

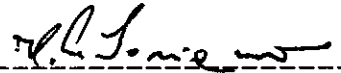
NOVAS TÉCNICAS DE PROJETO DE SISTEMAS

PARA ANÁLISE ESTRUTURAL

Sérgio Roberto Ferreira de Carvalho

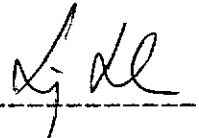
TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE
PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE
JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO
GRAU DE MESTRE EM CIÊNCIAS (M.Sc.) EM ENGENHARIA CIVIL

Aprovada por :



Prof. Humberto Lima Soriano


Presidente



Prof. Luiz Landau



Prof. Marcelo Gattass



Prof. Eber Assis Schmitz

RIO DE JANEIRO - BRASIL

MARÇO 1988

GARVALHO, SÉRGIO ROBERTO FERREIRA DE
Novas Técnicas de Projeto de Sistemas
para Análise Estrutural [Rio de Janeiro]
1988.

VI, 181p. 29,7 cm (COPPE/UFRJ, M.Sc,
Engenharia Civil, 1988)

Tese - Univ. Fed. Rio de Janeiro, COPPE

I. Projeto de Sistemas Computacionais
Para Análise Estrutural I. COPPE/UFRJ
II. Título (série)

A meus pais

Edival Ponciano de Carvalho e

Márcia Ferreira de Carvalho

Ao Prof. Soriano, pela dedicação e constante incentivo, aos demais professores, funcionários e companheiros da COPPE-CIVIL, aos meus amigos Roberto Rodrigues e Eber Schmitz, pelo apoio na análise e projeto, aos amigos Manoel Justino, Sérgio Hampshire, Tadeu Guimarães e Eduardo Thomaz da PROMON,

agradeço o apoio e a atenção recebidos

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

NOVAS TÉCNICAS DE PROJETO DE SISTEMAS
PARA ANÁLISE ESTRUTURAL

Sérgio Roberto Ferreira de Carvalho

Março 1988

Orientador : Humberto Lima Soriano

Programa : ENGENHARIA CIVIL

Este trabalho trata da aplicação de novas metodologias de geração de software na criação de sistemas computacionais para análise estrutural. Assim, na presente análise estática e dinâmica de estruturas reticuladas foi usada a Análise Estruturada, o Projeto Estruturado e, finalmente, linguagem Pascal.

Estas técnicas têm sido extensamente usadas em sistemas computacionais no mundo inteiro com excelentes resultados e sua aplicação em sistemas científicos, como atesta o presente trabalho, se mostra igualmente promissora.

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

NEW STRUCTURAL ANALYSIS SYSTEMS
DESIGN TECHNIQUES

Sérgio Roberto Ferreira de Carvalho

March 1988

Chairman : Humberto Lima Soriano
Department : CIVIL ENGINEERING

The application of new methodologies for structural analysis software generation is treated in this work. Tools of Structured Systems Analysis and Structured Design were used and finally the Pascal language was employed for the implementation.

These techniques have been used extensively into computational systems in the whole world with excellent results and the application to scientific systems is shown equally promising.

INDICE

I	- Introdução	1
II	- Análise Estática e Dinâmica de Estruturas Reticuladas	2
II.1	- Fundamentos da Análise Estática	2
II.2	- Fundamentos da Análise Dinâmica	4
III	- Análise Estruturada de Sistemas	7
III.1	- Descrição	7
III.2	- Simbologia usada	10
III.3	- Diagramas do Sistema para Análise Estática	11
III.4	- Diagramas do Sistema para Análise Dinâmica	13
IV	- Projeto Estruturado de Sistemas	15
IV.1	- Descrição	15
IV.2	- Simbologia Usada	19
IV.3	- Projeto do Sistema Para Análise Estática	20
IV.4	- Projeto do Sistema Para Análise Dinâmica	47
V	- Programação	76
V.1	- Estruturação de Dados	76
V.1.1	- Estruturação Tradicional	76
V.1.1.1	- Crítica à Estruturação Tradicional	77
V.1.1.1.1	- Difícil Entendimento	77
V.1.1.1.2	- Edição de Dados	77
V.1.1.1.3	- Gerenciamento do Uso da Memória	77
V.1.1.1.4	- Manutenção e Expansões	77
V.1.2	- Modelos Propostos	78
V.1.2.1	- Registro	78
V.1.2.2	- Vetor de Registros	79
V.1.2.3	- Vetor de Ponteiros para Registros	80
V.1.2.4	- Listas Encadeadas com Ponteiros	81
V.1.3	- Proposta para Análise Estrutural	83
V.2	- Principais Algoritmos Usados	87
V.2.1	- Montagem da Matriz de Rigidez Global	87
V.2.2	- Blocagem da Matriz de Rigidez	89
V.2.3	- Resolução do Sistema de Equações	91
V.2.3.1	- Triangularização	92
V.2.3.2	- Substituição	93
V.2.3.3	- Retrosubstituição	93
V.2.4	- Ortogonalização e normalização dos Vetores de Ritz	94
V.2.5	- Resolução do Sistema de Equações Diferenciais	96
VI	- Exemplos e Conclusões	98
VI.1	- Exemplos	98
VI.1.1	- Exemplo 1	98
VI.1.2	- Exemplo 2	102
VI.1.3	- Exemplo 3	106
VI.2	- Conclusões	113
VI.3	- Sugestões	114
VII	- Bibliografia	116
Apêndice I	- Listagem da Rotina Gráfica	118
Apêndice II	- Listagem da Rotina Estática	122
Apêndice III	- Listagem da Rotina Dinâmica	155

Capítulo I - Introdução

Os computadores têm sido ferramentas muito valiosas para a engenharia desde as primeiras unidades construídas. O acelerado desenvolvimento da tecnologia de circuitos integrados tem permitido o surgimento de máquinas cada vez mais poderosas e acessíveis. À medida que o hardware se torna mais poderoso e confiável, as atenções se voltam para a geração e manutenção do software, tarefas a cada dia mais desafiadoras devido à crescente sofisticação e complexidade do mesmo.

O objetivo do presente trabalho é abordar a aplicação de novas metodologias de geração de software na criação de sistemas computacionais para análise estrutural. Foram criados dois programas, um para análise estática linear de estruturas reticuladas e outro para análise dinâmica linear de estruturas reticuladas. Em cada sistema, usou-se inicialmente a técnica de Análise Estruturada com o objetivo de definir os dados e processos básicos envolvidos na solução do problema. A etapa seguinte foi o Projeto Estruturado, na qual se fez o projeto dos módulos que compõe o sistema, bem como o relacionamento entre os mesmos. Finalmente, a partir do projeto, programou-se o sistema em Pascal, uma linguagem estruturada de alto nível, de uso geral e que se mostrou muito adequada para programas de análise estrutural.

Capítulo 11 - Análise Estática e Dinâmica de Estruturas Reticuladas

11.1 - Fundamentos da Análise Estática

A análise estática de estruturas reticuladas é assunto muito conhecido em análise estrutural. Tem-se uma estrutura idealizada como um conjunto de elementos de barra interligados e sujeita a cargas nestes elementos. Deseja-se calcular os deslocamentos e esforços impostos à estrutura pelo carregamento aplicado.

Adotando-se a hipótese das seções planas, os deslocamentos de um ponto qualquer em uma seção transversal de elemento da estrutura podem ser perfeitamente determinados a partir dos deslocamentos do centro de gravidade da seção. Desta forma, o elemento estrutural fica idealizado no lugar geométrico dos centros de gravidade de suas seções, constituindo um modelo unidimensional, conforme SORIANO (3) 1983.

Uma estrutura reticulada é discretizada em um conjunto de elementos unidimensionais, cujo comportamento pode ser perfeitamente definido a partir dos deslocamentos de seus pontos nodais, que são interseções entre dois ou mais destes elementos unidimensionais, extremidade de elementos ou quaisquer outros pontos de interesse. Temos então o seguinte sistema de equações algébricas lineares de equilíbrio:

$$K d = f \quad 11.1$$

onde K é a matriz de rigidez da estrutura, d é o vetor de deslocamentos nodais e f o vetor de forças nodais. A matriz de rigidez do modelo é montada a partir das matrizes individuais dos elementos de barra dos quais a estrutura é constituída. O vetor de forças é formado pelas forças aplicadas diretamente aos pontos nodais somadas a forças fictícias que produzem o mesmo efeito das ações aplicadas às barras da estrutura.

Ao resolvermos a equação (11.1), obtemos os deslocamentos

nodais resultantes do carregamento aplicado. A partir destes deslocamentos podemos calcular as reações de apoio da estrutura e os esforços de extremidade das barras que compõem a mesma. Os esforços de extremidade são calculados por meio da equação

$$a_{1f,i} = K_{1,i} d_{1,i} + a_{1,i} \quad 11.2$$

onde temos que o vetor de esforços de extremidade da barra i no referencial local da barra é a soma do produto da matriz de rigidez da barra i no referencial local com os deslocamentos nodais de extremidade no referencial local e os esforços de engastamento perfeito no mesmo referencial, de acordo com SORIANO (3) 1983.

As reações de apoio podem ser calculadas usando-se a técnica do número grande, conforme descrito por SORIANO (5) 1985.

11.2 - Fundamentos da Análise Dinâmica

Seja a mesma estrutura anterior, porém sujeita à ação de cargas de amplitude variável segundo a função $g(t)$ no tempo. O comportamento da estrutura pode ser representado pelo seguinte sistema de equações diferenciais de equilíbrio :

$$M\ddot{d} + C\dot{d} + Kd = f g(t) \quad 11.3$$

onde M, C e K são matrizes de ordem N , respectivamente as matrizes de massa, amortecimento e rigidez da estrutura. Os vetores d, \dot{d} e \ddot{d} variam com o tempo e são as acelerações, velocidades e deslocamentos nodais, respectivamente. O vetor f é o vetor de cargas já descrito e $g(t)$ uma função que define a variação da amplitude das cargas aplicados no tempo .

As incógnitas do problema são as acelerações, velocidades, deslocamentos, reações de apoio e esforços de extremidade da estrutura ao longo do tempo. São conhecidas várias maneiras de resolução do problema proposto porém, para efeito deste texto, a solução do problema pode ser dividida em etapas distintas. Inicialmente gera-se um outro sistema de equações diferenciais em um espaço vetorial mais reduzido que represente satisfatoriamente o comportamento da estrutura. A seguir, desacopla-se este sistema reduzido de forma que se tenha um conjunto de equações diferenciais desacopladas. Resolve-se então estas equações, obtendo-se as acelerações, velocidades e deslocamentos generalizados do sistema reduzido. Por meio de uma transformação linear de mudança de referencial, obtém-se as acelerações, velocidades e deslocamentos nodais da estrutura no tempo. Os esforços de extremidade das barras e as reações de apoio são então calculadas a partir dos deslocamentos já obtidos.

O sistema reduzido foi obtido por meio de algoritmos apresentados por WILSON e alii (2), e, segundo esta referência, representa de forma adequada o comportamento da estrutura com um numero razoavelmente pequeno de vetores. A seguir apresenta-se um resumo desta formulação. Seja

$$d = X u \quad 11.4$$

onde u é o vetor de deslocamentos reduzidos e X uma matriz $N \times L$ que define uma transformação linear de mudança do espaço global de dimensão N para o espaço aqui denominado reduzido de dimensão L .

Seja

$$\begin{aligned} M^* &= X^t M X = I \\ C^* &= X^t C X \\ K^* &= X^t K X \\ f^* &= X^t f \end{aligned} \quad 11.5$$

Tem-se o seguinte sistema reduzido

$$M^* \ddot{u} + C^* \dot{u} + K^* u = f^* g(t) \quad 11.6$$

de ordem L bem menor que o sistema original. Em seguida, resolve-se o problema de autovalor

$$[K^* - w_i^2 I] z_i = 0 \quad 11.7$$

onde w_i^2 é um dos autovalores do sistema reduzido e w_i uma frequência aproximada da estrutura analisada. Este problema de autovalor foi resolvido por meio do método de Jacobi, conforme descrito por BATHE (1), obtendo-se os autovalores e autovetores do sistema reduzido.

A partir dos autovetores z_i do sistema reduzido, obtém-se uma nova base

$${}^0X = XZ \quad 11.8$$

que será usada para o cálculo da resposta do sistema completo a partir da resposta do sistema reduzido desacoplado, que pode ser escrito de uma maneira geral como um conjunto de equações do tipo

$$x_i(t) + 2 c_i w_i \dot{x}_i(t) + w_i^2 x_i(t) = P_i g(t) \quad i = 1..L$$

onde c_i é a porcentagem do amortecimento crítico associado ao vetor i da base 0X . A solução do sistema foi obtida através da integração exata de cada equação diferencial ao longo de trechos de variação linear de carga, como apresentada por LANDAU (4).

Capítulo III - Análise Estruturada de Sistemas

III.1 - Descrição

Uma das propostas existentes atualmente para desenvolvimento de software divide o ciclo de vida de um sistema nas seguintes fases : Levantamento de Requisitos, Análise Estruturada ou Especificação Funcional, Projeto Estruturado, Programação e Validação do sistema. A idéia básica é a mesma da engenharia, ou seja, dividirmos o ciclo de vida de um sistema de forma que tenhamos profissionais especializados executando funções bem definidas e produzindo documentos claros de forma que possam ser utilizados pelos profissionais da próxima etapa.

Assim como em Engenharia Civil temos o Arquiteto, o Engenheiro Projetista e o Engenheiro de Construção, nesta metodologia temos o Analista, o Projetista e o Programador. O Analista executará a etapa de Análise Estruturada e gerará a Especificação Funcional, o Projetista fará o Projeto Estruturado a partir desta especificação e, finalmente, o Programador codificará o programa a partir do Projeto.

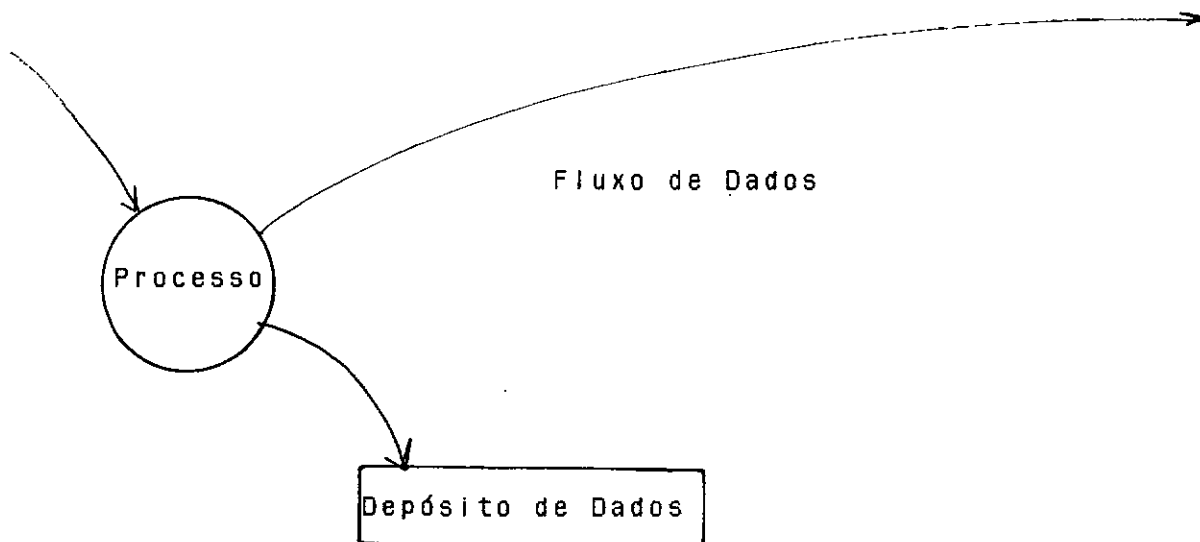
A Análise Estruturada de Sistemas é composta por um conjunto de técnicas usadas objetivando uma especificação lógica do problema e de sua resolução. Neste trabalho foi usada uma ferramenta denominada Diagrama de Fluxo de Dados, que tem por finalidade representar a nível lógico os dados e principais processos envolvidos na solução do problema em estudo. Estes diagramas, produtos da análise, são utilizados como entrada para a etapa seguinte de análise, que é o projeto estruturado. Os elementos componentes destes diagramas são descritos detalhadamente por GANE (8). Os arquivos do sistema também foram normalizados. A estrutura de dados adotada será analisada em outro capítulo.

Conforme descrito nesta referência, há diversas outras ferramentas que podem ser usadas para fazer-se uma especificação lógica de um sistema, tais como o Dicionário de Dados, Tabelas de Decisão, Pseudo-código e Português Estruturado. Aparentemente, o uso extensivo destas ferramentas se mostra mais necessário na especificação de sistemas comerciais, onde têm-se, tipicamente, uma grande quantidade e variedade de entidades externas, depósitos e fluxos de dados a manipular e um processamento relativamente simples com os mesmos. No caso de sistemas científicos, usualmente têm-se uma pequena quantidade de dados e um processamento muito pesado sobre os mesmos, o que sugere uma tendência do Projeto Estruturado, visto mais adiante, tornar-se a etapa mais importante do desenvolvimento de programas científicos. Por este motivo preferiu-se aplicar-se um esforço maior na etapa de Projeto Estruturado, o que não significa que a etapa de Análise Estruturada não seja importante, inclusive para sistemas científicos.

Segundo GANE (8), podemos resumir os passos envolvidos na criação de um Diagrama de Fluxo de Dados como se segue :

- Identificar as entidades externas envolvidas.
- Identificar as entradas e saídas programadas que podem ser esperadas no curso normal do processo.
- Identificar as consultas e pedidos de informação que possam surgir.
- Desenhar as entidades externas, os fluxos de dados que surgem, os processos logicamente necessários e os depósitos de dados que pareçam necessários.
- Desenhar um primeiro esboço a mão livre e concentrar-se em tudo, exceto erros, exceções e decisões.
- Aceitar o fato de que serão necessários, pelo menos, três esboços do Diagrama de nível superior.
- Quando o primeiro esboço estiver pronto, verificar se todas as entradas e saídas listadas estão incluídas, exceto aquelas que tratam de erros e exceções.
- Produzir um segundo esboço muito claro usando régua com gabaritos com símbolos.
- Rever o segundo esboço com um representante do usuário que esteja disposto a colaborar.
- Produzir uma expansão de nível inferior para cada processo definido no segundo esboço. Resolver tratamento de erros e exceções e se necessário incorporar as modificações no Diagrama de nível de topo.

