139	8	252
5	139	8	250
6	141	8	255

Tabela V.31 - Resultados da divisão automática

Este exemplo ainda manteve o critério de colocar cada subdomínio com aproximadamente 250 elementos, e desta forma explorar ao máximo a memória disponível de cada processador. A análise deste exemplo está na tabela V.32 .

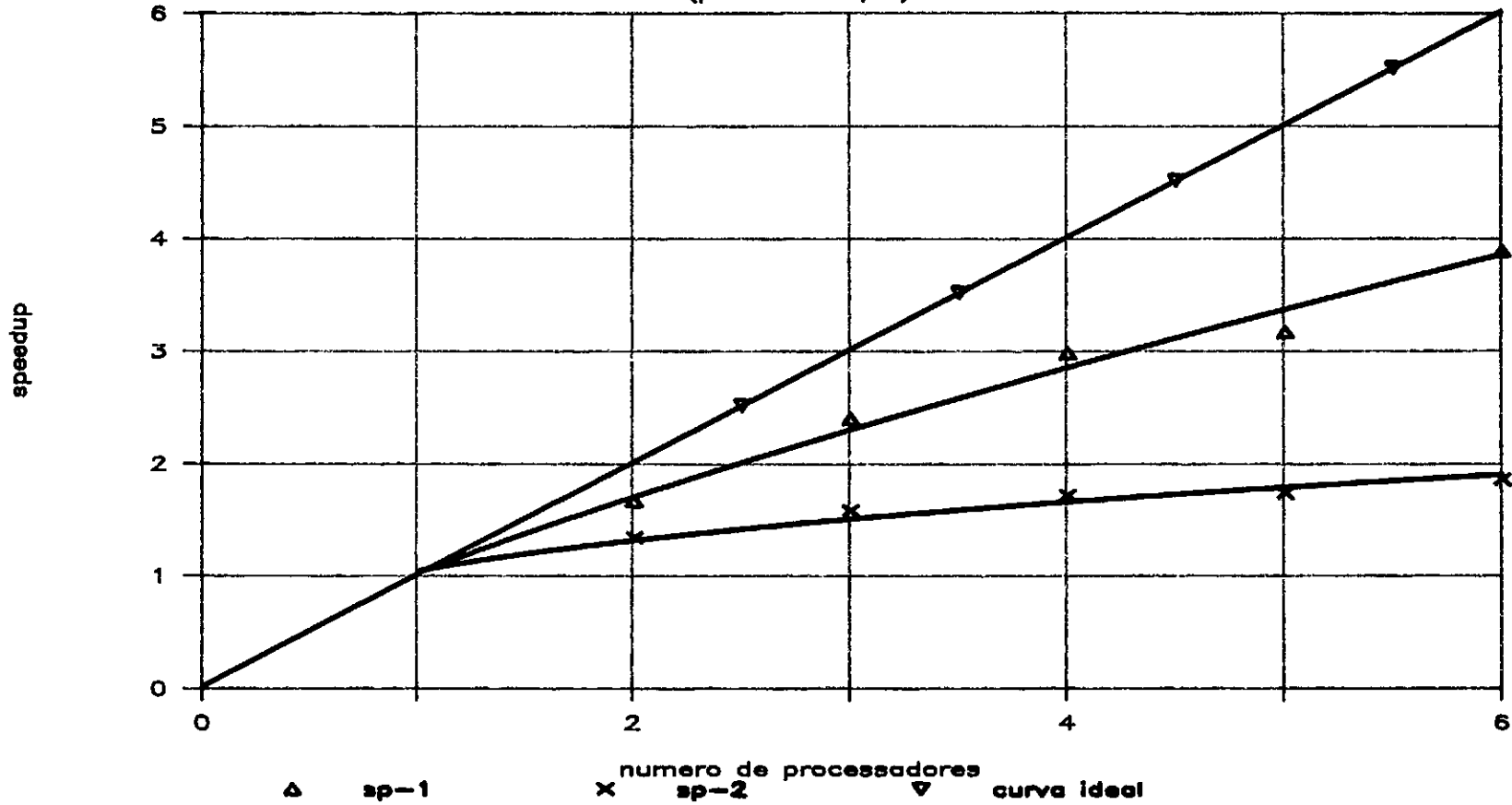
np	1	2	3	4	5	6
t1	25.09	25.24	25.47	25.20	25.38	25.59
t2	0.03	0.04	0.04	0.02	0.03	0.02
t3	14.96	15.25	15.16	15.02	15.67	14.88
t4	75.80	45.84	31.80	25.65	24.06	19.57
t5	0.25	0.23	0.25	0.24	0.25	0.29
t6	4.94	4.96	4.88	4.92	4.80	5.36
tt	121.08	91.56	77.59	71.06	70.20	65.71
sp-1	---	1.65	2.38	2.96	3.15	3.87
sp-2	---	1.32	1.56	1.70	1.72	1.84
ef-1	---	0.83	0.79	0.74	0.63	0.65
ef-2	---	0.66	0.52	0.43	0.34	0.31