III.2 - Simbologia usada



Seta
----->

Fluxo de dados

Círculo

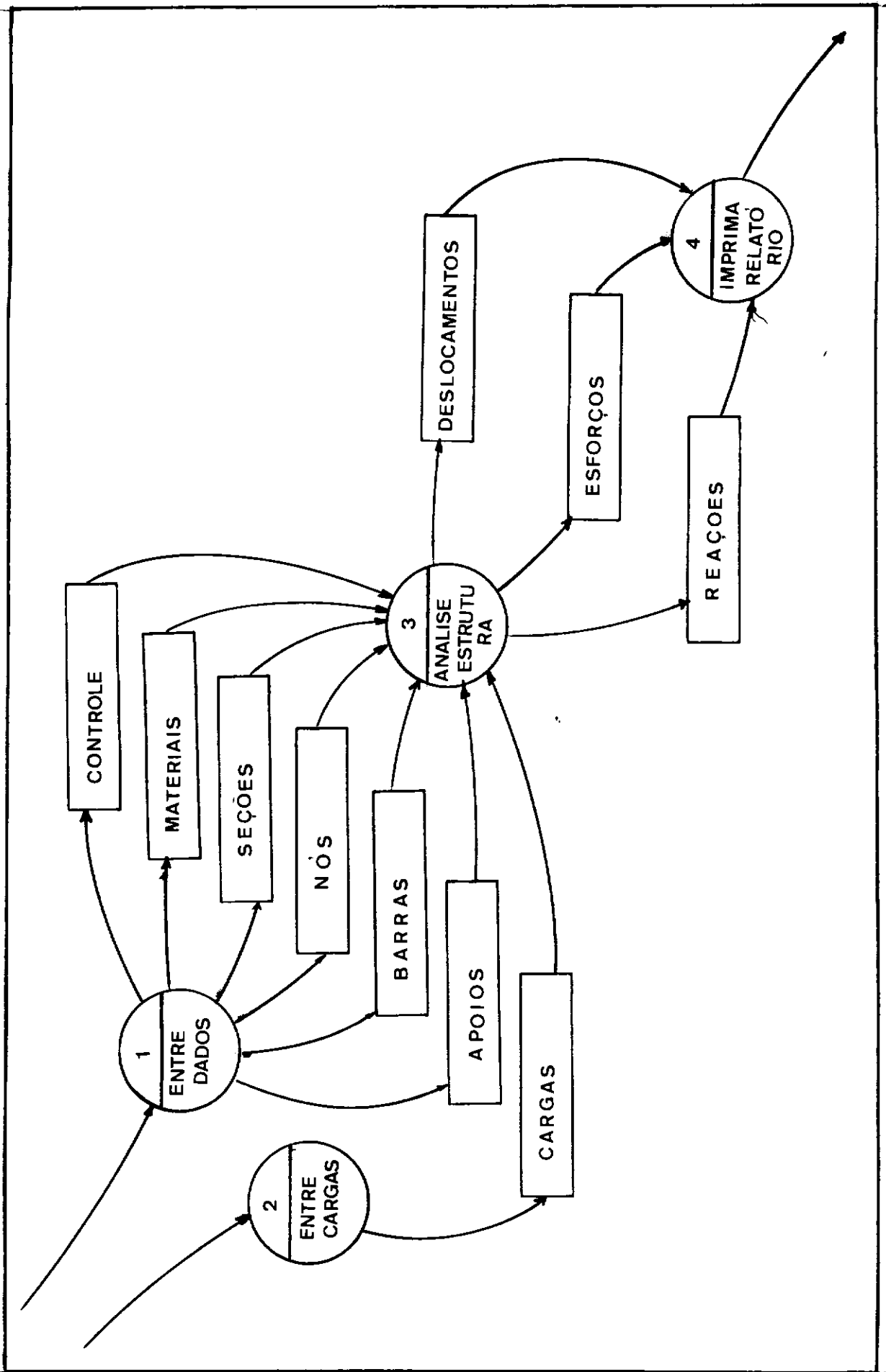
Processo que transforma fluxos de dados

Retângulo

Depósitos de dados

III.3 - Diagramas do Sistema para Análise Estática

Diagrama 0



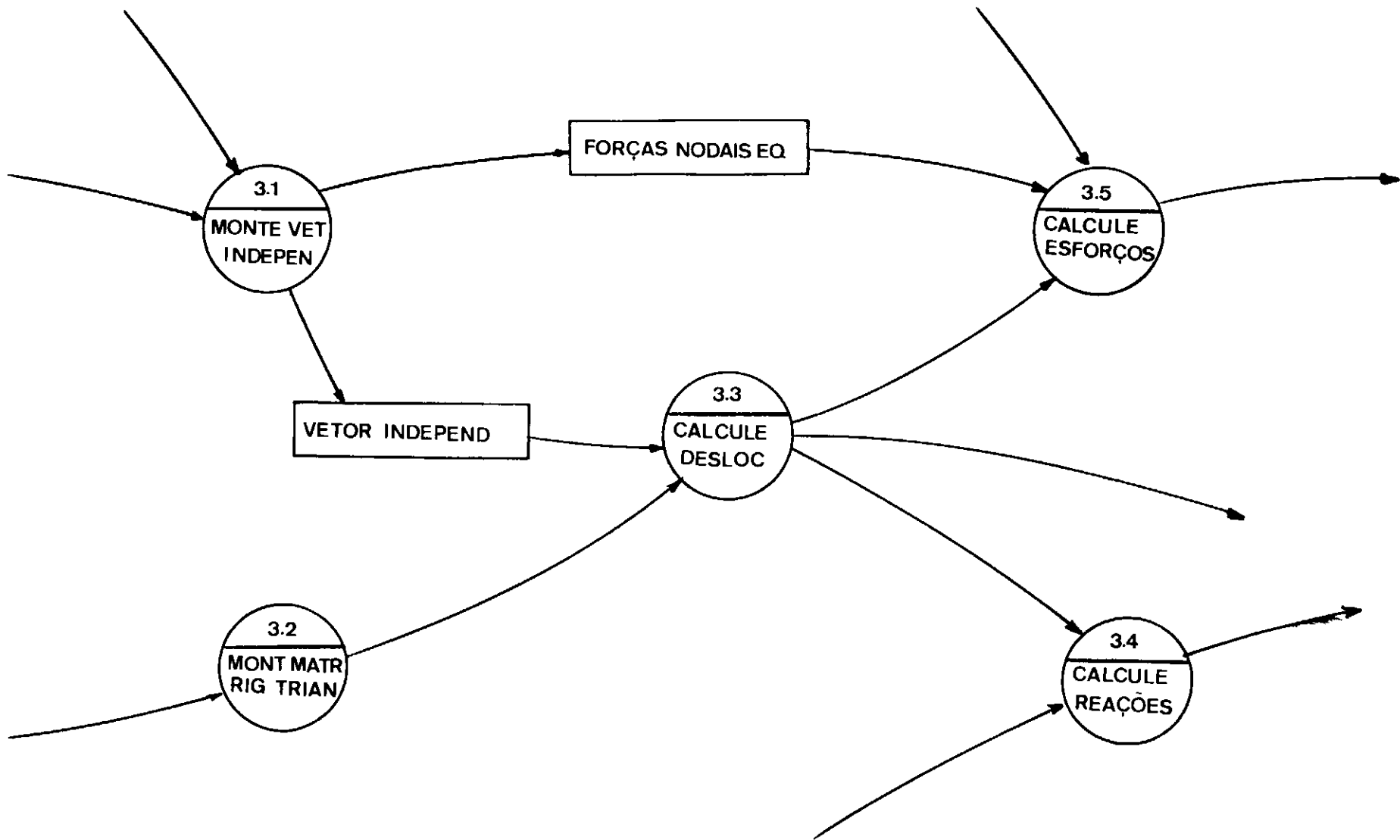
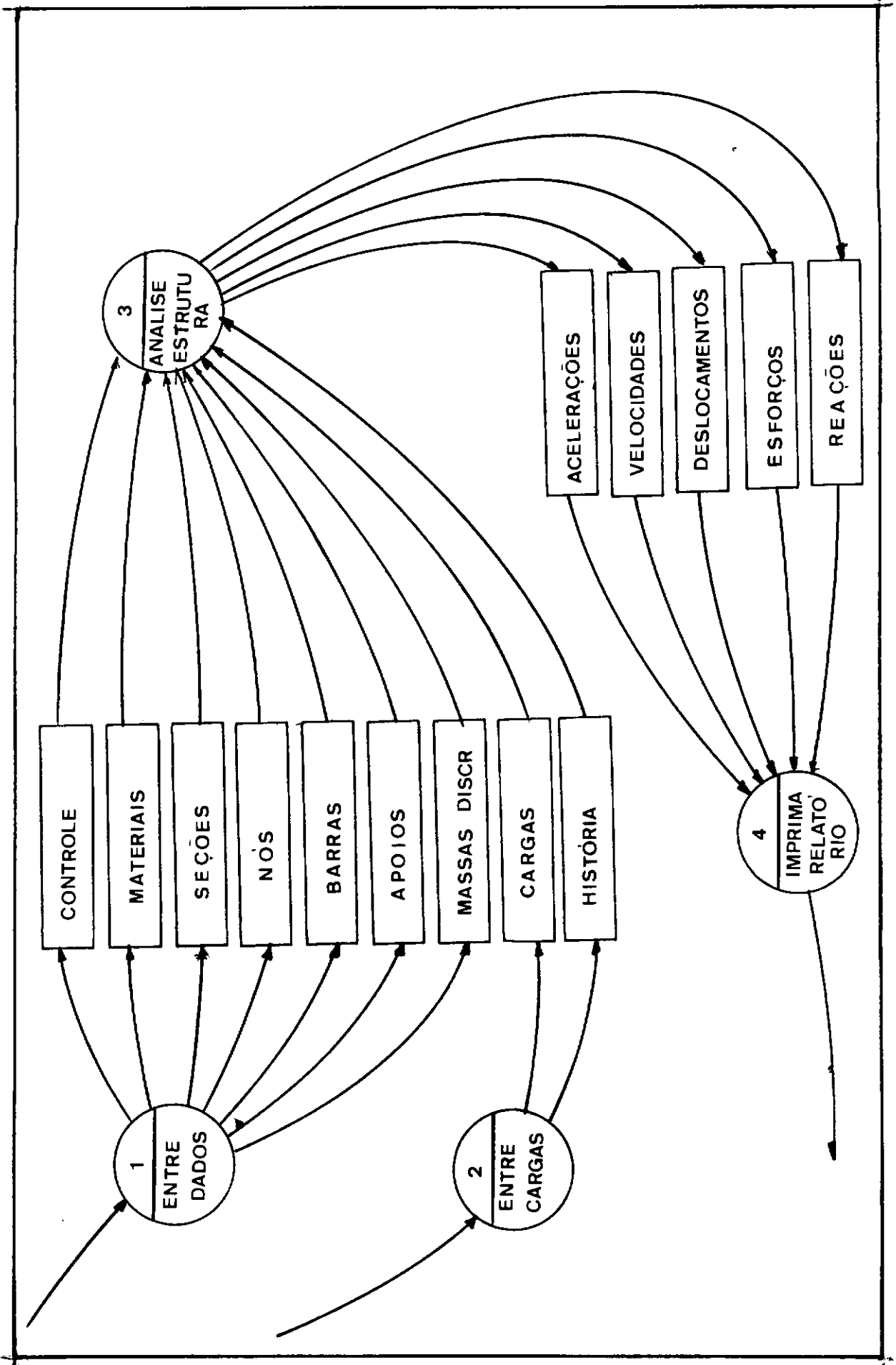


Diagrama 3

III.4 - Diagramas do Sistema para Análise Dinâmica

Diagrama 0



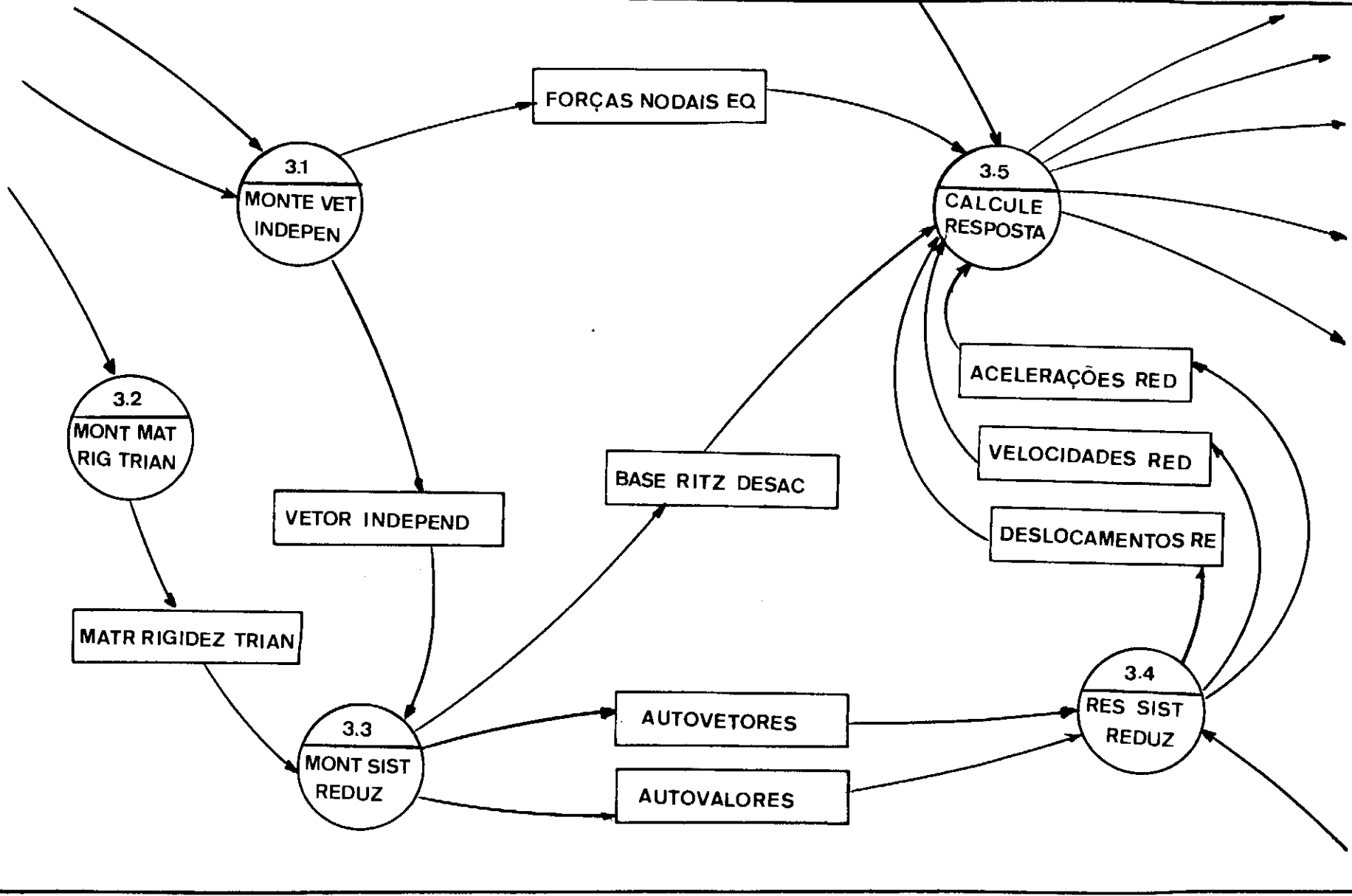


Diagrama 3

Capítulo IV - Projeto Estruturado de Sistemas

IV.1 - Descrição

A etapa de Projeto Estruturado tem como entrada a Especificação Funcional gerada pela Análise Estruturada e tem por finalidade gerar o Projeto Estruturado, que é composto basicamente por Diagramas de Estruturas Modulares. Estes diagramas em essência especificam os módulos que compõem o sistema bem como o relacionamento e posicionamento hierárquico dos mesmos. Podemos, simplificarmente, dividir a etapa de Projeto em três fases : derivação do projeto lógico, detalhamento e críticas e melhoramentos do diagrama.

A idéia central é a funcionalidade. Cada módulo deve executar uma função bem definida. Por exemplo, em um automóvel temos diversos módulos funcionais, tais como motor, suspensão, etc.. Inclusive o motor pode ser dividido em diversas partes funcionais, tais como carburador, velas, bateria, etc. Uma vantagem de termos módulos funcionais é que estes módulos podem ser substituídos por outros que tenham interface e função equivalentes sem que isto afete o funcionamento do todo. O mesmo carburador pode ser utilizado em diversos motores e o mesmo motor pode usar diversos carburadores diferentes.

Para que melhor se compreenda a técnica de projeto estruturado, é conveniente examinar por que o uso de estruturas modulares tende a reduzir o custo de desenvolvimento e manutenção dos programas. A resolução de um problema é mais difícil quando se considera simultaneamente todos os seus aspectos. É mais fácil se se puder achar uma maneira de resolver partes relativamente independentes do problema, uma a uma. Este princípio aparece na programação, quando observa-se que dobrando o tamanho de um programa, mais que duplica o tempo de implementá-lo. Por isso, projetando um programa em várias partes, em lugar de uma, pode-se reduzir o esforço total se as partes forem relativamente independentes.

As técnicas usadas no Projeto facilitam a construção de programas de forma a reduzir o esforço necessário para implementar e manter esses programas. Com o projeto estruturado gasta-se menos tempo para produzir-se o código do programa, sendo este mais alterável, flexível e mais facilmente reutilizável do que para produzir um programa monolítico sem estas características desejáveis.

Esta metodologia recomenda que se divida o programa em módulos unifuncionais cujos códigos possam, de preferência, ser escritos em uma única página de texto. Têm-se vantagens, pois escrever um módulo de uma página é mais simples que escrever uma página de um programa maior. Os programas feitos de forma estruturada são mais simples; portanto, tendem a ser executados tão ou mais rapidamente que os programas monolíticos.

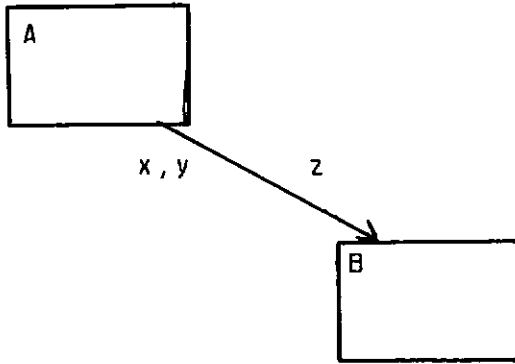
Duas grandezas são de especial importância para avaliar quais as boas alternativas de estruturas modulares. A primeira é o acoplamento, que é a medida de relações entre módulos. O objetivo é obter a máxima independência entre os mesmos, ou seja, o mínimo acoplamento. A segunda grandeza é a ligação, que mede o grau de relacionamento entre os elementos dentro de um módulo. Como todos os programas possuem elementos de código que são altamente relacionados, é conveniente identificá-los e agrupá-los no mesmo módulo. Se os elementos dependentes de código estiverem em módulos diferentes, os módulos serão altamente relacionados. Os conceitos de ligação e acoplamento medem, na realidade, a mesma coisa a partir de pontos de vista distintos.

O objetivo do projeto estruturado é montar um programa como uma estrutura de módulos unifuncionais independentes. O programa resultante possui as seguintes características :

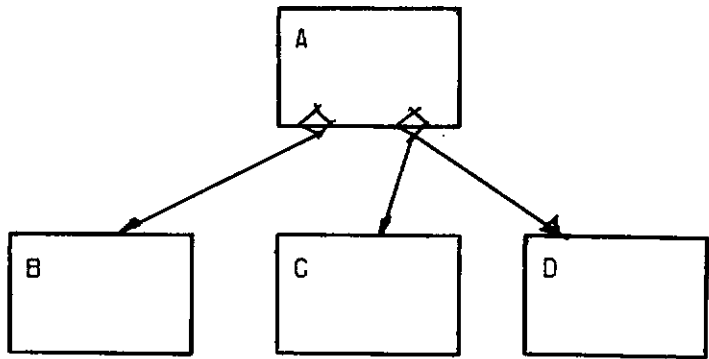
- O programa é mais simples.
- O programa pode ser escrito, entendido, testado e alterado parte a parte.
- Há menos possibilidade de se cometerem erros.
- Os efeitos colaterais das alterações reduzem-se drasticamente.
- Podem-se aplicar esforços de otimização a áreas críticas.
- Os módulos unifuncionais podem ser reutilizados em outros programas.

O uso desta técnica, descrita por STEVENS (7), mostrou-se muito adequada para o desenvolvimento de sistemas computacionais de análise estrutural. Além da facilidade encontrada no projeto em si, os diagramas gerados são muito importantes como documentação do programa gerado, servindo não só para eventuais modificações e acréscimos, como também para a eventualidade de reprogramar o sistema em outra linguagem, o que seria praticamente inviável sem o projeto.

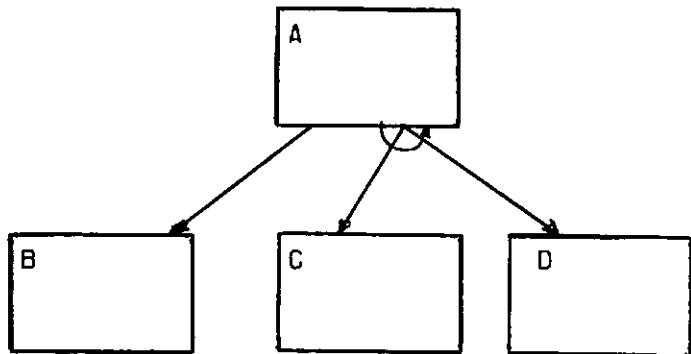
V.2 - Simbologia Usada



Módulo A chama Módulo B

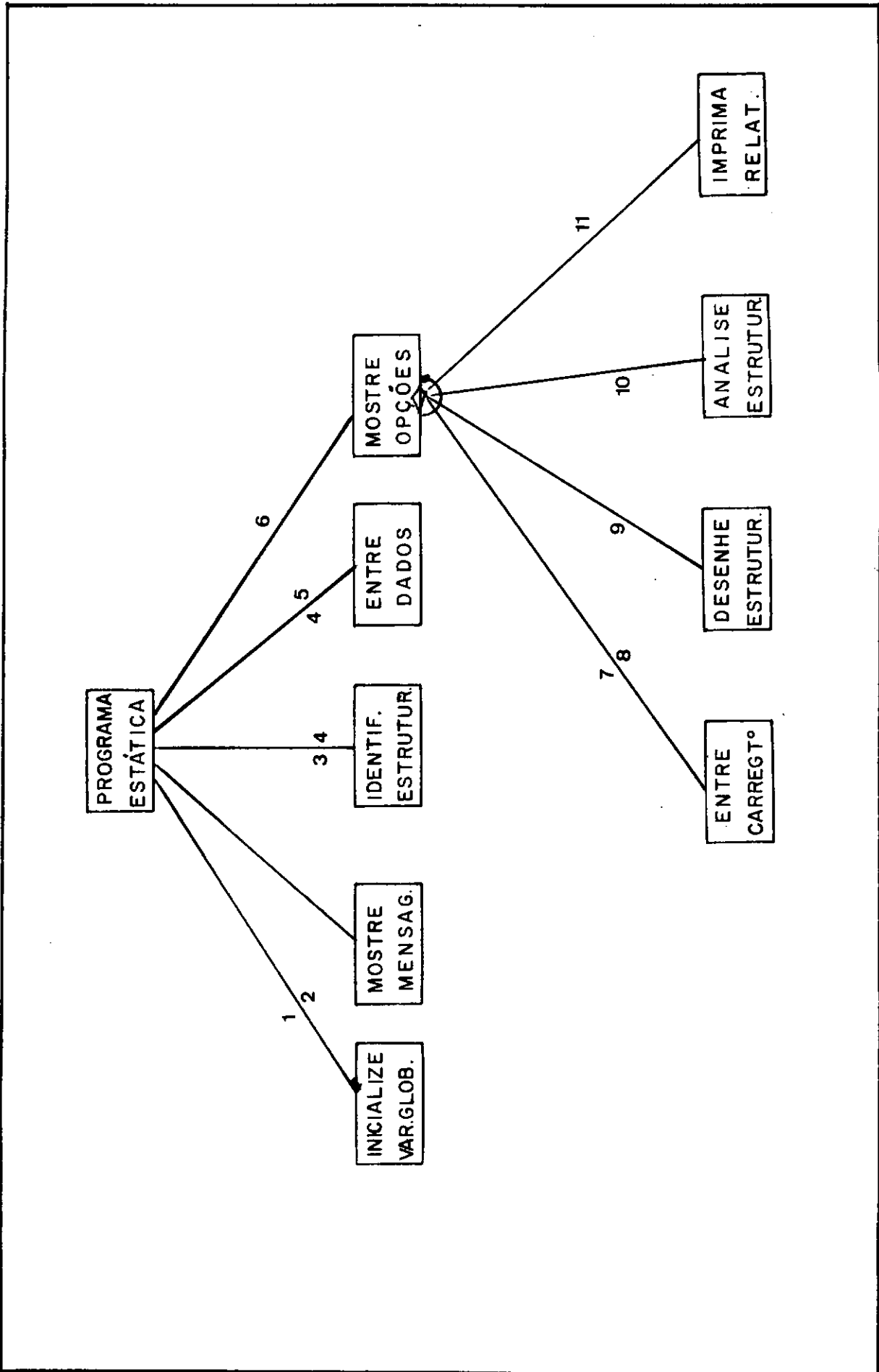


Chamadas condicionais



Chamadas iterativas e singulares

IV.3 - Projeto do Sistema Para Análise Estática - figura IV.1



Parâmetros da figura IV.1

- 1 - Variáveis Globais
- 2 - Variáveis Globais Inicializadas
- 3 - Drive+Nome
- 4 - Drive+Nome
- 5 - Modelo+Controle
- 6 - Drive+Nome+Modelo+Controle
- 7 - Drive+Nome+Modelo+Controle
- 8 - Controle
- 9 - Drive+Nome+Modelo+Controle
- 10 - Drive+Nome+Modelo+Controle
- 11 - Drive+Nome+Modelo

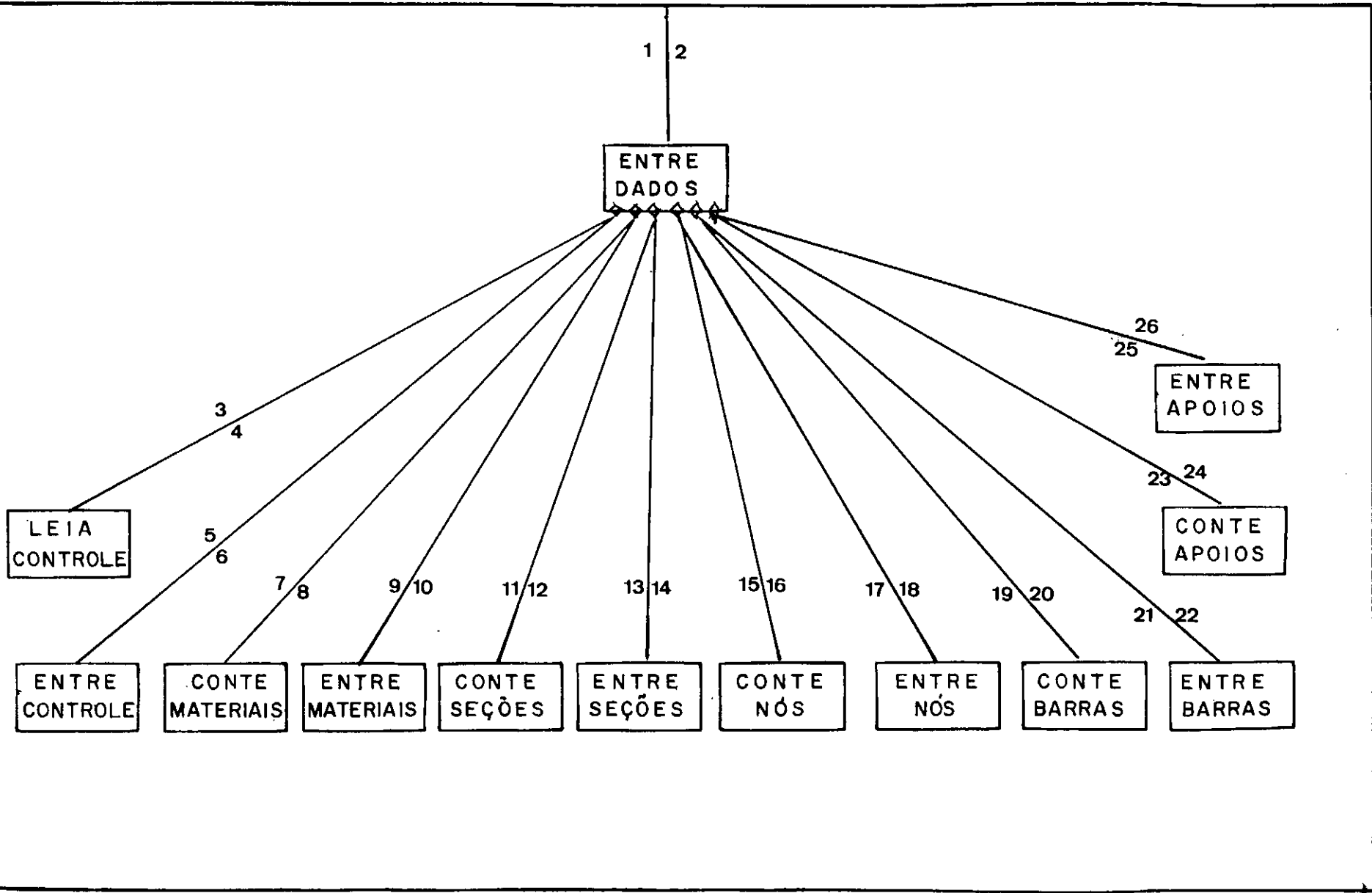


Figura IV.2

Parâmetros da figura IV.2

- 1 - Drive+Nome
- 2 - Drive+Nome+Modelo+Controle
- 3 - Drive+Nome
- 4 - Drive+Nome
- 5 - Modelo+Controle
- 6 - Número de Materiais
- 7 - Drive+Nome
- 8 - Número de Materiais
- 9 - Drive+Nome+Controle
- 10 - Número de Seções
- 11 - Drive+Nome+Controle
- 12 - Número de Seções
- 13 - Drive+Nome+Controle
- 14 - Número de Nós
- 15 - Drive+Nome+Controle
- 16 - Número de Nós
- 17 - Drive+Nome+Controle
- 18 - Número de Barras
- 19 - Drive+Nome+Controle
- 20 - Número de Barras
- 21 - Drive+Nome+Controle
- 22 - Número de Apoios
- 23 - Drive+Nome+Controle
- 24 - Número de Apoios
- 25 - Drive+Nome+Controle
- 26 - Número de Apoios

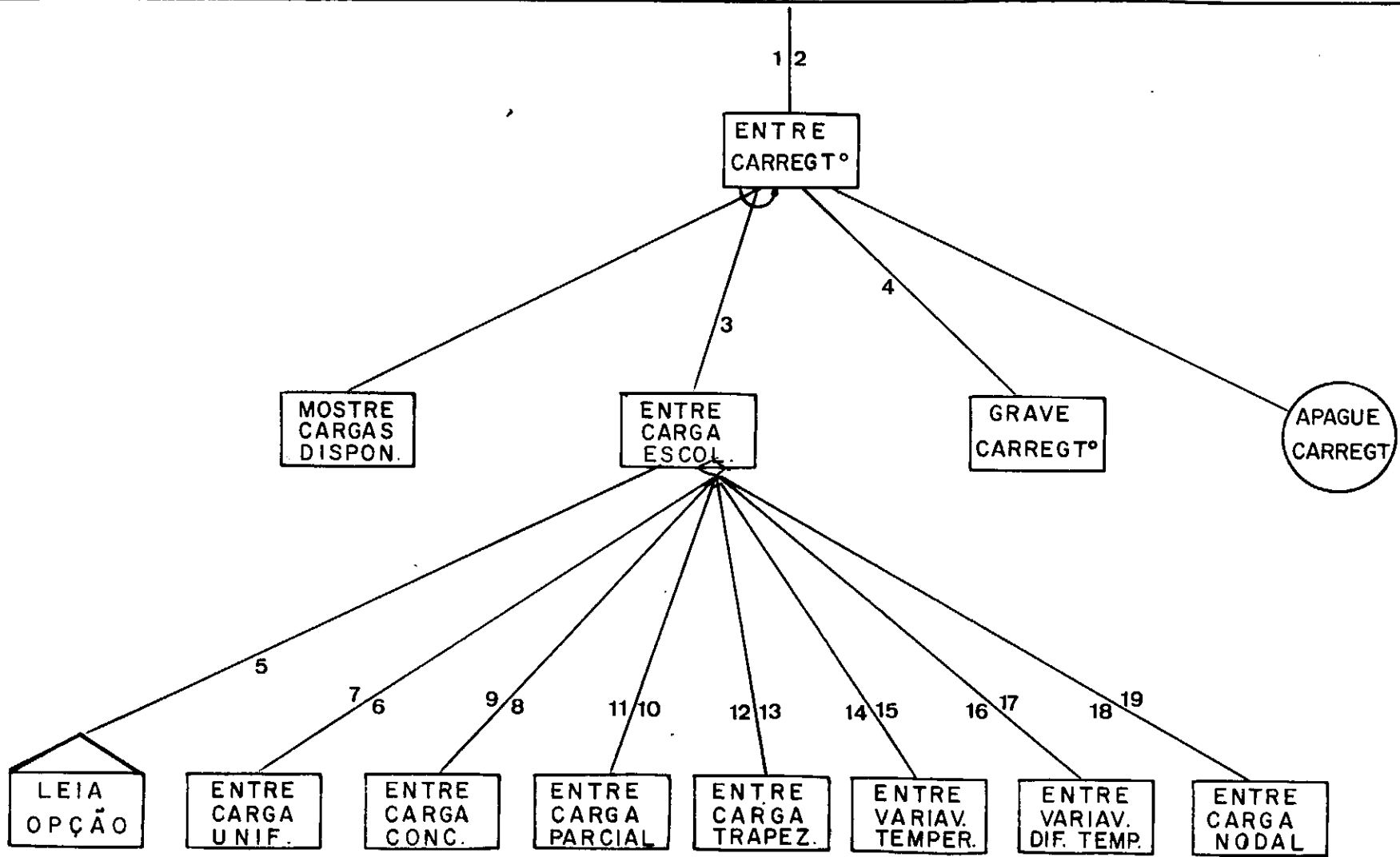


figura IV.3

Parâmetros da figura IV.3

- 1 - Controle
- 2 - Drive+Nome+Modelo+Controle
- 3 - Cargas
- 4 - Drive+Nome+Cargas
- 5 - Opção
- 6 - Carga
- 7 - Modelo
- 8 - Carga
- 9 - Modelo
- 10 - Carga
- 11 - Modelo
- 12 - Carga
- 13 - Modelo
- 14 - Carga
- 15 - Modelo
- 16 - Carga
- 17 - Modelo
- 18 - Carga
- 19 - Modelo

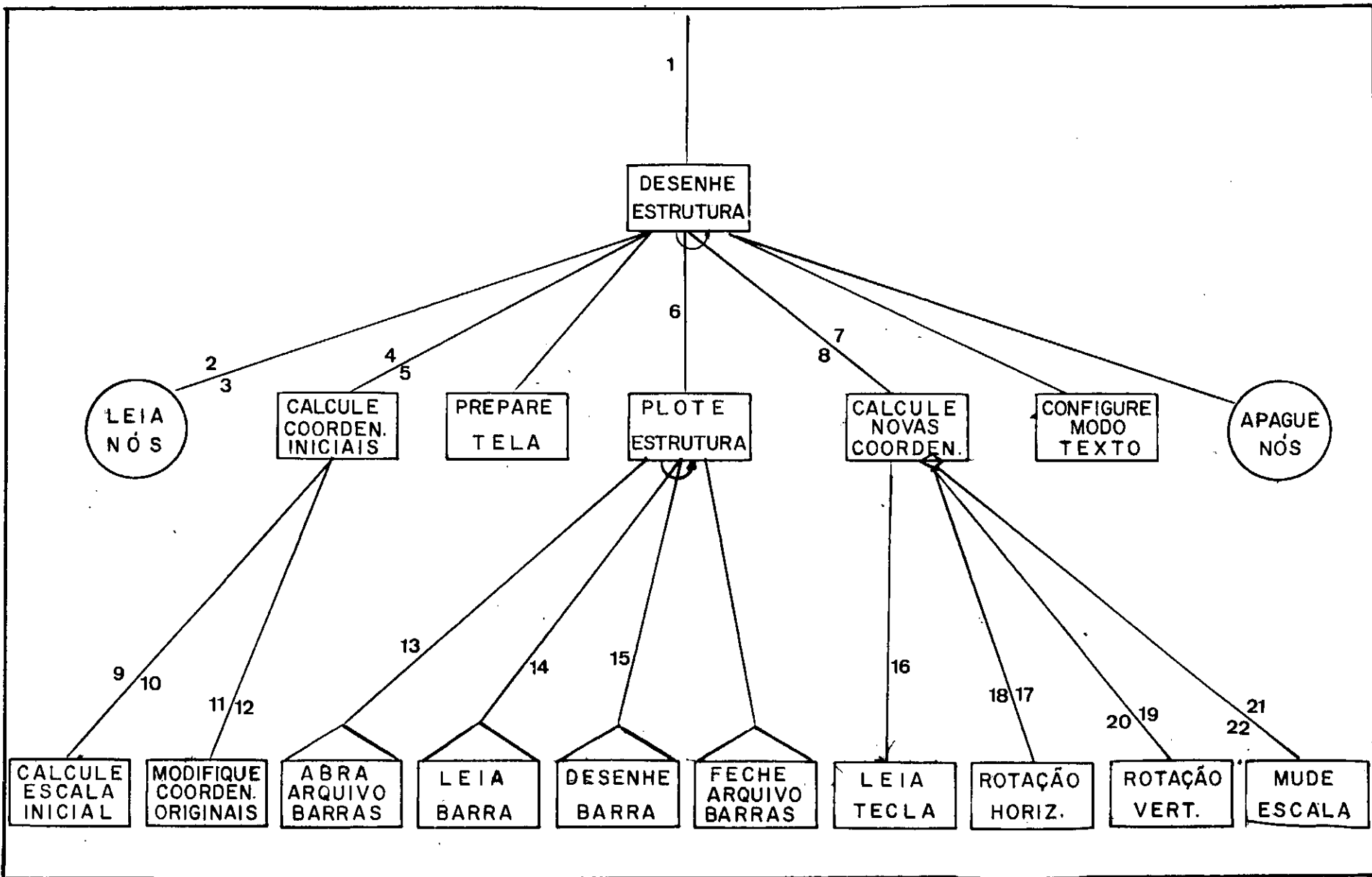


Figura IV.4

Parâmetros da figura IV.4

- 1 - Drive+Nome+Controle
- 2 - Drive+Nome
- 3 - Nós
- 4 - Nós
- 5 - Nós
- 6 - Drive+Nome+Nós
- 7 - Nós
- 8 - Nós
- 9 - Nós
- 10 - Escala
- 11 - Nós+Escala
- 12 - Nós
- 13 - Drive+Nome
- 14 - Barra
- 15 - Barra+Nós
- 16 - tecla
- 17 - Nós
- 18 - Nós
- 19 - Nós
- 20 - Nós
- 21 - Nós
- 22 - Nós

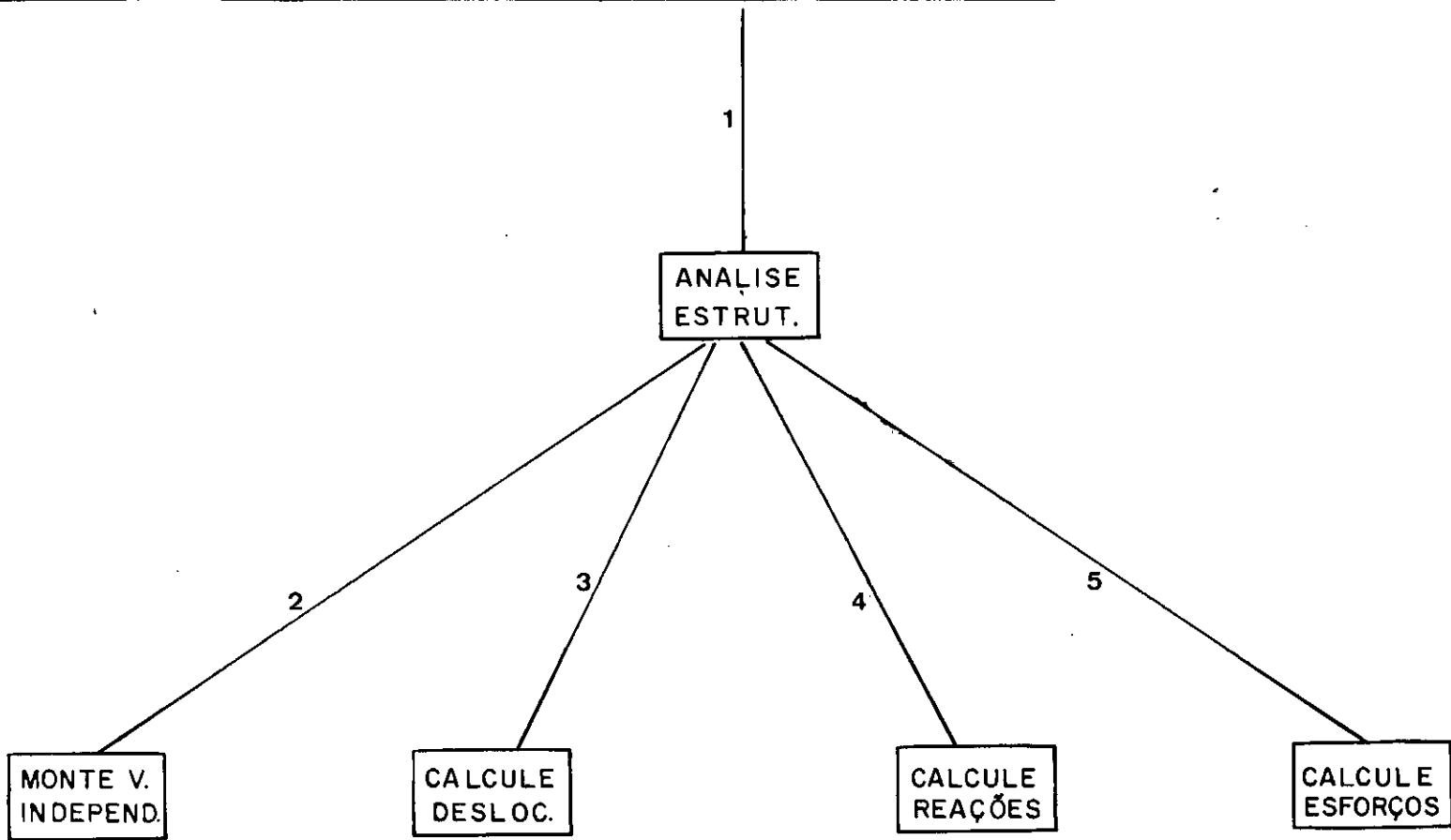


figura IV.5

Parâmetros da figura IV.5

- 1 - Drive+Nome+Modelo+Controle
- 2 - Drive+Nome+Modelo+Controle
- 3 - Drive+Nome+Modelo+Controle
- 4 - Drive+Nome+Modelo+Controle
- 5 - Drive+Nome+Modelo+Controle