Tabela V.32 - Análise por subestruturas com até 6 processadores

O gráfico V.3 traça o comportamento da curva de speedup para este exemplo.

# FLUXOGRAMA III

(precisao dupla)



Usando o mesmo princípio adotado para os exemplos com o fluxograma I, podemos calcular o tempo aproximado de montagem da matriz de rigidez e do superelemento por:

	tempo t4 com um processador em precisao		numero de subdominios ( ns )	t4/ns em precisao	
	simples	dupla		simples	dupla
1 <sup>o</sup> EXEMPLO	15.84	17.84	2	8	9
2 <sup>o</sup> EXEMPLO	29.77	31.84	3	10	10
3 <sup>o</sup> EXEMPLO	42.17	46.14	4	10.5	11.5
4 <sup>o</sup> EXEMPLO	56.97	61.64	5	11	12
5 <sup>o</sup> EXEMPLO	----	75.80	6		12.5

TABELA V.33

A tabela V.33 mostra um aumento no tempo gasto para se processar cada subdomínio a medida que aumentamos o número de eventos. A primeira suposição é a de que os subdomínios aumentaram de tamanho, entretanto, as tabelas que dão o resultado da subestruturação mostram que o número de elementos em cada subdomínio se manteve praticamente constante para todos os exemplos, algo em torno de 250 elementos.

Eliminada esta possibilidade, a única outra possível é considerar que o "rendimento" de um único processador se altera de evento para evento, mesmo que eles tenham o mesmo tamanho. Isto é comprovado pela análise do tempo de C.P.U. dos nós. Eles mostraram uma oscilação entre 7 e 12 segundos, tanto em precisão simples como em dupla.

A eficiência não acompanhou a evolução dos valores de speedup. Foram encontrados índices altos ao se trabalhar

com 2 e 3 processadores, mas eles tendem a diminuir a medida que aumentamos o número de processadores. Os gráficos V.4 e V.5 mostram as medidas de eficiência encontradas no quatro exemplo operando-se em precisão simples, e do quinto em precisão dupla. Os valores indicados como ef-1 e ef-2 significam ,respectivamente, eficiência na parte paralela e eficiência em todo a execução.

A queda de rendimento que ocorreu só pode ser explicada como uma característica do sistema. A transmissão de informações foi praticamente imediata, como comprova a alta eficiência ao se operar com poucos processadores. O tempo de C.P.U foi elevado ao máximo possível com a completa exploração da sua disponibilidade em memória, chegando sempre a estar acima dos 7 segundos.