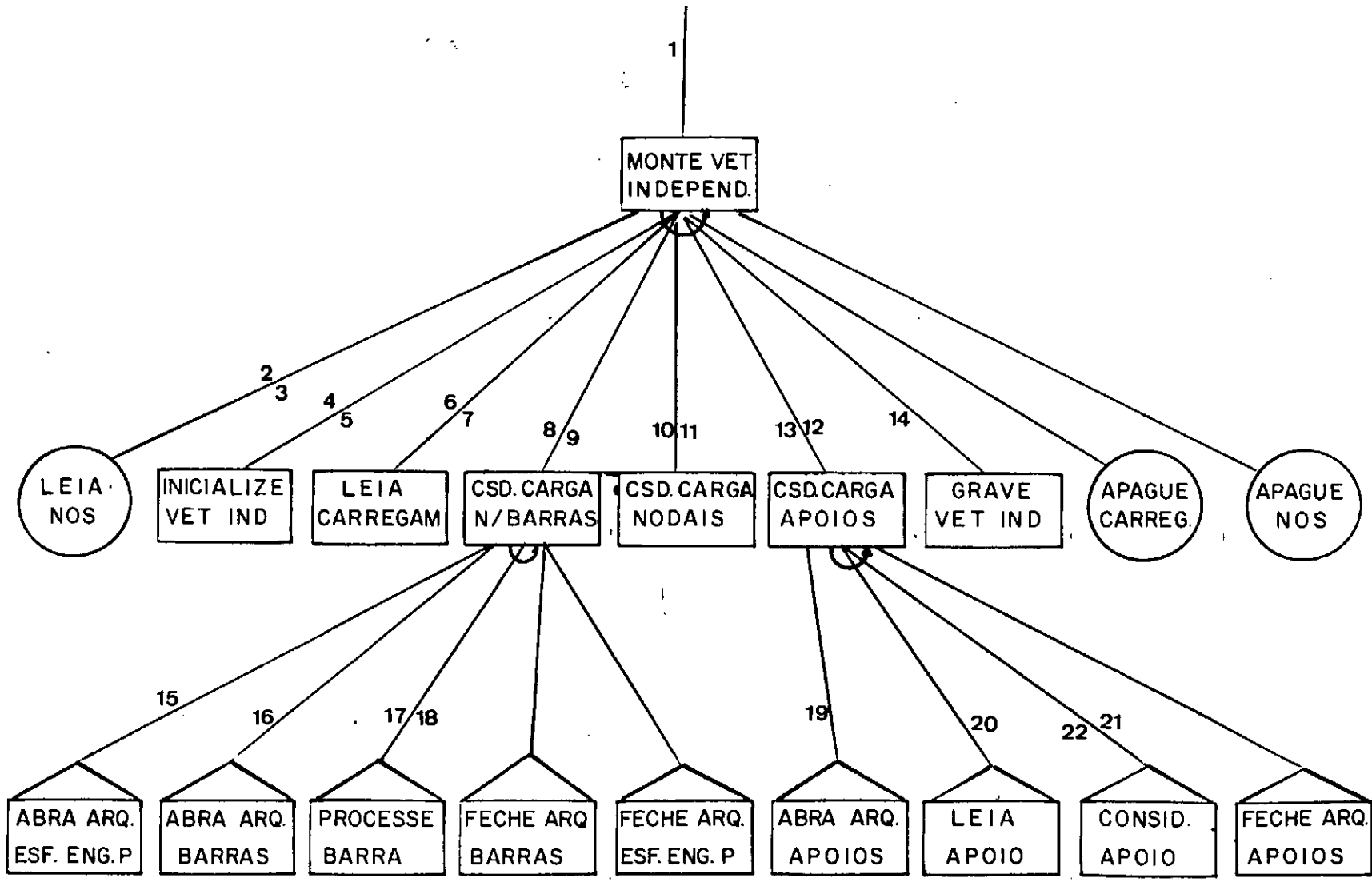


figura 1 v. 6

, Parâmetros da figura IV.6

- 1 - Drive+Nome+Modelo+Controle
- 2 - Drive+Nome
- 3 - Nós
- 4 - Vetor Independente
- 5 - Vetor Independente
- 6 - Drive+Nome
- 7 - Cargas
- 8 - Drive+Nome+Modelo+Controle
- 9 - Vetor Independente
- 10 - Controle+Cargas+Vetor Independente
- 11 - Vetor Independente
- 12 - Vetor Independente
- 13 - Drive+Nome+Controle+Vetor Independente
- 14 - Drive+Nome+Vetor Independente
- 15 - Drive+Nome
- 16 - Drive+Nome
- 17 - Controle+Modelo+Cargas+Vetor Independente
- 18 - Vetor Independente
- 19 - Drive+Nome
- 20 - Apoio
- 21 - Vetor Independente
- 22 - Controle+Apoio+Vetor Independente

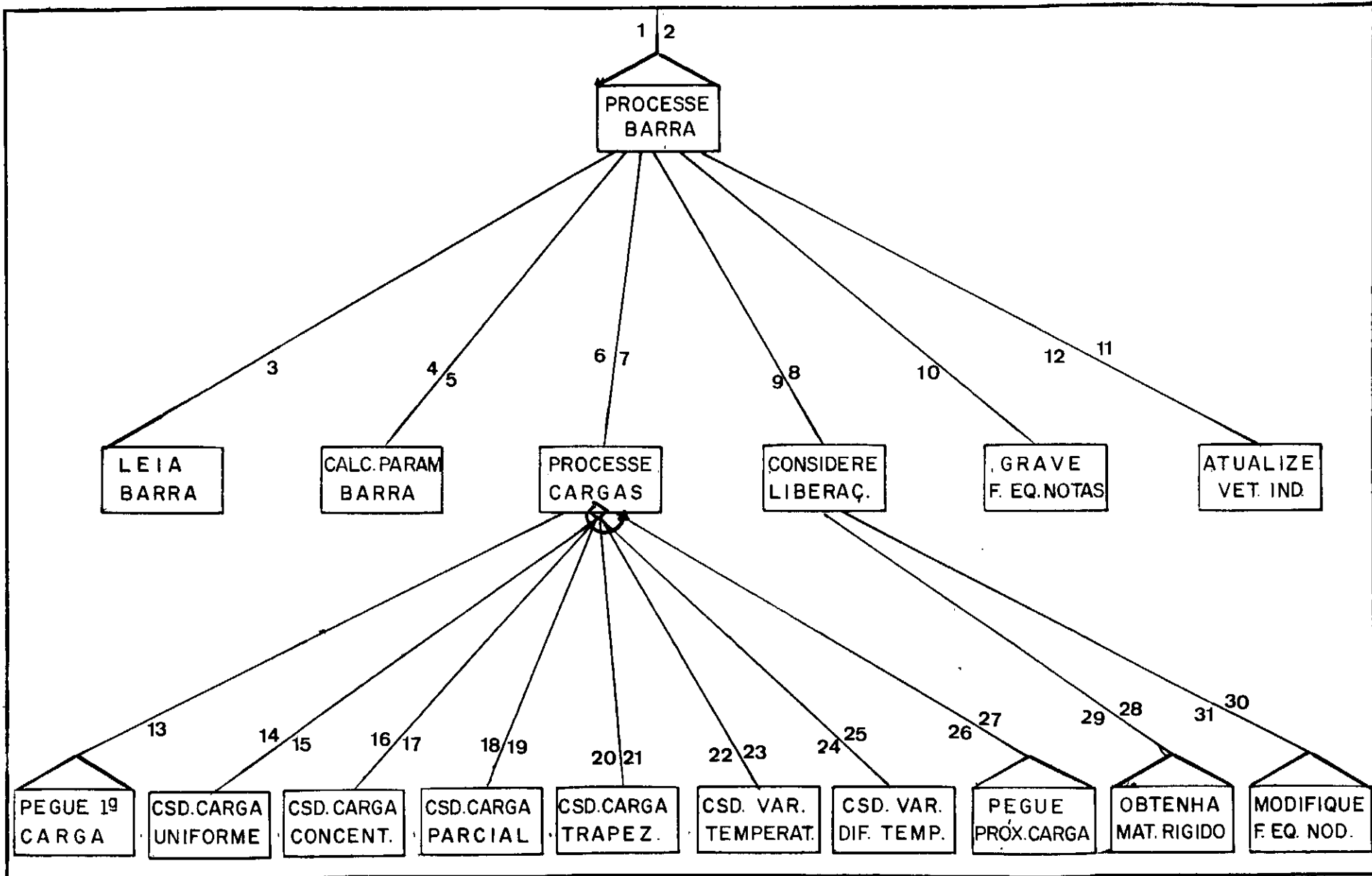


figura IV.7

Parâmetros da figura IV.7

- 1 - Controle+Modelo+Cargas+Vetor Independente
- 2 - Vetor Independente
- 3 - Barra
- 4 - Barra
- 5 - Parâmetros
- 6 - Cargas+Parâmetros
- 7 - Esforços
- 8 - Barra+Esforços+Controle
- 9 - Esforços
- 10 - Esforços
- 11 - Esforços+Vetor Independente
- 12 - Vetor Independente
- 13 - Carga
- 14 - Parâmetros
- 15 - Carga
- 16 - Parâmetros
- 17 - Carga
- 18 - Parâmetros
- 19 - Carga
- 20 - Parâmetros
- 21 - Carga
- 22 - Parâmetros
- 23 - Carga
- 24 - Parâmetros
- 25 - Carga
- 26 - Parâmetros
- 27 - Carga
- 28 - Matriz Rigidez Barra
- 29 - Barra+Controle
- 30 - Esforços

31 - Esforços+Controle+Matriz Rigidez Barra

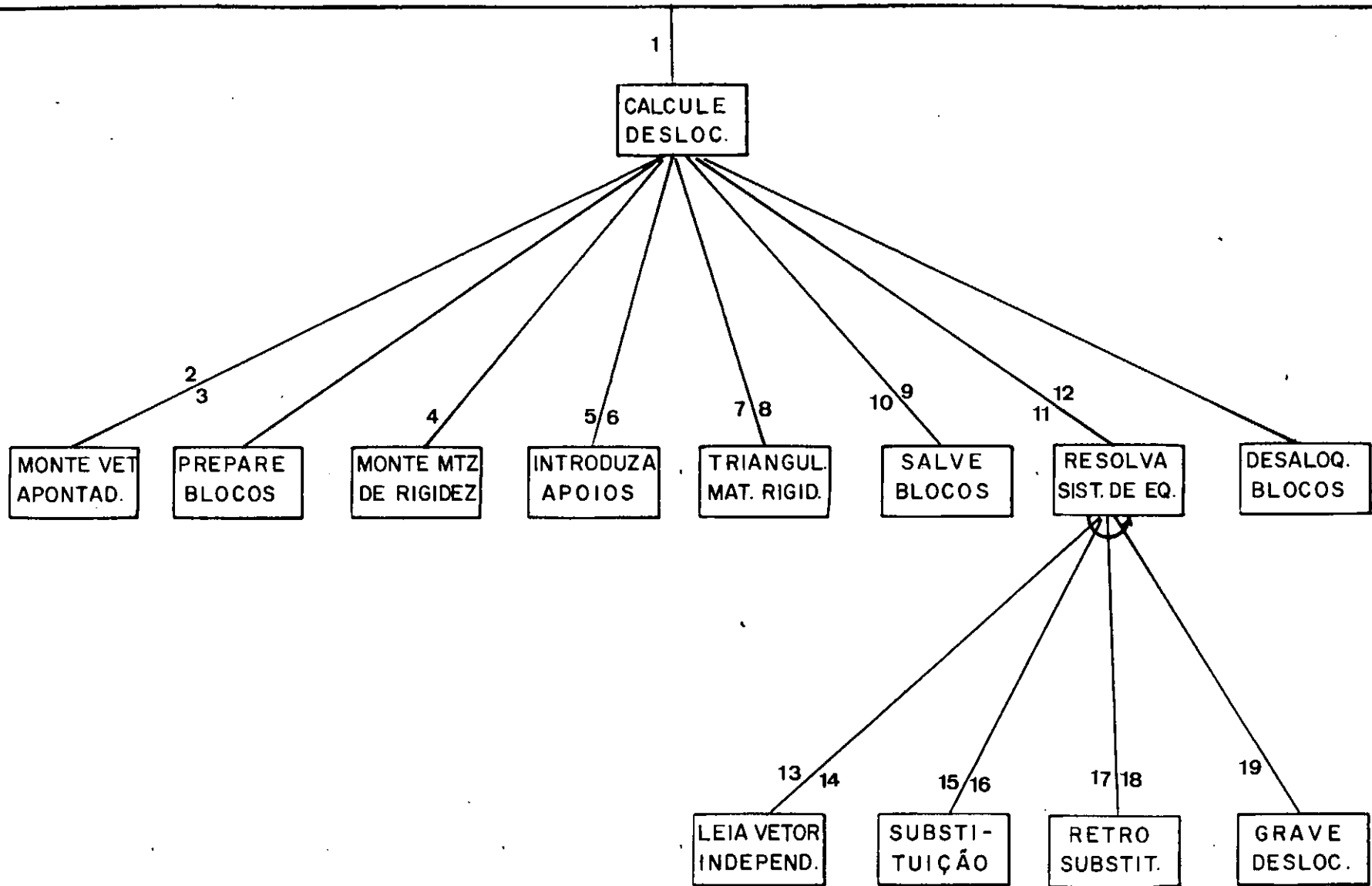


figura IV.8

Parâmetros da figura IV.8

- 1 - Drive+Nome+Modelo+Controle
- 2 - Drive+Nome+Controle
- 3 - Vetor Apontador
- 4 - Drive+Nome+Controle+Vetor Apontador
- 5 - Drive+Nome+Modelo+Controle+Vetor Apontador
- 6 - Matriz de Rigidez
- 7 - Drive+Nome+Controle+Vetor Apontador+Matriz de Rigidez
- 8 - Matriz de Rigidez
- 9 - Matriz de Rigidez
- 10 - Matriz de Rigidez+Vetor Apontador+Controle
- 11 - Matriz de Rigidez
- 12 - Drive+Nome+Controle+Matriz de Rigidez
- 13 - Materials
- 14 - Drive+Nome
- 15 - Seções
- 16 - Drive+Nome
- 17 - Nós
- 18 - Drive+Nome
- 19 - Drive+Nome+Controle+Materials+Seções+Nós
- 20 - Bloco Matriz Rigidez
- 21 - Drive+Nome
- 22 - Vetor Independente
- 23 - Matriz de Rigidez+Vetor Apontador+Vetor Independente
- 24 - Vetor Independente
- 25 - Matriz de Rigidez+Vetor Apontador+Vetor Independente
- 26 - Deslocamentos
- 27 - Drive+Nome+Deslocamentos

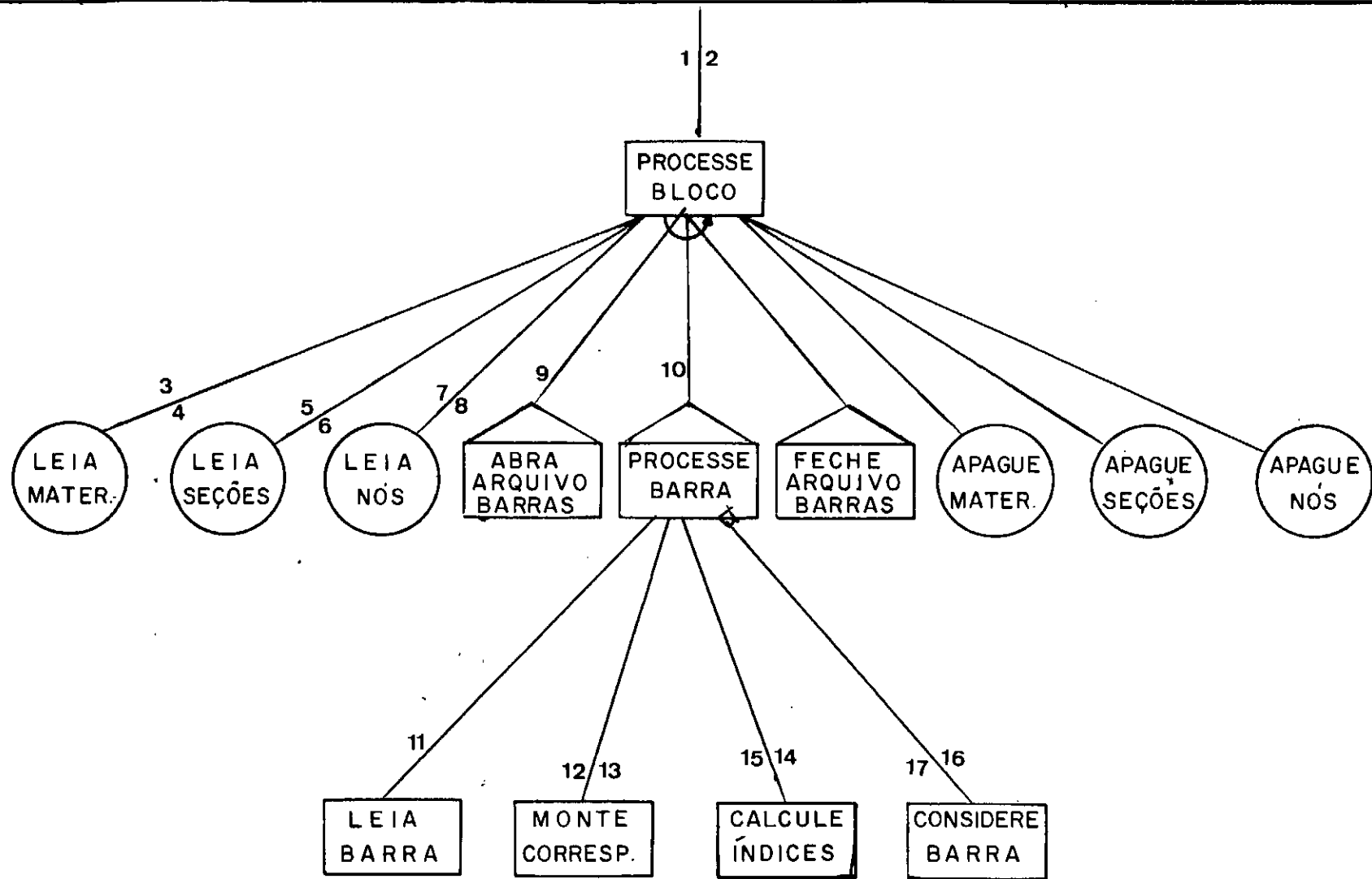


figura IV.9

Parâmetros da figura IV.9

- 1 - Drive+Nome+Controle+Modelo+Materiais+Seções+Nós
- 2 - Bloco Matriz de Rigidez
- 3 - Drive+Nome
- 4 - Modelo+Controle+Materiais+Seções+Nós
- 5 - Barra
- 6 - Controle+Barra
- 7 - Correspondência
- 8 - Índices
- 9 - Controle+Correspondência
- 10 - Bloco
- 11 - Modelo+Controle+Barra+Bloco+Materiais+Seções+Nós+Índices

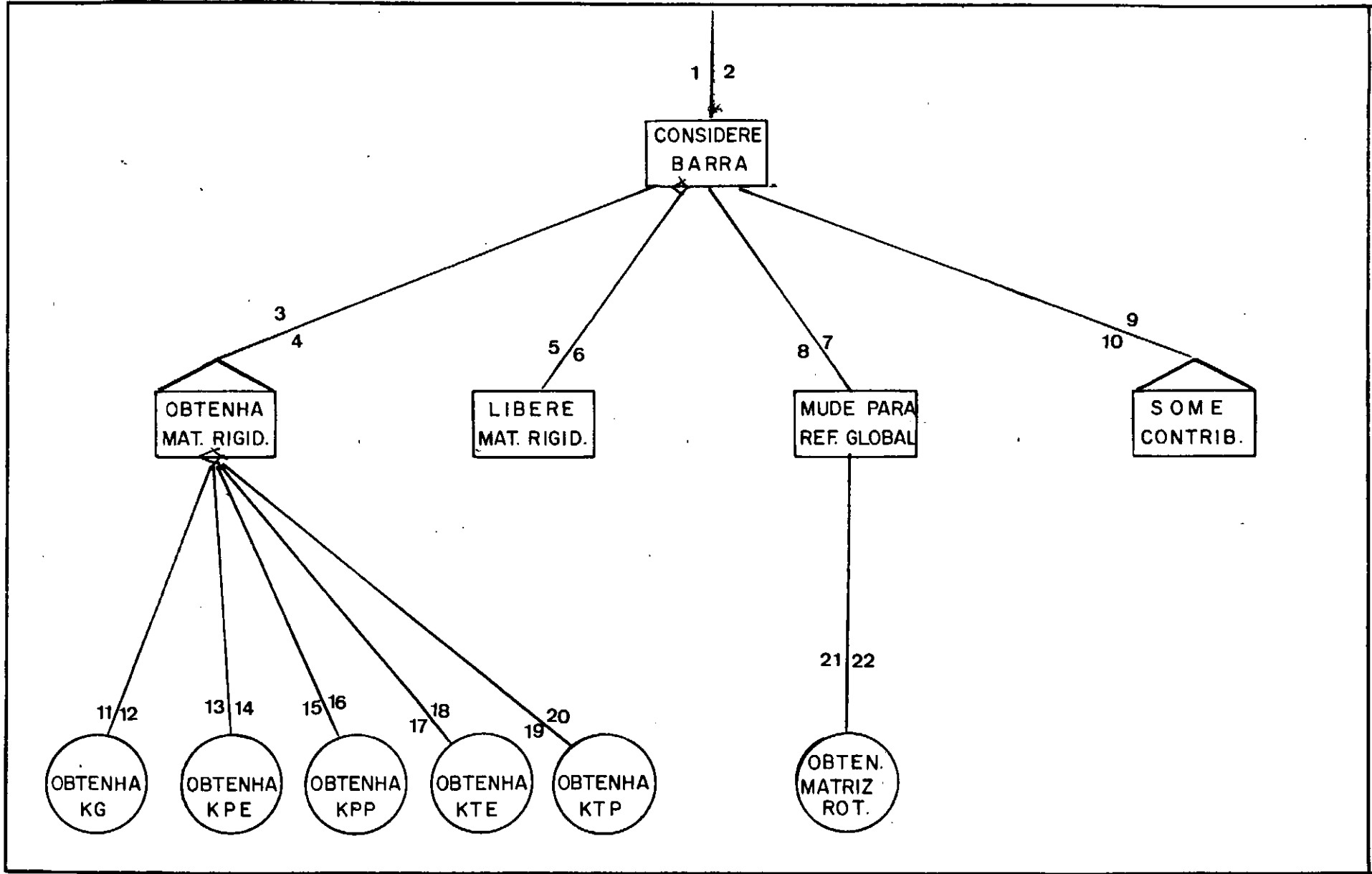


figura IV.10

Parâmetros da figura IV.10

- 1 - Modelo+Controle+Barra+Bloco+Materiais+Seções+Nós
- 2 - Bloco
- 3 - Modelo+Barra+Materiais+Seções+Nós
- 4 - Matriz Rigidez Barra
- 5 - Barra+Matriz Rigidez Barra
- 6 - Matriz Rigidez Barra
- 7 - Matriz Rigidez Barra
- 8 - Modelo+Controle+Barra+Matriz Rigidez Barra
- 9 - Bloco
- 10 - Bloco+Matriz Rigidez Barra+Indices+Correspondência
- 11 - Barra+Materiais+Seções+Nós
- 12 - Matriz Rigidez Barra
- 13 - Barra+Materiais+Seções+Nós
- 14 - Matriz Rigidez Barra
- 15 - Barra+Materiais+Seções+Nós
- 16 - Matriz Rigidez Barra
- 17 - Barra+Materiais+Seções+Nós
- 18 - Matriz Rigidez Barra
- 19 - Barra+Materiais+Seções+Nós
- 20 - Matriz Rigidez Barra
- 21 - Barra
- 22 - Matriz de Rotação Barra

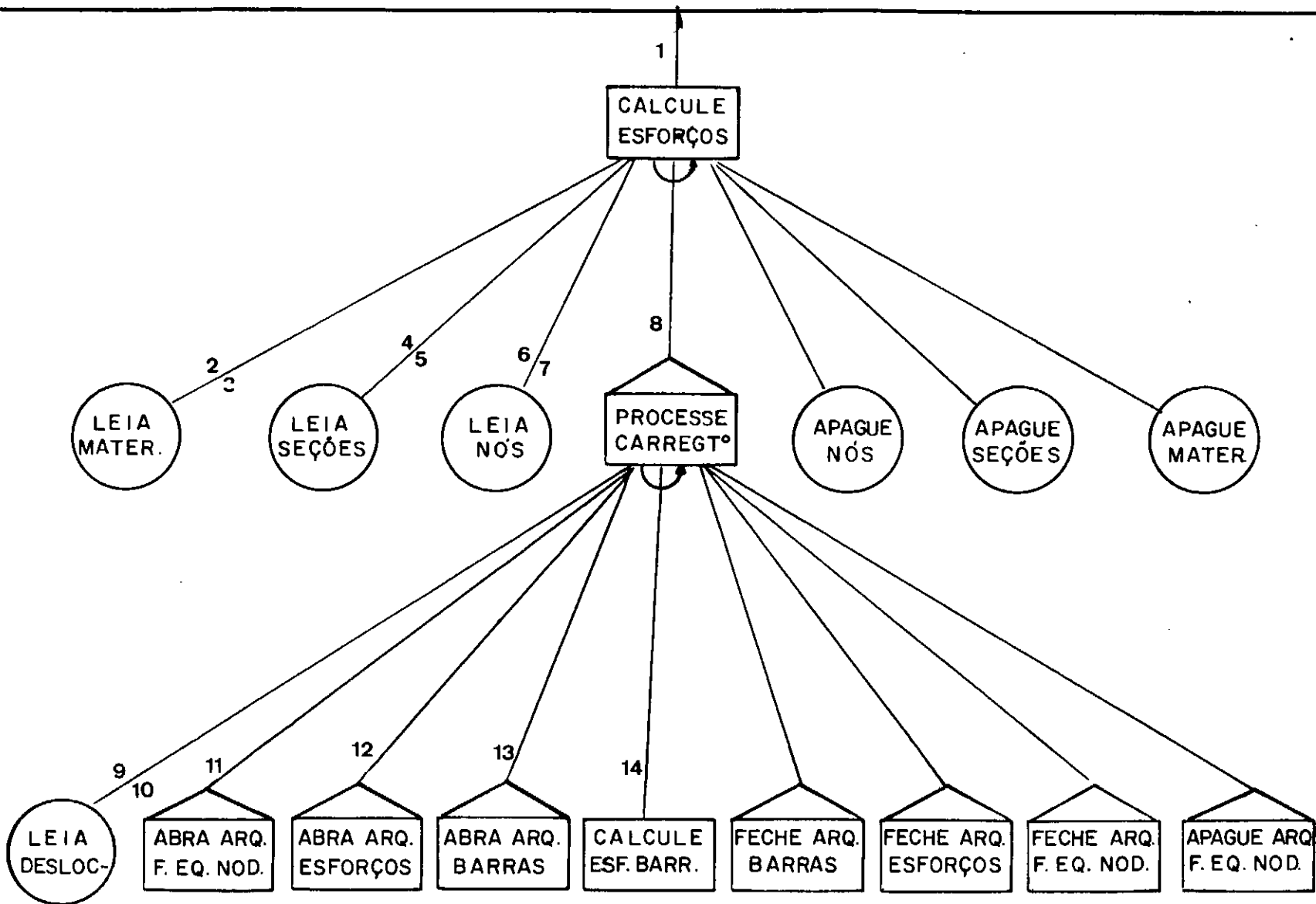


figura IV.11

Parâmetros da figura IV.11

- 1 - Drive+Nome+Modelo+Controle
- 2 - Drive+Nome
- 3 - Materiais
- 4 - Drive+Nome
- 5 - Seções
- 6 - Drive+Nome
- 7 - Nós
- 8 - Drive+Nome+Controle+Modelo
- 9 - Drive+Nome
- 10 - Deslocamentos
- 11 - Drive+Nome
- 12 - Drive+Nome
- 13 - Drive+Nome
- 14 - Modelo+Controle

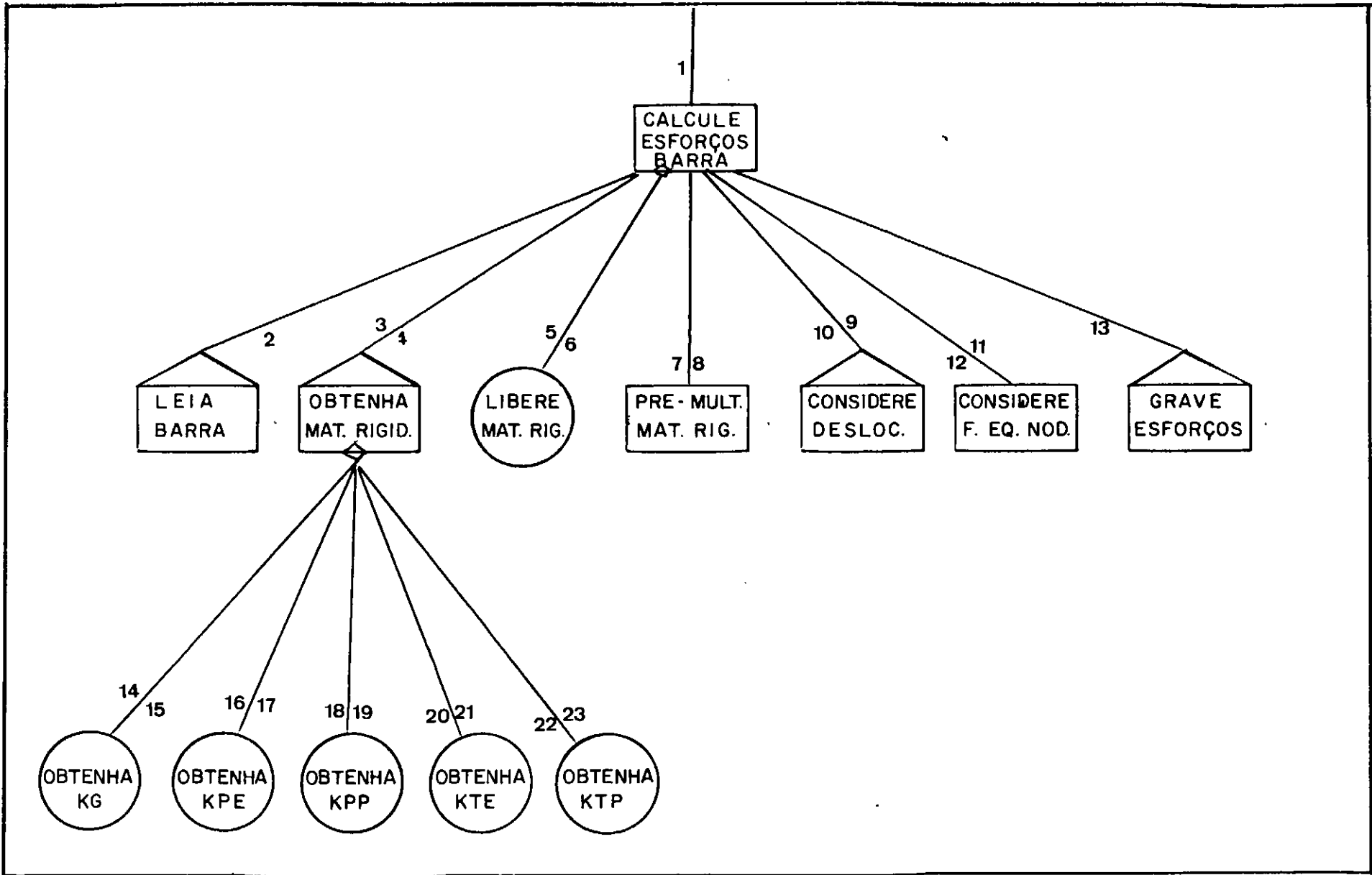


Figura IV.12

Parâmetros da figura IV.12

- 1 - Modelo+Controle
- 2 - Barra
- 3 - Modelo+Barra+Materials
- 4 - Matriz Rigidez Barra
- 5 - Barra+Matriz Rigidez Barra
- 6 - Matriz Rigidez Barra
- 7 - Modelo+Matriz Rigidez Barra
- 8 - Matriz Rigidez Barra
- 9 - Esforços
- 10 - Matriz Rigidez Barra+Deslocamentos
- 11 - Esforços
- 12 - Esforços+Modelo
- 13 - Esforços
- 14 - Barra+Materials+Seções+Nós
- 15 - Matriz Rigidez Barra
- 16 - Barra+Materials+Seções+Nós
- 17 - Matriz Rigidez Barra
- 18 - Barra+Materials+Seções+Nós
- 19 - Matriz Rigidez Barra
- 20 - Barra+Materials+Seções+Nós
- 21 - Matriz Rigidez Barra
- 22 - Barra+Materials+Seções+Nós
- 23 - Matriz Rigidez Barra

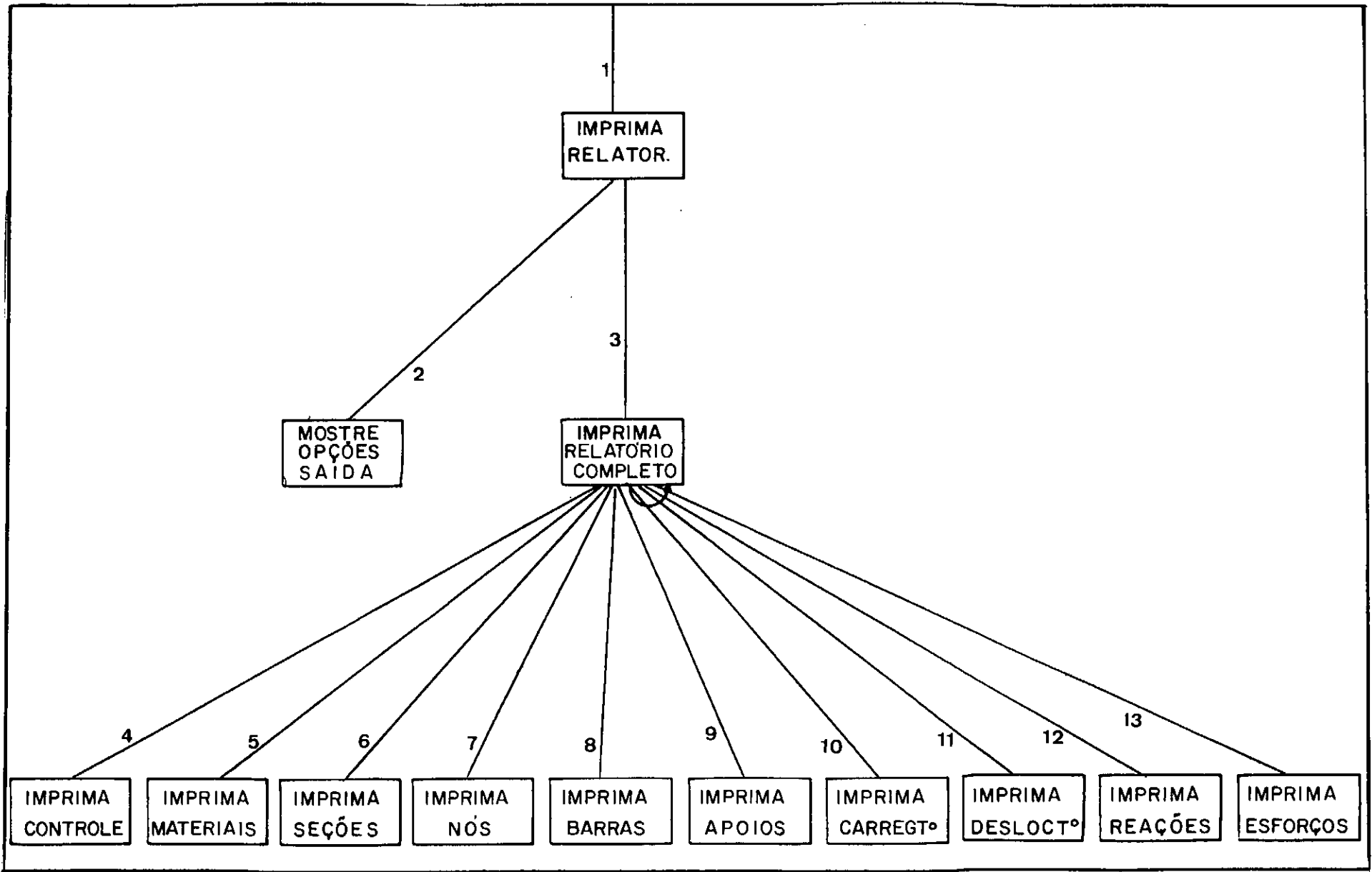
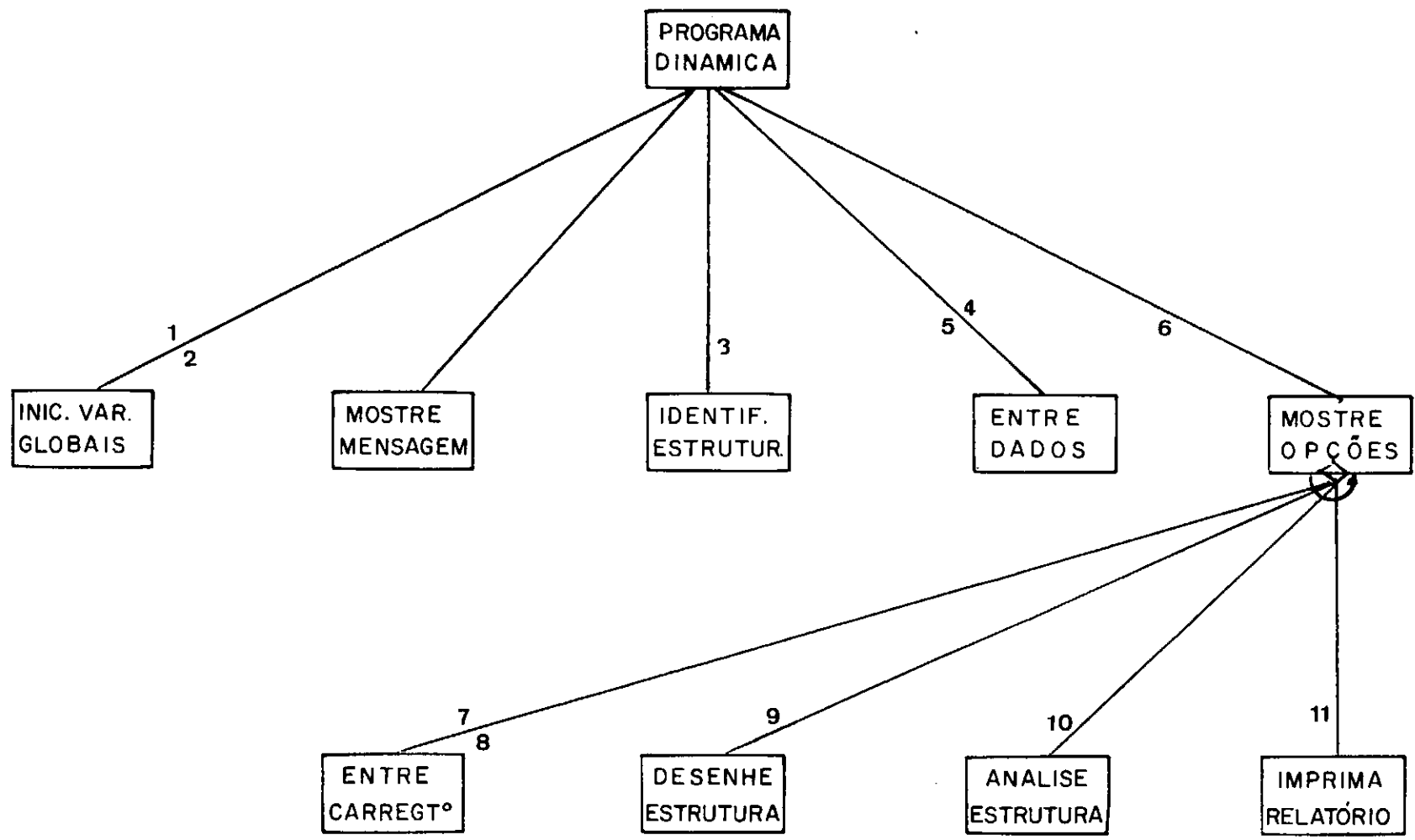


figura IV.13

Parâmetros da figura IV.13

- 1 - Drive+Nome+Modelo
- 2 - Opção de Saida
- 3 - Drive+Nome+Modelo
- 4 - Drive+Nome+Modelo
- 5 - Drive+Nome+Modelo
- 6 - Drive+Nome+Modelo
- 7 - Drive+Nome+Modelo
- 8 - Drive+Nome+Modelo
- 9 - Drive+Nome+Modelo
- 10 - Drive+Nome+Modelo
- 11 - Drive+Nome+Modelo
- 12 - Drive+Nome+Modelo
- 13 - Drive+Nome+Modelo

IV.5 - Projeto do Sistema Para Análise Dinâmica
figura IV.14



Parâmetros da figura IV.14

- 1 - Variáveis globais
- 2 - Variáveis globais inicializadas
- 3 - Drive+Nome
- 4 - Modelo + Controle
- 5 - Drive+Nome
- 6 - Drive+Nome+Modelo+Controle
- 7 - Drive+Nome+Modelo+Controle
- 8 - Controle
- 9 - Drive+Nome+Modelo+Controle
- 10 - Drive+Nome+Modelo+Controle
- 11 - Drive+Nome+Modelo

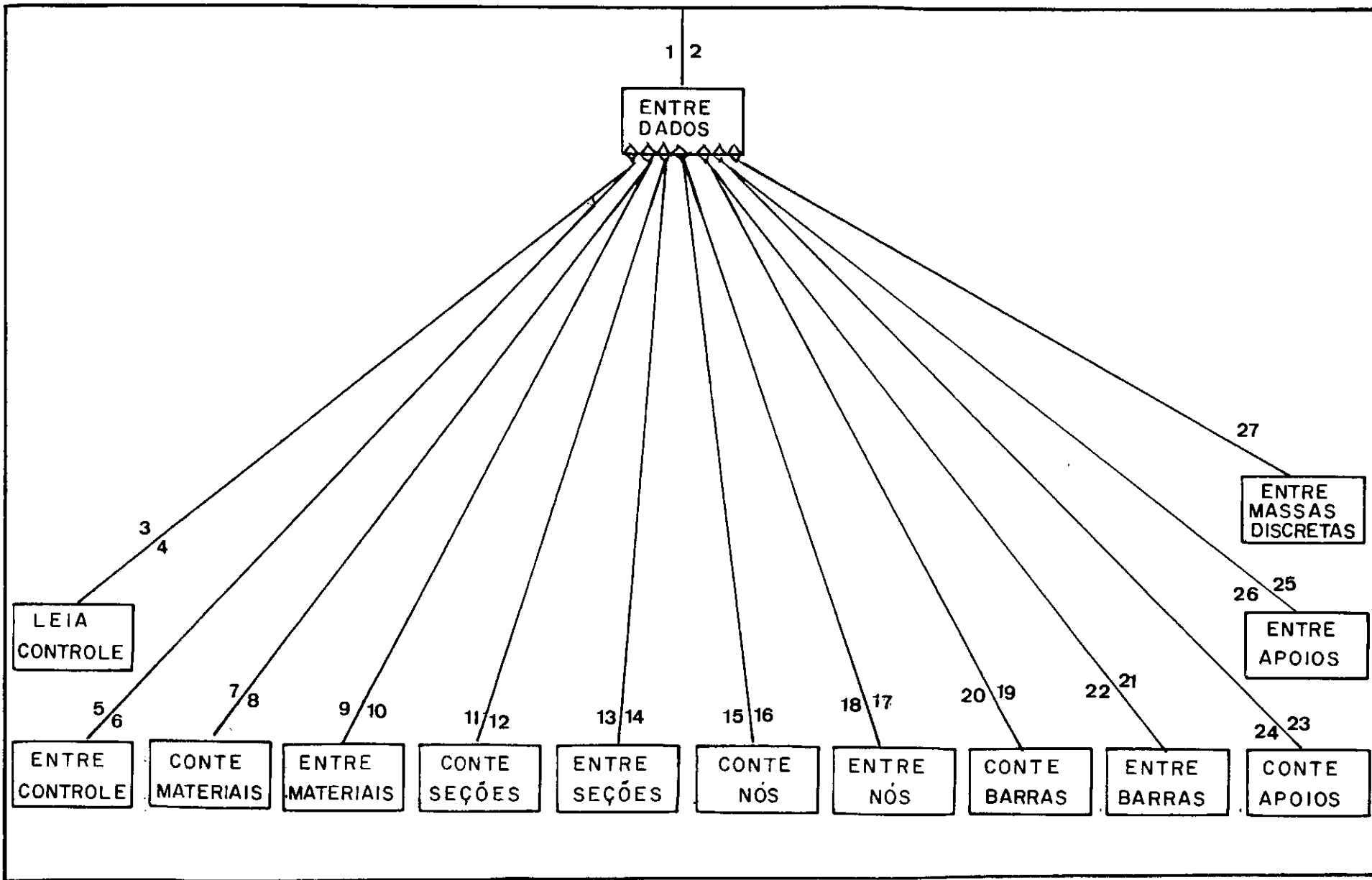


Figura IV.15

Parâmetros da figura IV.15

- 1 - Drive+Nome+Modelo
- 2 - Modelo
- 3 - Drive+Nome
- 4 - Modelo
- 5 - Drive+Nome
- 6 - Modelo
- 7 - Drive+Nome
- 8 - Número de materiais
- 9 - Drive+Nome
- 10 - Número de materiais
- 11 - Drive+Nome+Modelo
- 12 - Número de seções
- 13 - Drive+Nome+Modelo
- 14 - Número de seções
- 15 - Drive+Nome+Modelo
- 16 - Número de nós
- 17 - Drive+Nome+Modelo
- 18 - Número de nós
- 19 - Drive+Nome
- 20 - Número de barras
- 21 - Drive+Nome
- 22 - Número de barras
- 23 - Drive+Nome+Modelo
- 24 - Número de Apoios
- 25 - Drive+Nome+Modelo
- 26 - Número de apoios
- 27 - Drive+Nome+Modelo