### FLUXOGRAMA III (precisao simples)

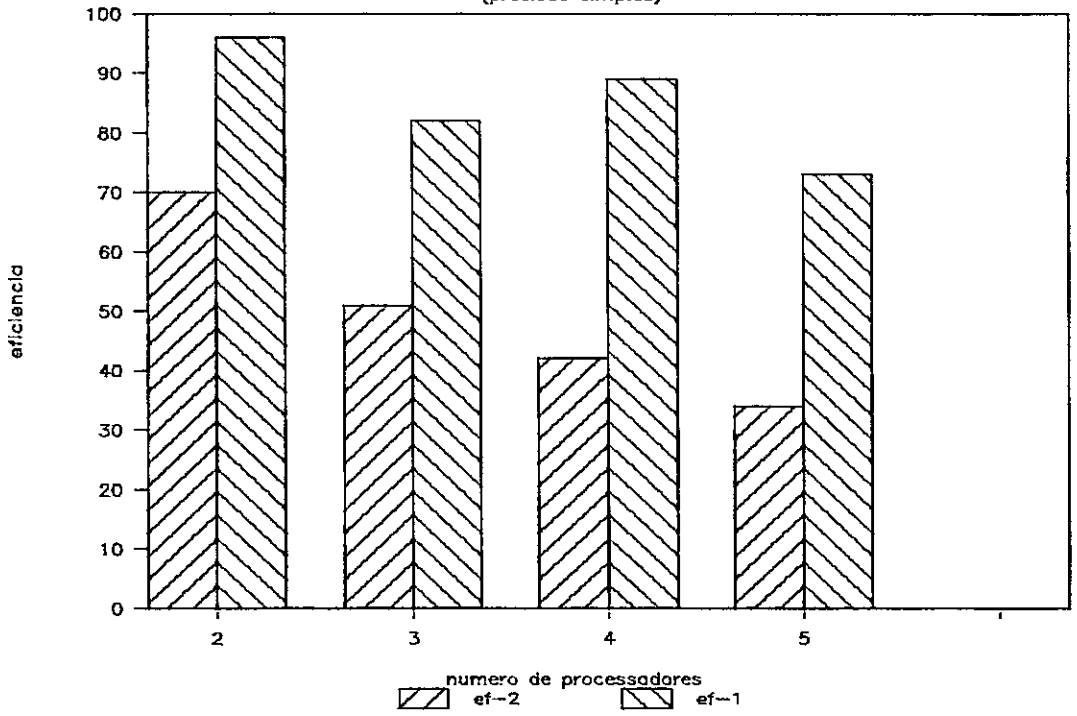


Gráfico V.4 - Eficiência com o fluxograma III em precisão simples

### FLUXOGRAMA III (precisao dupla)

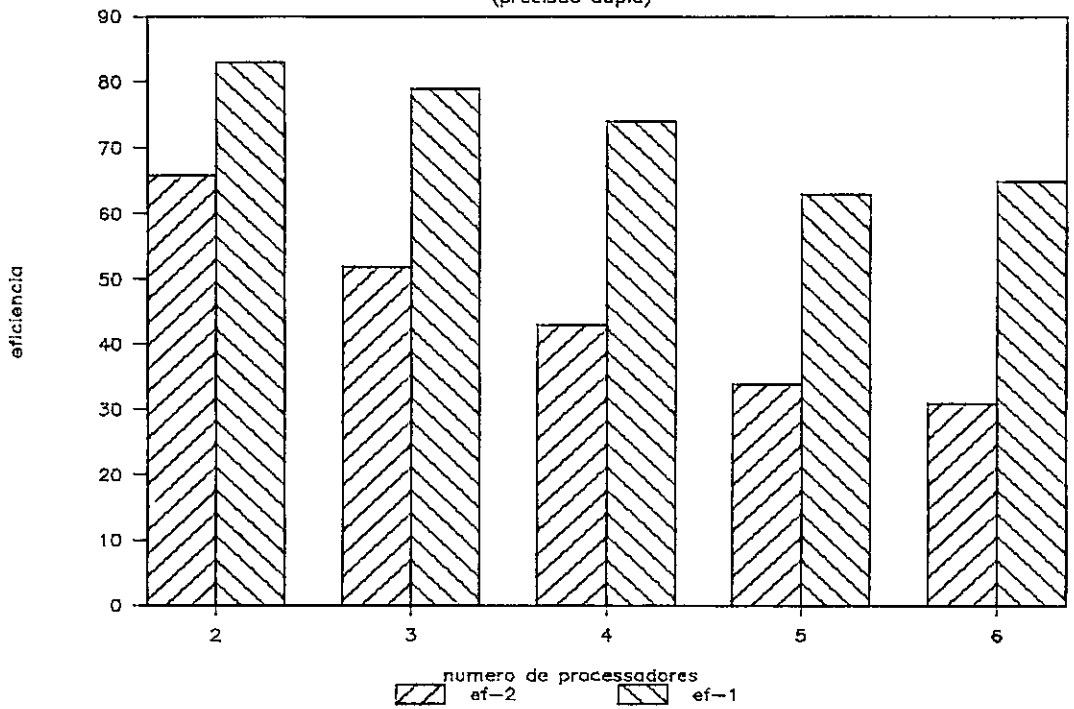


Gráfico V.5 - Eficiência com o fluxograma III em precisão dupla

## CAPÍTULO VI

## CONCLUSÃO

Nos capítulos anteriores foi salientada a importância da partição do domínio para que obtenção de melhora de performance no programa que realiza a análise por subestruturas. O algoritmo apresentado para partição automática mostrou-se satisfatório para prover o usuário de um programa com flexibilidade na escolha do número de subdomínios e rapidez na análise.