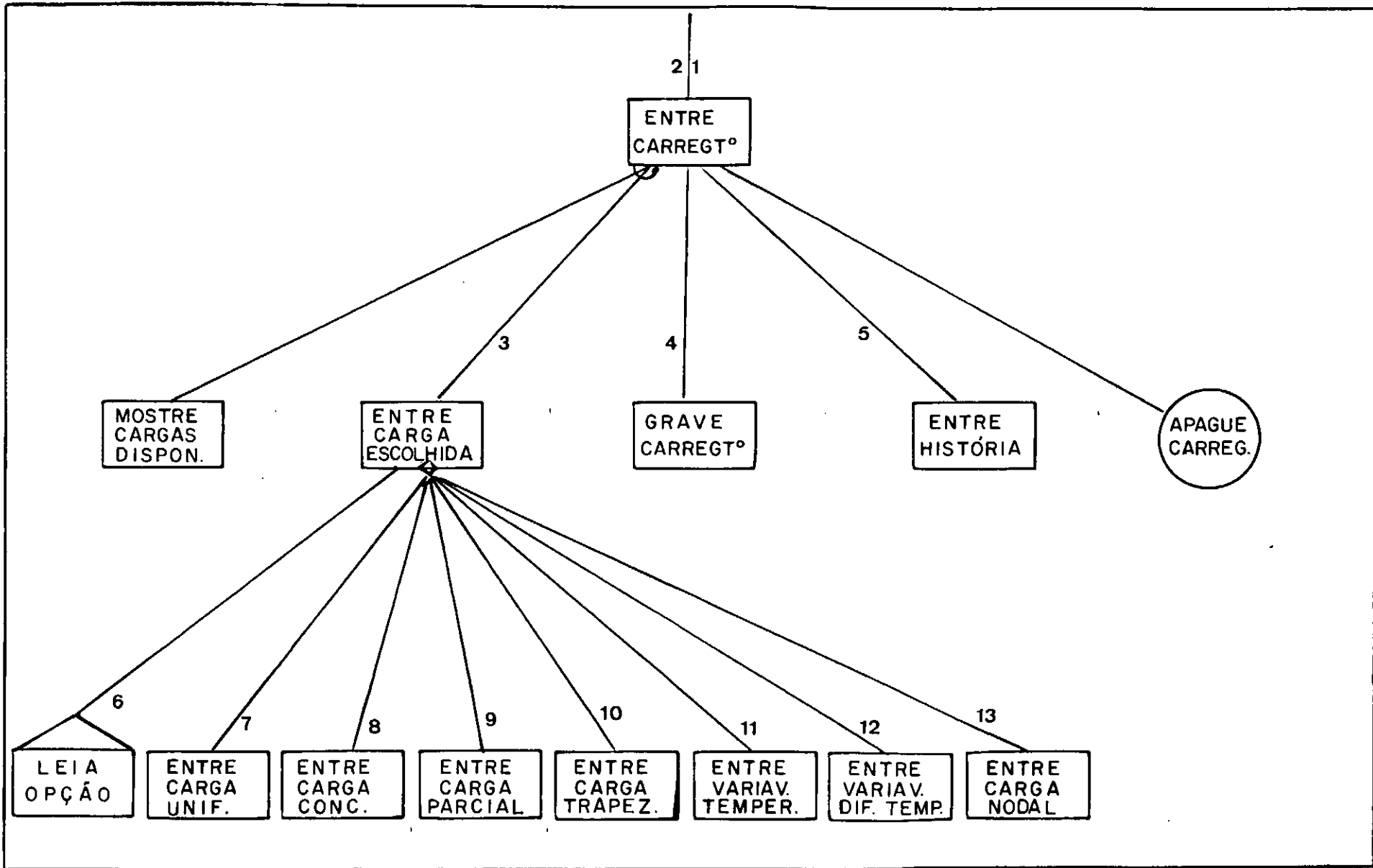


figura IV.18

Parâmetros da figura IV.16

- 1 - Controle
- 2 - Drive+Nome+Modelo+Controle
- 3 - Cargas
- 4 - Drive+Nome+cargas
- 5 - Drive+Nome
- 6 - opção
- 7 - carga
- 8 - carga
- 9 - carga
- 10 - carga
- 11 - carga
- 12 - carga
- 13 - carga

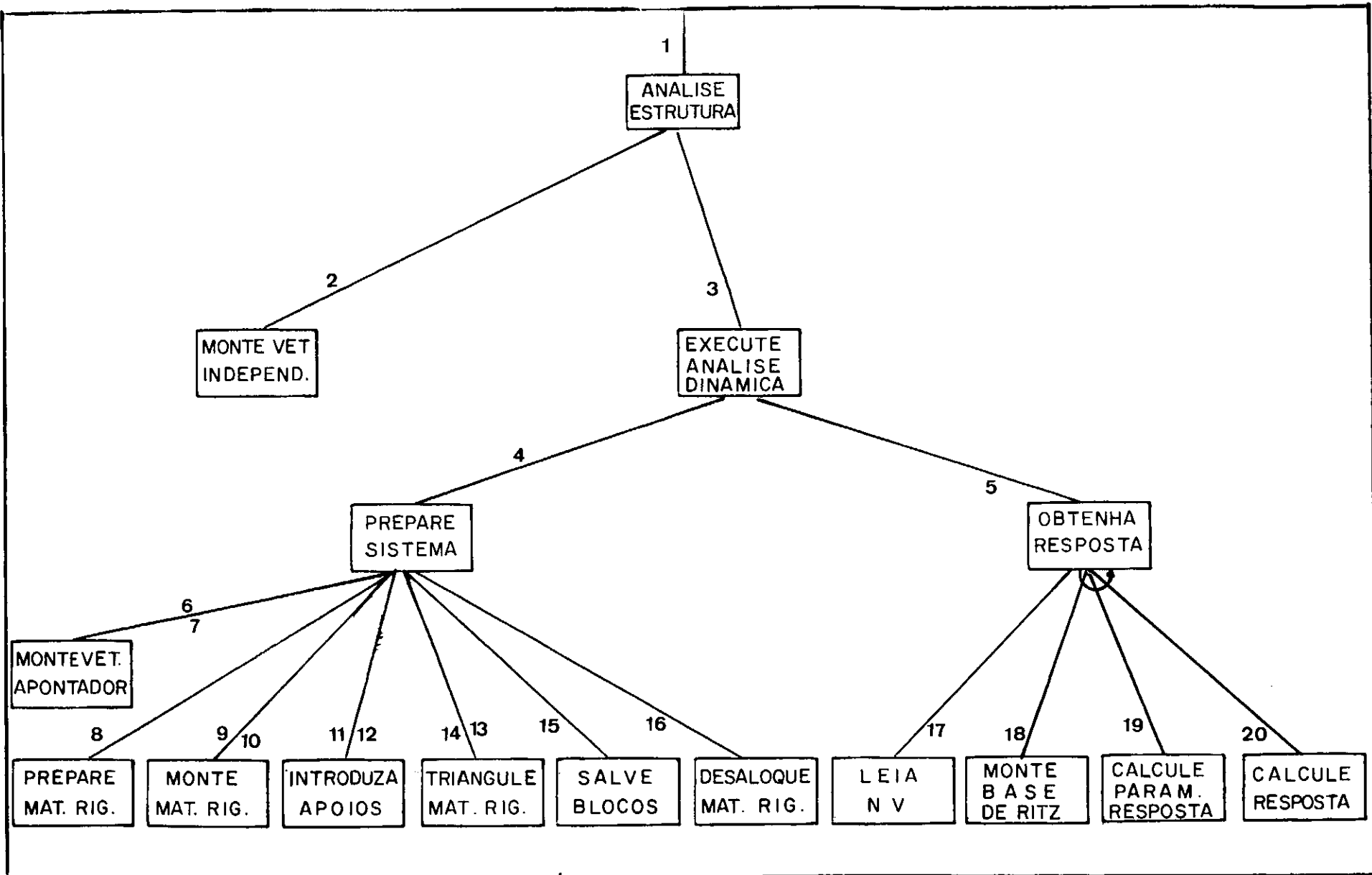


Figura IV.17

Parâmetros da figura IV.17

- 1 - Drive+Nome+Controle+Modelo
- 2 - Drive+Nome+Controle+Modelo
- 3 - Drive+Nome+Controle+Modelo
- 4 - Drive+Nome+Controle+Modelo
- 5 - Drive+Nome+Controle+Modelo
- 6 - Drive+Nome+Controle
- 7 - Vetor Apontador
- 8 - Drive+Nome+Controle+Vetor Apontador
- 9 - Drive+Nome+Controle+Vetor Apontador
- 10 - Matriz de Rigidez
- 11 - Drive+Nome+Controle+Vetor Apontador+Matriz de Rigidez
- 12 - Matriz de Rigidez
- 13 - Matriz de Rigidez
- 14 - Matriz de Rigidez+Vetor Apontador+Controle
- 15 - Matriz de Rigidez
- 16 - Matriz de Rigidez
- 17 - Número de Vetores de Ritz
- 18 - Número de Vetores de Ritz+Drive+Nome+Controle+Modelo
- 19 - Número de Vetores de Ritz+Drive+Nome+Controle
- 20 - Número de Vetores de Ritz+Drive+Nome+Controle+Modelo

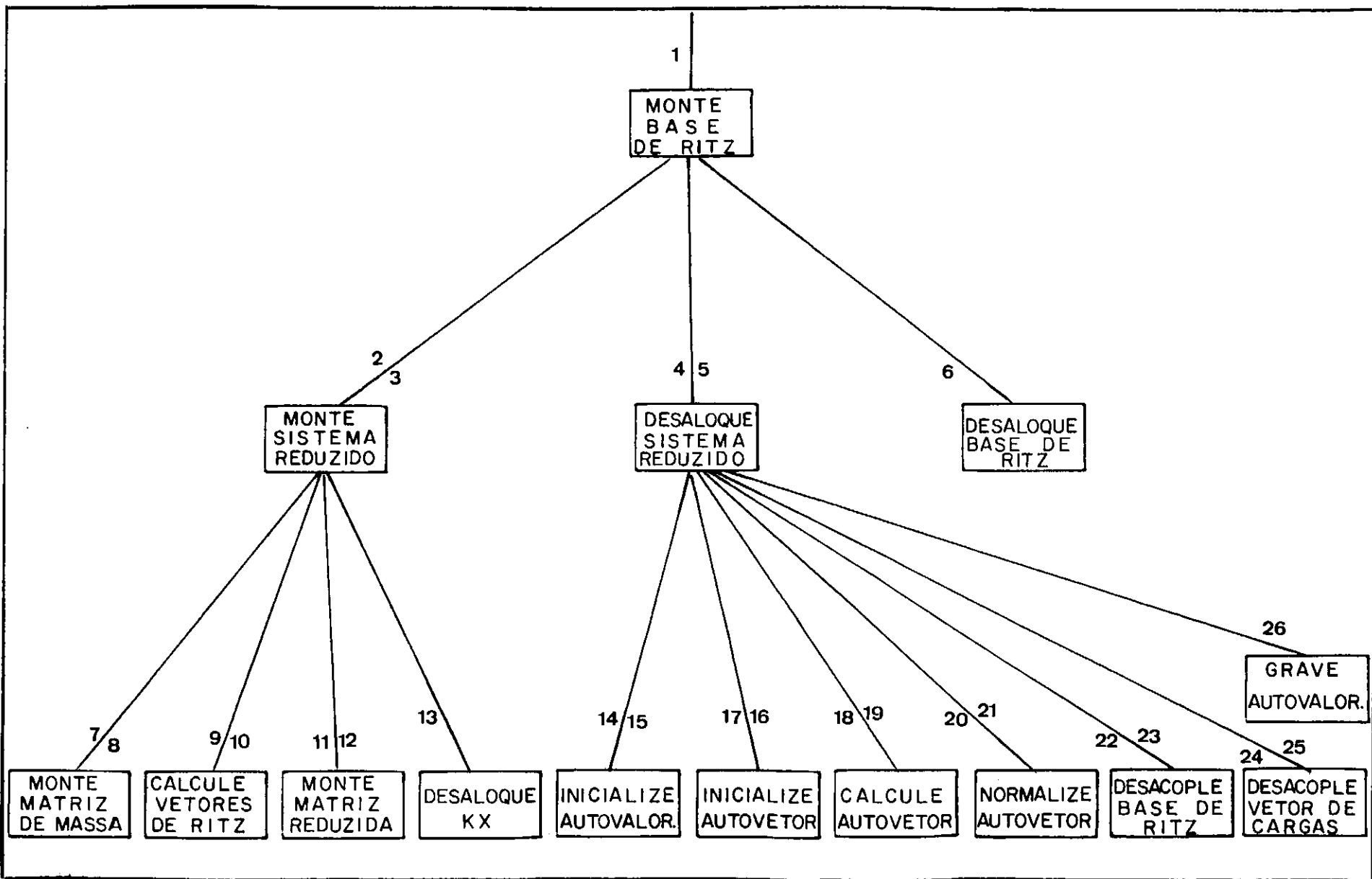


figura IV.18

Parâmetros da figura IV.18

- 1 - Drive+Nome+Controle+Modelo+Número Vet Ritz
- 2 - Drive+Nome+Controle+Número Vet Ritz
- 3 - Vetores de Ritz+Matriz Reduzida
- 4 - Drive+Nome+Número Vet Ritz+Vetores de Ritz+Matriz Reduzida
- 5 - Base de Ritz
- 6 - Número Vet Ritz+Base de Ritz
- 7 - Drive+Nome+Controle+Modelo
- 8 - Matriz de Massa
- 9 - Drive+Nome+Controle+Modelo+Número Vet Ritz+Matriz de Massa
- 10 - Vetores de Ritz+Kx
- 11 - Número Vet Ritz+Vetores de Ritz+Kx
- 12 - Matriz Reduzida
- 13 - Número Vet Ritz+Kx
- 14 - Número Vet Ritz
- 15 - AutoValores Iniciais
- 16 - Número Vet Ritz
- 17 - AutoVetores Iniciais
- 18 - Número Vet Ritz+Matriz Reduzida+AutoValoresIniciais+AutoVetoresIn
- 19 - AutoValores+AutoVetores
- 20 - Número Vet Ritz+AutoVetores
- 21 - AutoVetores
- 22 - Número Vet Ritz+Vetores de Ritz+AutoVetores
- 23 - Base De Ritz
- 24 - Controle+Número Vet Ritz+Vetor Independente
- 25 - Vetor Independente Reduzido
- 26 - Drive+Nome+Número Vet Ritz+AutoVetores

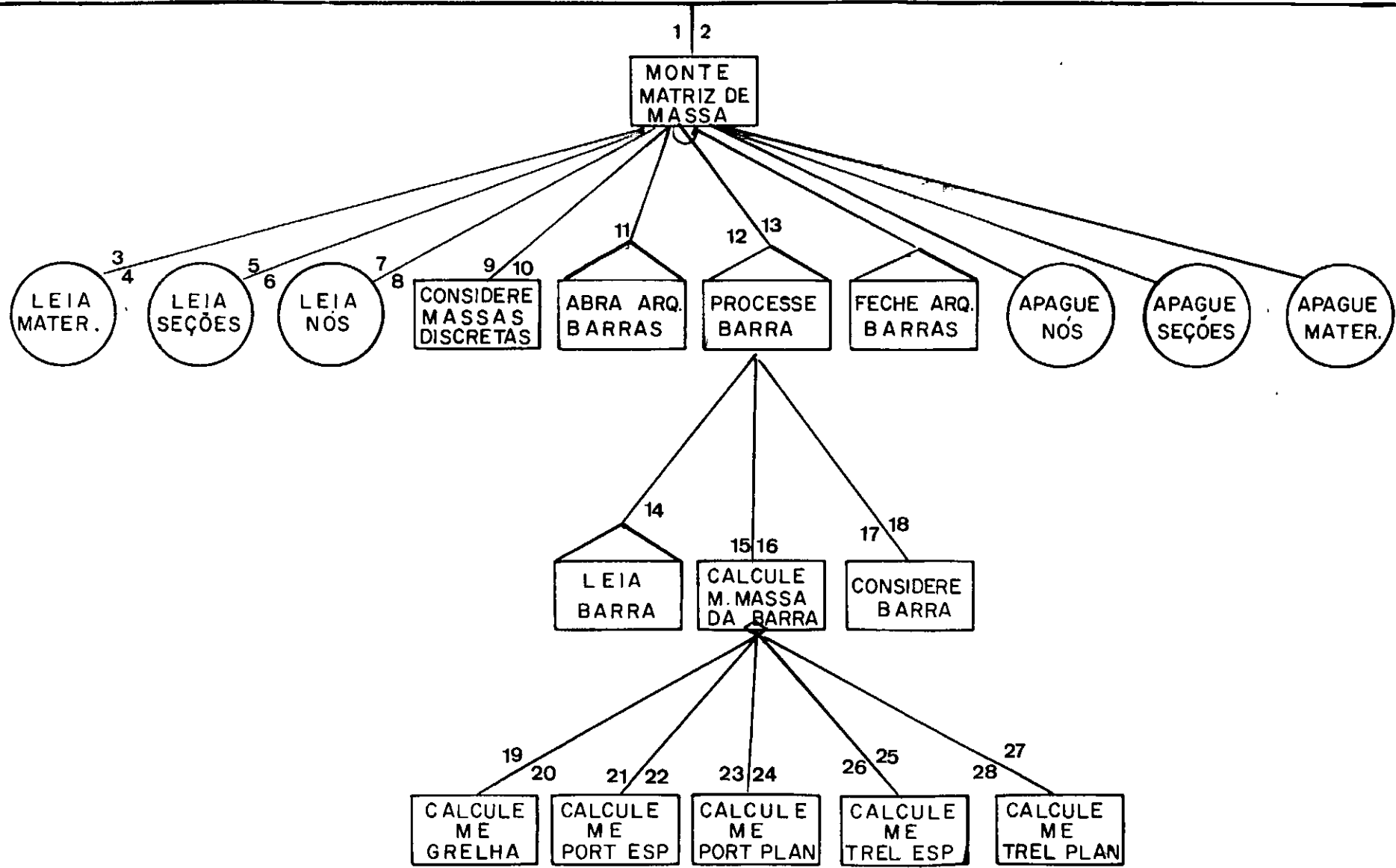


FIGURA IV.19

Parâmetros da figura IV.19

- 1 - Drive+Nome+Modelo+Controle
- 2 - Matriz de Massa
- 3 - Drive+Nome
- 4 - Materiais
- 5 - Drive+Nome
- 6 - Seções
- 7 - Drive+Nome
- 8 - Nós
- 9 - Drive+Nome
- 10 - Matriz de Massa
- 11 - Drive+Nome
- 12 - Modelo+Controle+Materiais+Seções+Nós+Matriz de Massa
- 13 - Matriz de Massa
- 14 - Barra
- 15 - Modelo+Materiais+Seções+Nós+Barra
- 16 - Matriz Massa Barra
- 17 - Matriz de Massa+Matriz Massa Barra
- 18 - Matriz de Massa
- 19 - Parâmetros
- 20 - Matriz Massa Barra
- 21 - Parâmetros
- 22 - Matriz Massa Barra
- 23 - Parâmetros
- 24 - Matriz Massa Barra
- 25 - Parâmetros
- 26 - Matriz Massa Barra
- 27 - Parâmetros

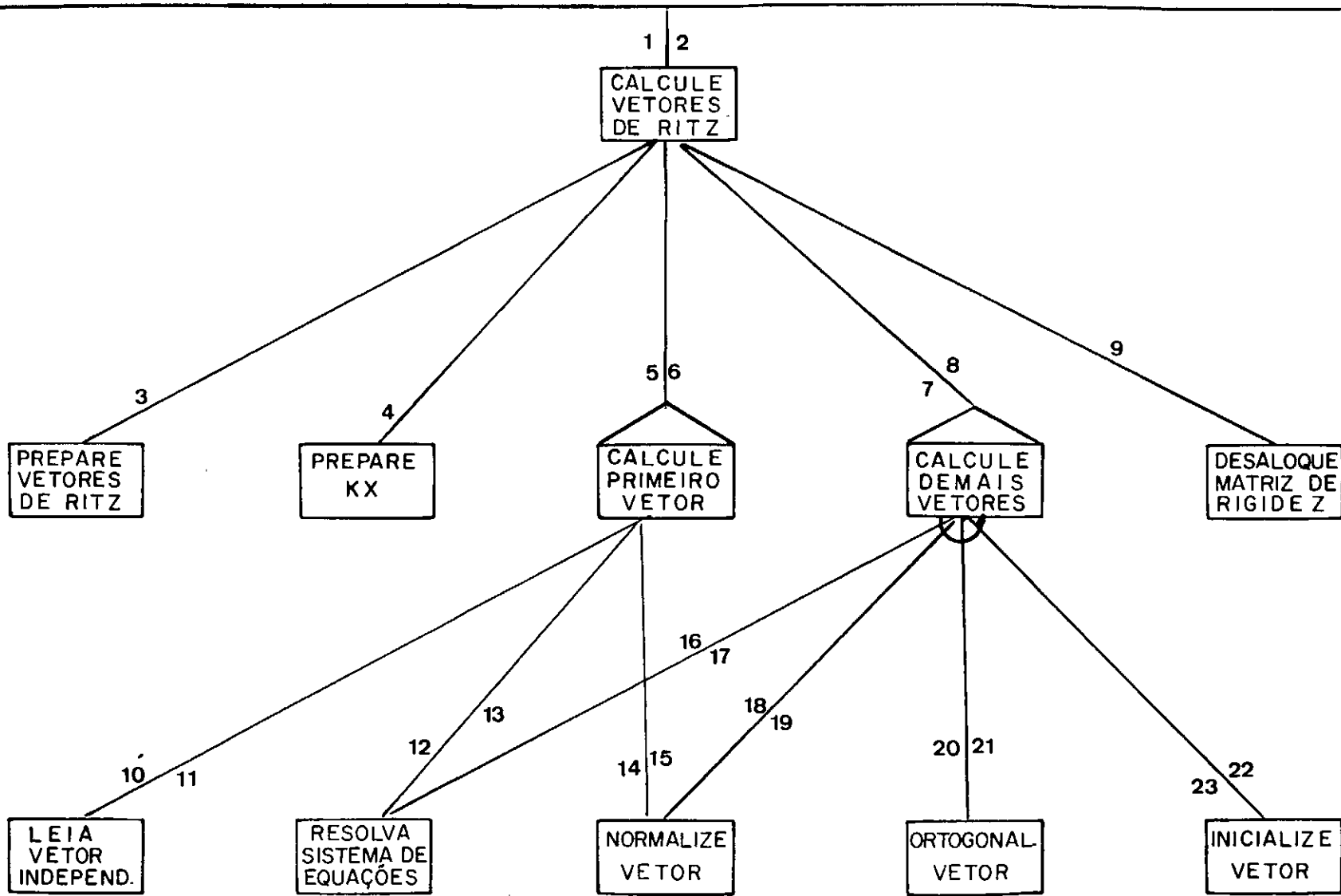


figura IV.20

Parâmetros da figura IV.20

- 1 - Drive+Nome+Controle+Modelo+Número Vet Ritz+Matriz da barra
- 2 - Vetores de Ritz+Kx
- 3 - Drive+Nome+Controle+Número Vet Ritz
- 4 - Drive+Nome+Controle+Número Vet Ritz
- 5 - Drive+Nome+Controle+Número Vet Ritz+Matriz de Massa
- 6 - Vetore de Ritz+Kx
- 7 - Número Vet Ritz+Kx+Vetor de Ritz+Matriz de Massa
- 8 - Vetores de Ritz+Kx
- 9 - Matriz de Rigidez
- 10 - Drive+Nome+Controle
- 11 - Vetor Independente
- 12 - Vetor Independente
- 13 - Vetor de Ritz
- 14 - Número Vet Ritz+Vetor de Ritz+Matriz de Massa+Kx
- 15 - Vetor de Ritz+Kx
- 16 - Vetor Solicitante
- 17 - Vetor de Ritz
- 18 - Número Vet Ritz+Vetor de Ritz
- 19 - Vetor de Ritz+Kx
- 20 - Número Vet Ritz+Vetor de Ritz+Kx
- 21 - Vetore de Ritz+Kx
- 22 - Vetor de Ritz
- 23 - Vetor de Ritz

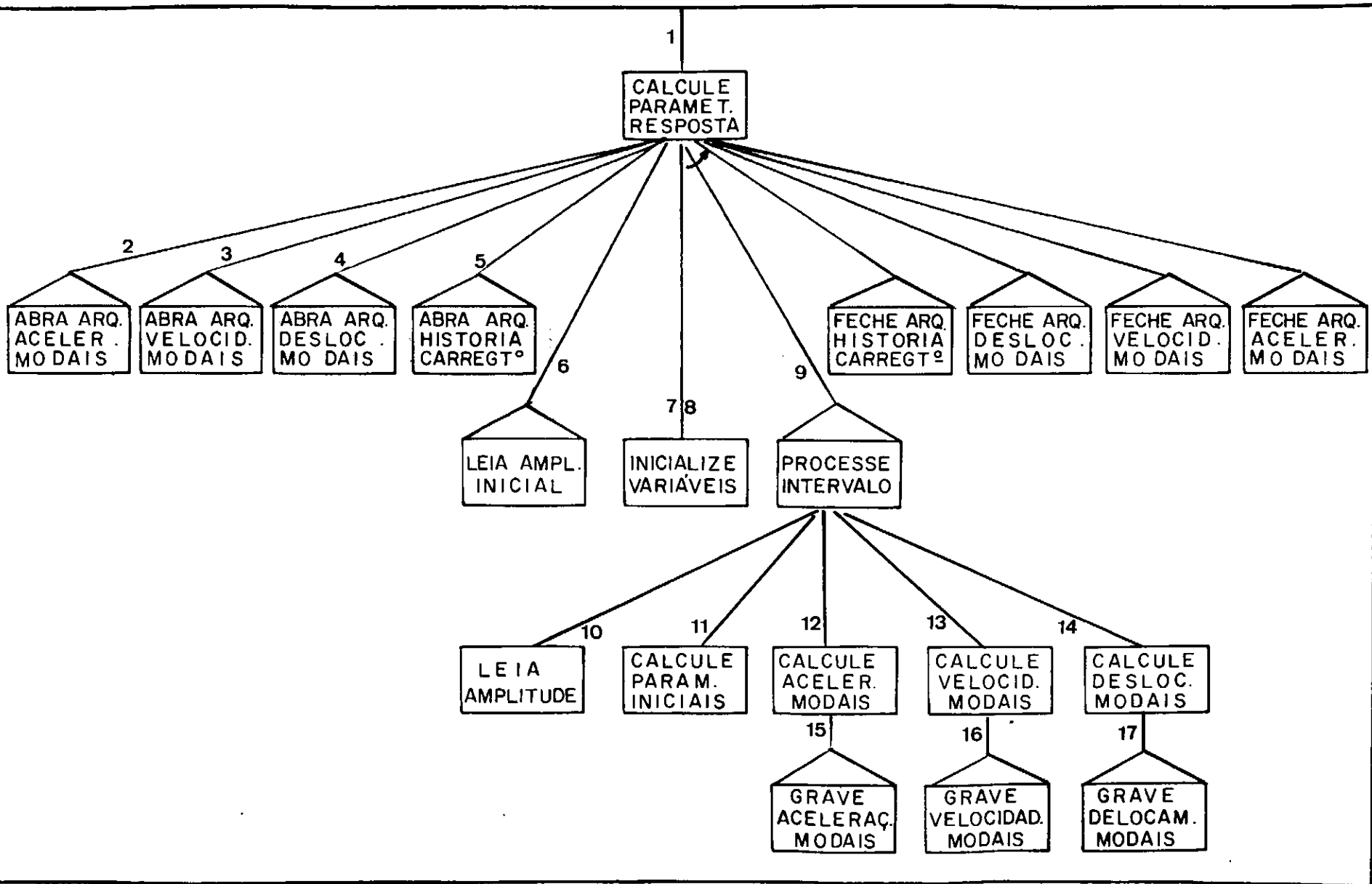


Figura IV.21

Parâmetros da figura IV.21

- 1 - Drive+Nome+Controle+Número Vet Ritz
- 2 - Drive+Nome
- 3 - Drive+Nome
- 4 - Drive+Nome
- 5 - Drive+Nome
- 6 - Amplitude Inicial
- 7 - Número Vet Ritz+Variáveis
- 8 - Variáveis Inicializadas
- 9 - Controle+Número Vet Ritz
- 10 - Amplitude
- 11 - Parâmetros
- 12 - Número Vet Ritz+Parâmetros
- 13 - Número Vet Ritz+Parâmetros
- 14 - Número Vet Ritz+Parâmetros
- 15 - Número Vet Ritz+Acelerações Reduzidas
- 16 - Número Vet Ritz+Velocidades Reduzidas
- 17 - Número Vet Ritz+Deslocamentos Reduzidos

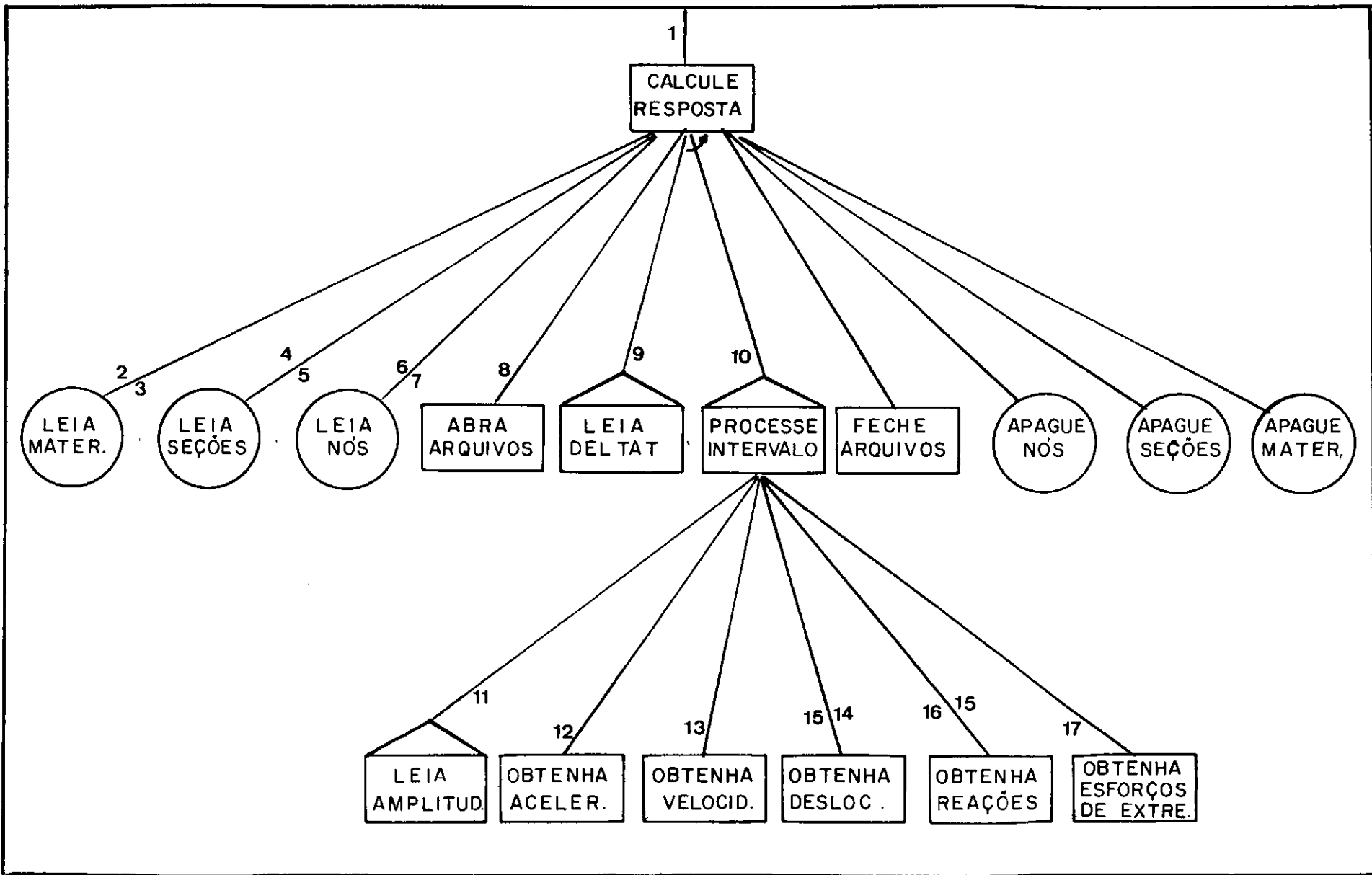


figura IV.22

Parâmetros da figura IV.22

- 1 - Drive+Nome+Controle+Modelo+Número Vet Ritz
- 2 - Drive+Nome
- 3 - Materiais
- 4 - Drive+Nome
- 5 - Seções
- 6 - Drive+Nome
- 7 - Nós
- 8 - Drive+Nome
- 9 - Delta T
- 10 - Controle+Modelo+Número Vet Ritz+Materiais+Seções+Nós
- 11 - Amplitude
- 12 - Controle+Número Vet Ritz
- 13 - Controle+Número Vet Ritz
- 14 - Deslocamentos
- 15 - Controle+Número Vet Ritz
- 16 - Drive+Nome+Controle+Deslocamentos
- 17 - Drive+Nome+Controle+Modelo+Deslocamentos

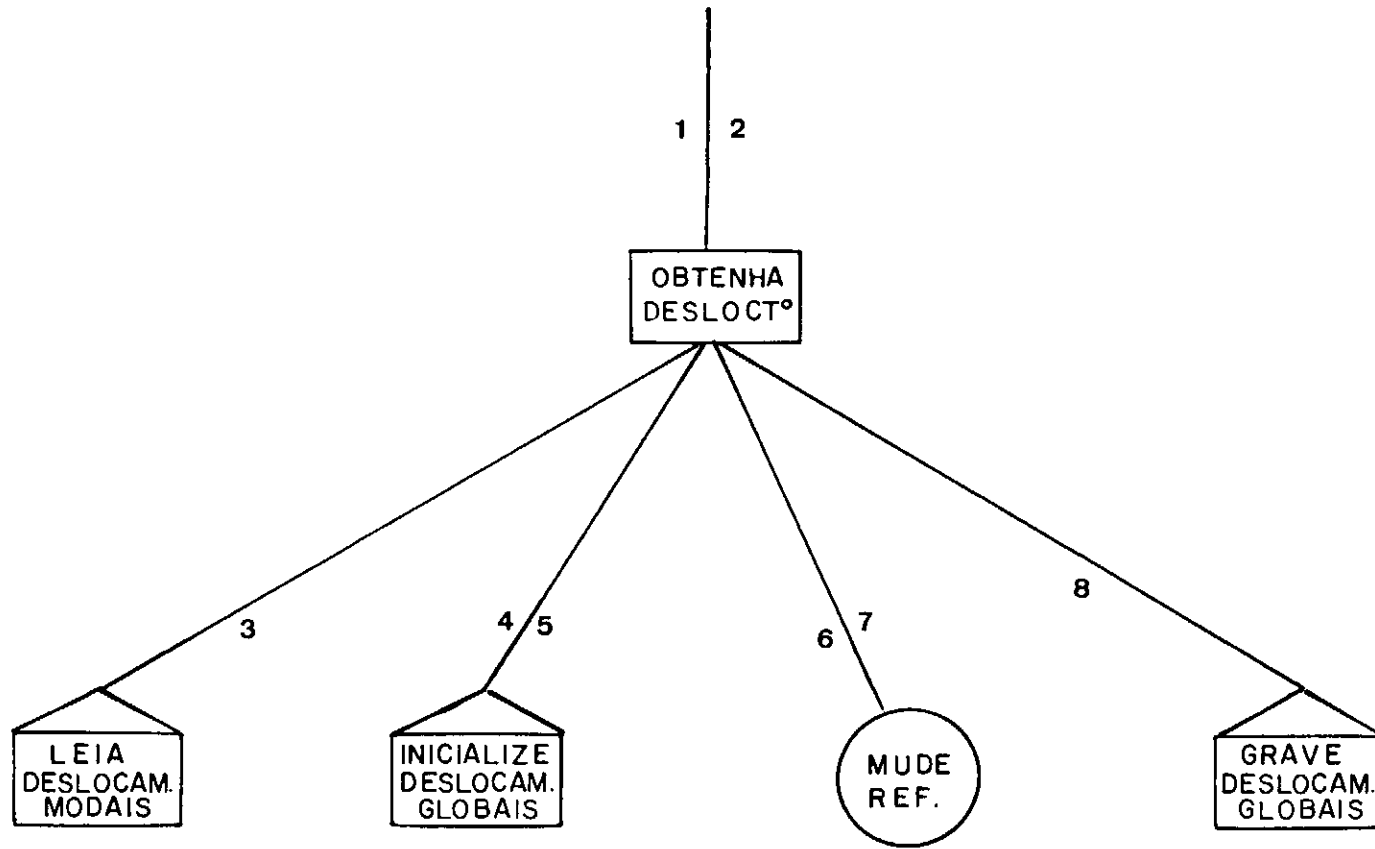


figura IV.23

Parâmetros da figura IV.23

- 1 - Controle+Número Vet Ritz
- 2 - Deslocamentos
- 3 - Deslocamentos
- 4 - Deslocamentos
- 5 - Deslocamentos
- 6 - Deslocamentos Reduzidos
- 7 - Deslocamentos
- 8 - Deslocamentos

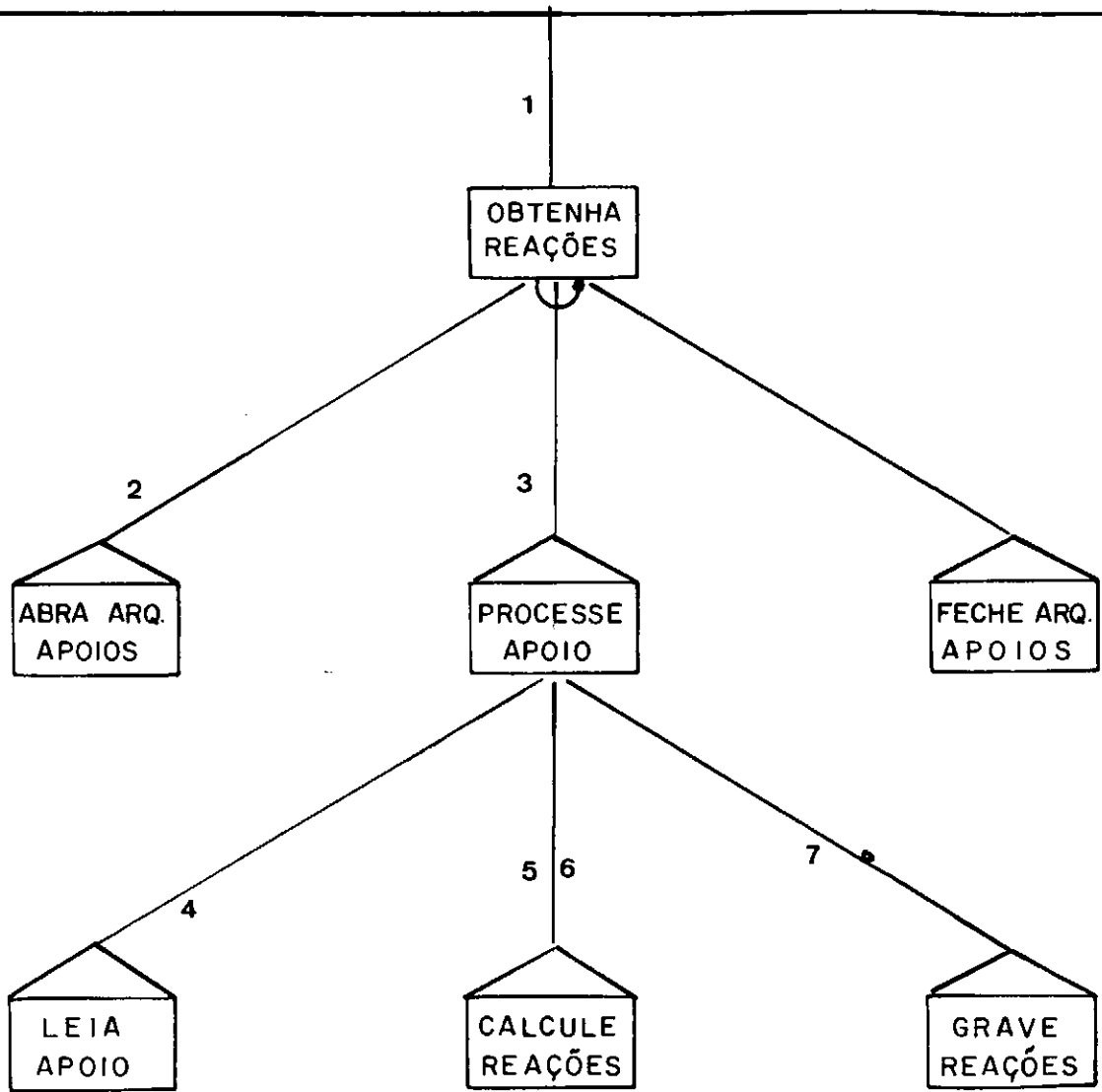


figura IV.24

Parâmetros da figura IV.24

- 1 - Drive+Nome+Controle+Deslocamentos
- 2 - Drive+Nome
- 3 - Controle+Deslocamentos
- 4 - Apoio
- 5 - Controle+Deslocamentos+Apoio
- 6 - Reação
- 7 - Reação