A escolha do número de subestruturas, traduzindo-se em número de eventos a serem analisados no A.C.P., deve ser feita sempre de forma a conseguir um tempo significativo de processamento e um balanceamento de cargas para todos os processadores.

O algoritmo pode ser aplicado a estruturas discretizadas com qualquer tipo de elemento, como demonstram os resultados apresentados no capítulo V, item V.2. Contudo a sua utilização tem de ser acompanhada de muita atenção, pois ele deixa de atender de modo mais eficiente ao princípio de que cada subdomínio deve ser escolhido de forma tal a gerar um número mínimo de nós de interface e um número máximo de nós internos. O algoritmo, para escolha dos elementos pertencentes a cada subdomínio, baseia-se somente no conceito de peso nodal mínimo e de vizinhança entre elementos. O ideal seria que ele levasse em conta a disposição dos elementos na estrutura e as eventuais irregularidades de forma.

Estes fatores são muito dependentes do tipo da estrutura analisada e, portanto, difíceis de serem considerados, de forma geral, a nível de programação.

Uma solução proposta seria usar um pré-processador que operasse de forma interativa e que incorporasse características de um sistema especialista, de forma a controlar "inteligentemente" a escolha dos elementos e a configuração de cada subestrutura.

O tipo de problema abordado ao longo de todo o trabalho sempre se caracterizou por tratar de grandes estruturas. Grande, no sentido de envolver um número alto de graus de liberdade e um número elevado de operações envolvidas na condensação das matrizes de rigidez de cada subestrutura, em relação a capacidade do equipamento utilizado.

A resolução de problemas pequenos envolve pouca área de memória e baixo tempo de processamento. Resolver problemas pequenos neste tipo de arquitetura de computação paralela é desaconselhável, pois não se utiliza o sistema com todo o seu potencial.

O fluxograma I, apresentado no capítulo IV, teve um desempenho baixo quando aplicado no sistema A.C.P. O que ocorreu é que a quantidade de informações a serem transmitidas consumiu grande parte do tempo gasto na análise em paralelo. A máquina não oferece uma forma de armazenar valores gerados nos nós (processadores) em unidades periféricas, e esta restrição prejudicou um melhor desempenho porque forçou a transmissão de valores que poderiam ser operados nos nós (processadores).

Todos os exemplos resolvidos através do fluxograma I conseguiram, no máximo, serem operados por 2 processadores. O aumento no número de processadores disponíveis não foi acompanhado de ganho em tempo de resolução do problema.

O segundo tipo de aplicação apresentada estuda o comportamento do fluxograma III que envolve a transmissão somente da matriz do superelemento.

Os exemplos apresentados no capítulo anterior resolvidos pelo fluxograma III em precisão simples e dupla conseguiram atingir altos índices de eficiência e ganhos em tempo de execução (speedup).

Os resultados do capítulo V mostraram que o rendimento se alterou pouco quando passamos a operar com precisão dupla. O fato de dobrarmos a quantidade de informações transmitidas não ocasionou perda no ganho em tempo de execução, ao contrário, por vezes o sistema até melhorou o seu desempenho. O alto tempo de processamento conseguido pela utilização total da memória de cada processador fez com que o tempo de comunicação tivesse pouca influência no tempo total de operação com o subdomínio.

A eficiência conseguida pelo fluxograma III se mostrou bem alta ao se operar com poucos processadores, contudo, a elevação no número de processadores fez com que ela viesse a apresentar uma tendência de queda.

Os sistemas de computação paralela devem ser desenvolvidos de forma a conseguir minimizar a transmissão de informações e ao mesmo tempo conseguir um alto grau de

paralelização do problema. Os resultados com fluxograma I comprovaram a primeira afirmação, e a comparação entre os ganhos em tempo de execução envolvendo todo o problema ou envolvendo somente a parte paralela, seja para o fluxograma I ou o fluxograma III, comprovam a segunda.

Todos os trabalhos foram desenvolvidos dentro da arquitetura existente no sistema A.C.P. A sua configuração rígida limitou a adoção de estratégias de programação mais versáteis.

A existência de um alto grau de vetorização nas operações que envolvem a condensação da matriz de rigidez de cada subestrutura é uma indicativa válida para se levar este problema para um computador de arquitetura paralela e de processamento vetorial.

## REFERÊNCIAS BIBLIOGRÁFICAS

- [1] KIRSCH, U., REISS, M. e SHAMIR, U. - "Optimum Design by Partitioning into Substructures" - Journal of the Structural Division, ASCE , pp. 249-265, (1972).
- [2] PRZEMIENIECKI, J.S. - "Matrix Structural Analysis of Substructures" - AIAA Journal , Vol. 1 , pp. 138-147, (1963).
- [3] PETERSON, H. e POPOV, E.P. - "Substructuring and Equation System Solutions in Finite Element Analysis" - Computers & Structures, Vol. 7 , pp. 197-206, (1977).
- [4] AHMED, K.N. , KAMEL, H.A. e FULTON, R.E. - "Substructuring Techniques - Status and Projections" - Computer & Structures , Vol.8 , pp. 621-632, (1978).
- [5] FARHAT, C e WILSON, E.L. - "Modal Superposition Dynamic Analysis on Concurrent Multiprocessors" - Eng. with Computers, Vol 3 , pp. 305-311, (1986).
- [6] FARHAT, C e WILSON, E.L. - "Concurrent Iterative Solution of Large Finite Element Systems" - Communications in Applied Numerical Methods , Vol. 3 , pp. 319-326, (1987).

- [7] FARHAT, C e WILSON, E.L. - "Linear and Nonlinear Finite Element Analysis on Multiprocessor Computer System" - Communications in Applied Numerical Methods, Vol 4 , pp. 425-434, (1988).
- [8] OMID, B.N., RAEFSKY, A. e LYZENGA, G. - "Solving Finite Element Equations on Concurrent Computers" - Symposium on Parallel Computations and their Impact on Mechanics, 13 a 18 de dezembro, (1987).
- [9] PARK, K.C. e OMID, B.N. - "Solving Structural Mechanics Problems on the Caltech Hipercube Machine" - Computer Methods in Applied Mechanics and Engineering , Vol. 61 , pp. 161-176, (1987).
- [10] ORTEGA, J.M. - "Introduction to Parallel and Vector Solution of Linear System" - Plenum Press, New York, (1988).
- [16] BATHE, K.J. , Finite Element Procedures in Engineering Analysis - Prentice hall, Englewood Cliffs, N.J., (1982).
- [11] BARROS, C.O. , "O Sistema Multiprocessador A.C.P. no C.B.P.F." , Simposio Brasileiro de Arquitetura de Computadores - Processamento Paralelo , II SBAC-PP , 26 a 28 de setembro, Águas de Lindóia (S.P.), (1988).

- [12] FISCHLER ,M. e outros - "Software for Event Oriented Processing on Multiprocessor System" - Symposium on Recent Developments in Computing , 8 a 11 de agosto,(1984).
- [13] FISCHLER, M. e outros - "The A.C.P. Multiprocessor System at Fermilab" - Computer Physics Communication - Vol 45 , pp. 323-329, (1987).
- [14] GOMES, C.C. - "A Utilização dos Vetores de Lanczos-Ritz na Condensação Dinâmica de Estruturas" - Tese M.Sc , COPPE/UFRJ, (1988).
- [15] LIMA, E.C.P. , COUTINHO, A.L.G.A. e outros - "Dynamic Substructuring Analysis using Enhanced Lanczos-Ritz Vectors" - Proc. 2nd Int. Conf. on Numerical Methods in Engineering: Theory and Applications, NUMETA 1987 Edt. G.N. Pande, J. Middleton, Martinus Nijhoff Publishers - T38, (1987).