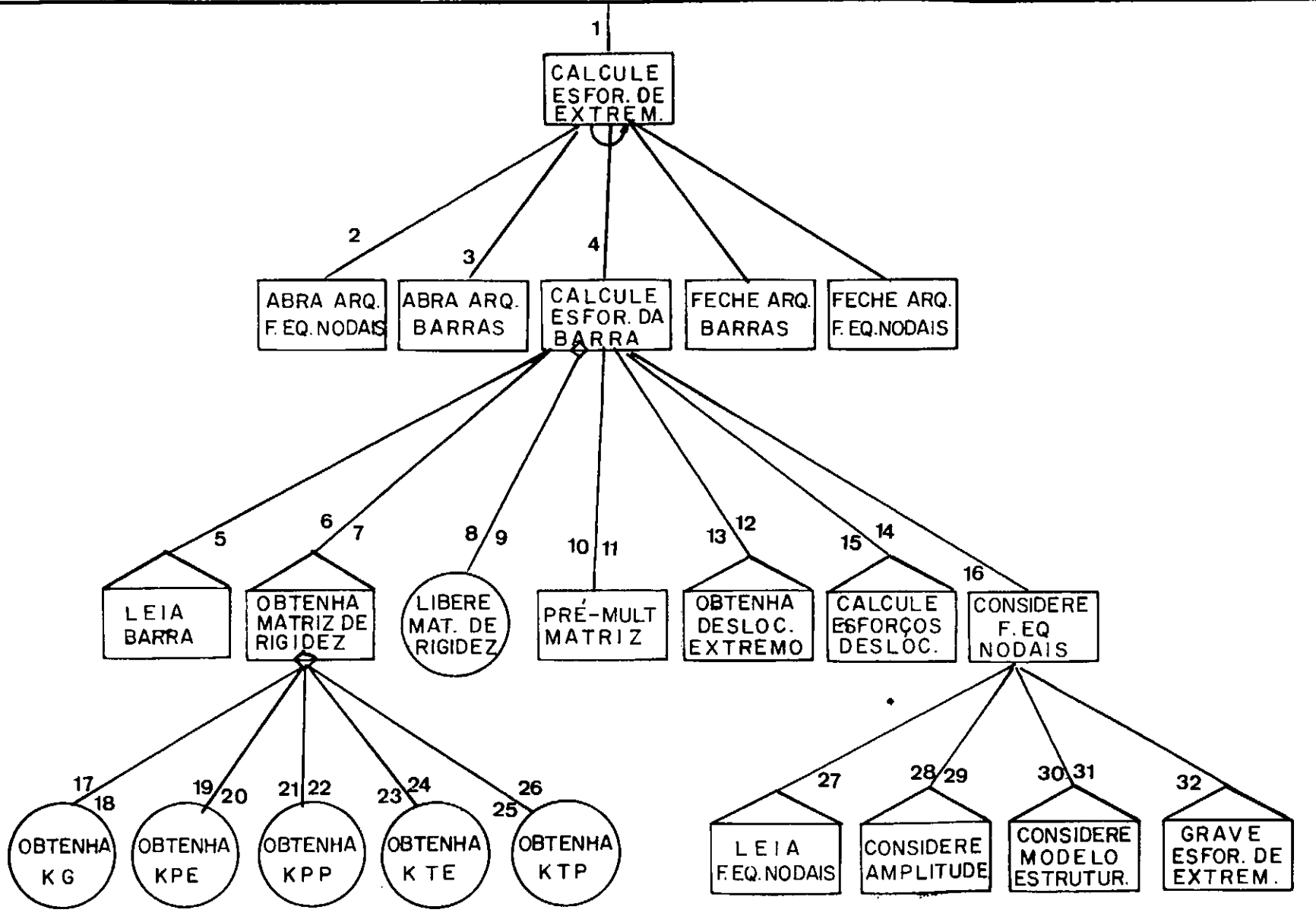


figura IV.25

Parâmetros da figura IV.25

- 1 - Drive+Nome+Modelo+Controle+Deslocamentos+Materiais+Seções+Nós
- 2 - Drive+Nome
- 3 - Drive+Nome
- 4 - Modelo+Controle+Deslocamentos+Materiais+Seções+Nós
- 5 - Barra
- 6 - Modelo+Controle+Materiais+Seções+Nós+Barra
- 7 - Matriz Rigidez Barra
- 8 - Barra+Matriz Rigidez Barra
- 9 - Matriz Rigidez Barra
- 10 - Modelo+Matriz Rigidez Barra+Barra+Nós
- 11 - Matriz Rigidez Barra
- 12 - Deslocamentos de Extremidade
- 13 - Deslocamentos+Barra
- 14 - Esforços de Extremidade
- 15 - Matriz Rigidez Barra+Deslocamentos de Extremidade
- 16 - Esforços de Extremidade+Modelo
- 17 - Barra+Materiais+Seções+Nós
- 18 - Matriz Rigidez Barra
- 19 - Barra
- 20 - Matriz Rigidez Barra
- 21 - Barra+Materiais+Seções+Nós
- 22 - Matriz Rigidez Barra
- 23 - Barra+Materiais+Seções+Nós
- 24 - Matriz Rigidez Barra
- 25 - Barra+Materiais+Seções+Nós
- 26 - Matriz Rigidez Barra
- 27 - Barra+Matriz Rigidez Barra+Nós
- 28 - Matriz Rigidez Barra
- 29 - Forças Equivalentes Nodais

- 31 - Esforços de Extremidade
- 32 - Esforços de Extremidade+Modelo
- 33 - Esforços de Extremidade
- 34 - Esforços de Extremidade

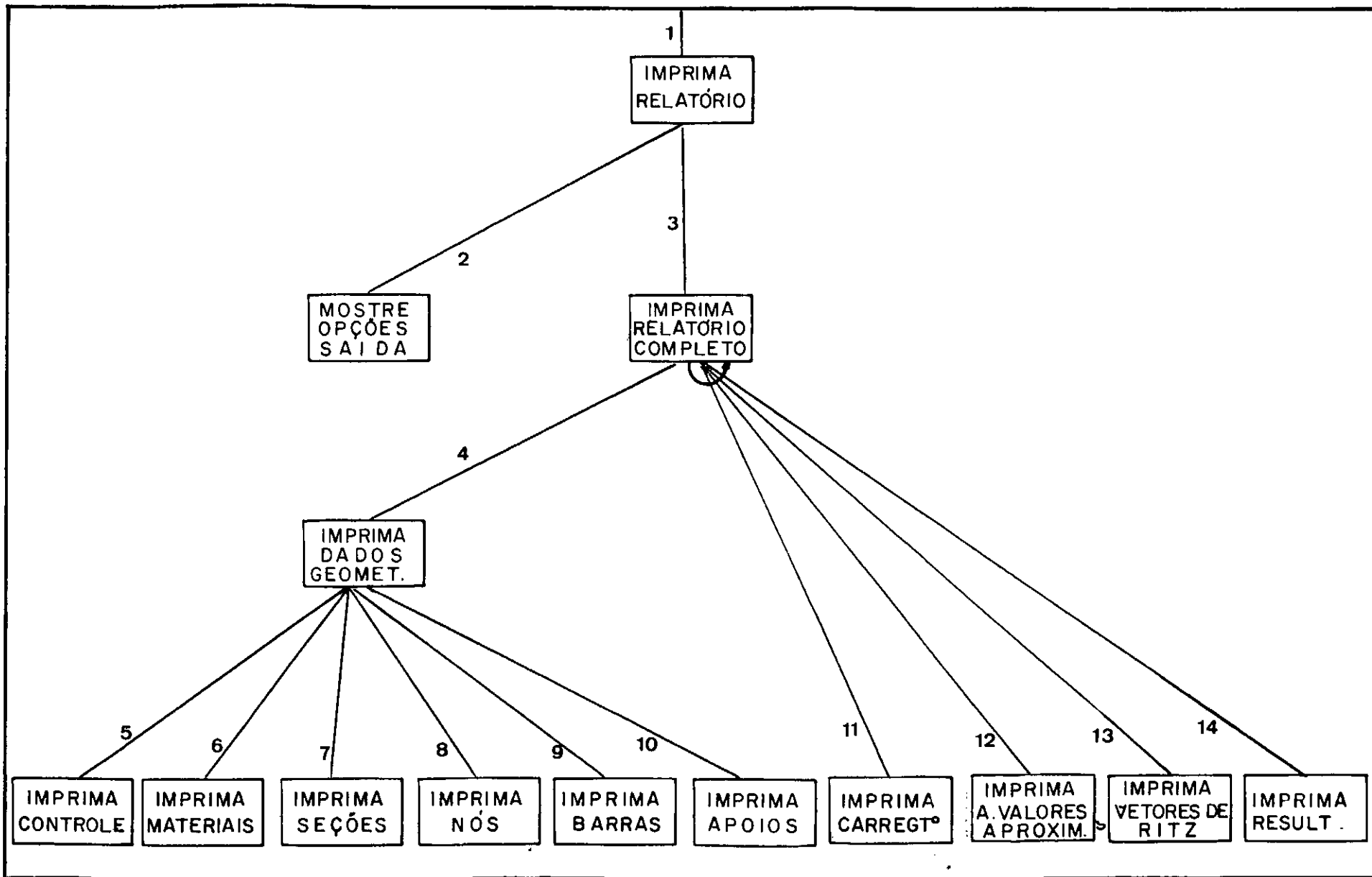


figura IV.26

Parâmetros da figura IV.26

- 1 - Drive+Nome+Controle
- 2 - Opção de Saída
- 3 - Drive+Nome+Controle
- 4 - Drive+Nome+Controle
- 5 - Drive+Nome+Controle
- 6 - Drive+Nome+Controle
- 7 - Drive+Nome+Controle
- 8 - Drive+Nome+Controle
- 9 - Drive+Nome+Controle
- 10 - Drive+Nome+Controle
- 11 - Drive+Nome+Controle
- 12 - Drive+Nome+Controle
- 13 - Drive+Nome+Controle
- 14 - Drive+Nome+Controle

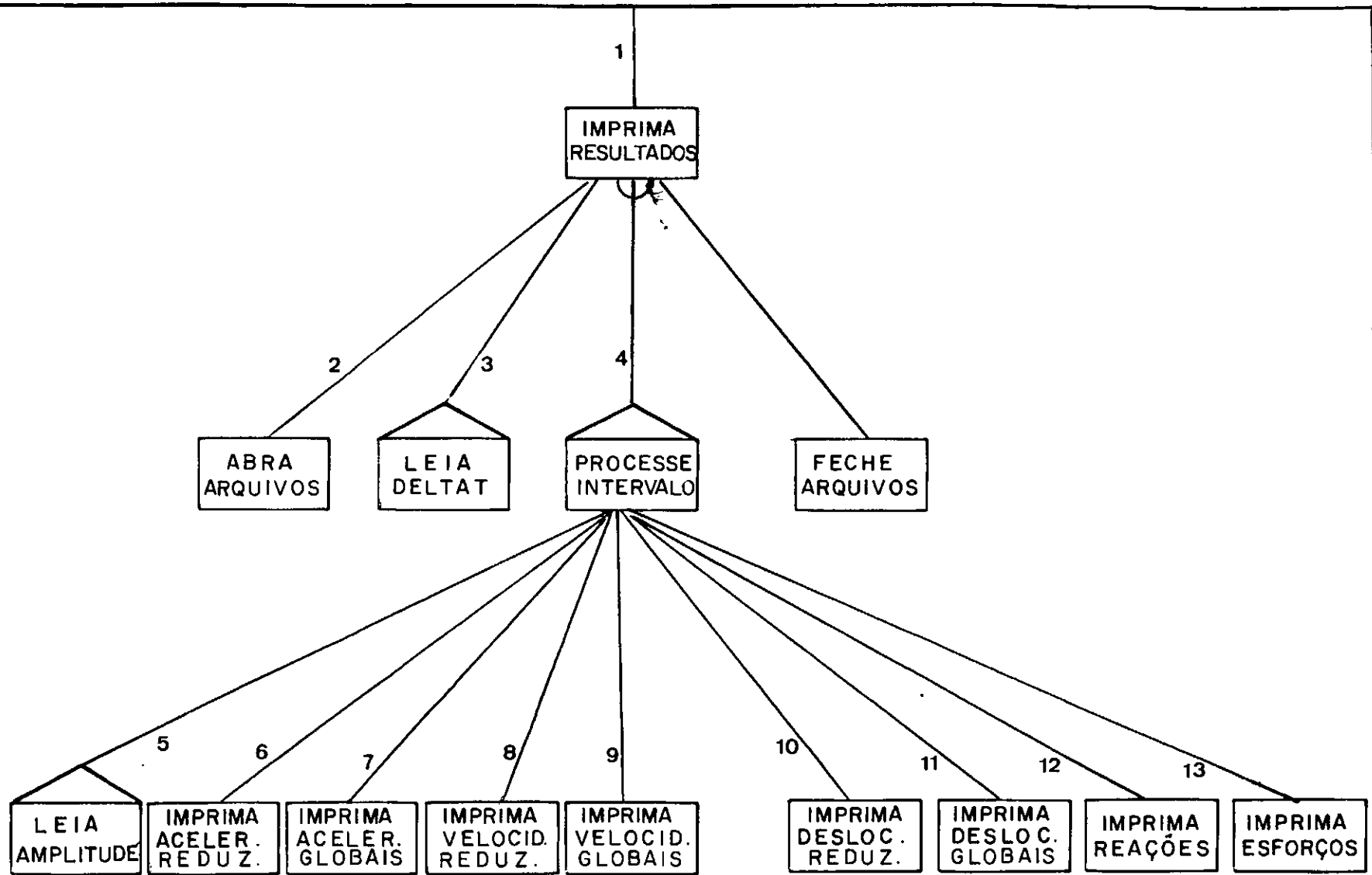


Figura IV.27

Parâmetros da figura IV.27

- 1 - Drive+Nome+Controle
- 2 - Drive+Nome
- 3 - Delta T
- 4 - Drive+Nome+Controle
- 5 - Amplitude
- 6 - Drive+Nome+Controle
- 7 - Drive+Nome+Controle
- 8 - Drive+Nome+Controle
- 9 - Drive+Nome+Controle
- 10 - Drive+Nome+Controle
- 11 - Drive+Nome+Controle
- 12 - Drive+Nome+Controle
- 13 - Drive+Nome+C

Capítulo V - Programação

Neste capítulo serão abordadas as principais partes integrantes das implementações computacionais realizadas. Inicialmente mostra-se a estruturação interna dos dados nos programas e a seguir apresentam-se alguns dos algoritmos empregados nos mesmos.

V.1 - Estruturação de Dados

V.1.1 - Estruturação Tradicional

Os dados usados por programas de análise estrutural são usualmente guardados em vetores que precisam ser alocados na memória central do computador. O dimensionamento fixo destes vetores é dispendioso em termos da memória alocada, pois é necessário superestimar o tamanho destes vetores.

Para amenizar estas condições desfavoráveis é usual, como utilizado por BATHE (1), adotar vetores de trabalho onde são alocados grupos de informações distintos. Pode-se, por exemplo, dimensionar um vetor de trabalho para variáveis reais, outro para variáveis inteiras e um terceiro para variáveis lógicas. Para cada vetor de trabalho é definido um conjunto de índices que servirão para permitir o acesso a cada informação contida no vetor de trabalho correspondente. Desta forma, restringe-se o problema de dimensionamento ao de se arbitrar apenas o tamanho dos vetores de trabalho.

V.1.1.1 - Crítica à Estruturação Tradicional

V.1.1.1.1 - Díficil Entendimento

Mesmo durante a programação, é díficil para o programador se lembrar da localização dos dados dentro dos vetores de trabalho, dificultando desta forma a programação e, principalmente, a depuração dos sistemas.

V.1.1.1.2 - Edição de Dados

A estruturação tradicional dificulta enormemente a edição de dados da estrutura - especialmente adições e exclusões - devido ao aglomeramento de informações distintas dentro de um mesmo vetor de trabalho.

V.1.1.1.3 - Gerenciamento do Uso da Memória

Torna-se muito complexo um gerenciamento eficiente de forma que se tenha em memória apenas os dados necessários em cada etapa da análise.

V.1.1.1.4 - Manutenção e Expansões

Devido à dificuldade de entendimento e à natureza pouco flexível da estruturação tradicional, a manutenção e, principalmente, a expansão do sistema ficam seriamente dificultadas, especialmente se realizadas por outra pessoa que não os autores do programa.

V.1.2 - Modelos Propostos

A linguagem PASCAL permite que se criem estruturas de dados muito eficientes como descritas abaixo.

V.1.2.1 - Registro

É um tipo de variável que pode ter campos internos de tipos diferentes entre si, como por exemplo :

TYPE

```
TipoBarra = RECORD
                Material : INTEGER;
                Secao     : INTEGER;
                NoInicial : INTEGER;
                NoFinal   : INTEGER;
                AnguloBeta: REAL;
            END;
```

VAR

```
Barra : TipoBarra;
```

Definiu-se acima uma variável Barra de tipo TipoBarra que possui cinco campos internos, a saber:

- Material : indica o tipo de material da barra
- Seção : indica o tipo de seção da barra
- NoInicial : indica o Nó Inicial da barra
- NoFinal : indica o Nó Final da barra
- AnguloBeta : indica o valor do ângulo beta da barra

Esta variável Barra é capaz de guardar todas as informações relativas a um determinado elemento estrutural.

V.1.2.2 - Vetor de Registros

Como a análise estrutural normalmente envolve vários elementos, pode-se definir um vetor com N componentes, constituindo cada componente um registro, como o exemplo abaixo :

```

CONST
    MaxBarras = 1000;

TYPE
    TipoBarra = RECORD
        Material : INTEGER;
        Secao    : INTEGER;
        NoInicial : INTEGER;
        NoFinal  : INTEGER;
        AnguloBeta: REAL;
    END;

    Tabela = ARRAY[1..MaxBarras] OF TipoBarra;

VAR
    Barra : Tabela;

```

A variável Barra descrita acima é um vetor que possui 1000 registros, tendo cada registro cinco campos com as informações relativas a cada elemento.

Uma crítica que se pode fazer à estruturação acima é devido ao fato de se ter permanentemente alocada memória suficiente para 1000 barras, não importando quantos elementos possui a estrutura a ser analisada.

V.1.2.3 - Vetor de Ponteiros para Registros

A solução pra o problema anteriormente ressaltado é a adoção do vetor de ponteiros para registros.

Ponteiros são tipos de variáveis que guardam o endereço de uma posição de memória e, por meio do seu uso, pode-se alocar e desalocar memória à medida que seja necessário.

Esta estrutura de dados é muito semelhante à anterior, com a diferença que em lugar de guardar registros, o vetor agora guarda ponteiros, que são endereços que podem conter registros.

CONST

MaxBarras = 1000;

TYPE

Ponteiro = ^TipoBarra;

TipoBarra = RECORD

Material : INTEGER;

Secao : INTEGER;

NoInicial : INTEGER;

NoFinal : INTEGER;

AnguloBeta:REAL;

END;

Tabela = ARRAY[1..MaxBarras] OF Ponteiro;

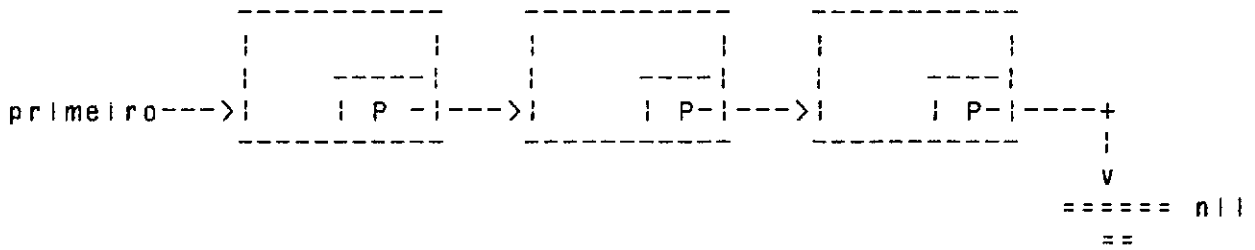
VAR

Barra : Tabela;

Cada ponteiro usa 4 bytes da memória, em lugar dos 14 usados por cada registro TipoBarra, o que totaliza um gasto fixo de 4000 bytes em lugar dos 14000 anteriores. À medida que se alocam barras no vetor, gastam-se mais 14 bytes por barra.

V.1.2.4 - Listas Encadeadas com Ponteiros

Este tipo de estrutura de dados possui o aspecto de um trem onde se pode acessar o primeiro vagão e, por meio deste, os demais, conforme ilustrado na figura :



Cada registro possui um campo do tipo ponteiro que é usado para guardar o endereço do registro seguinte. O endereço do primeiro elemento é guardado em uma variável do tipo ponteiro e o último elemento da lista aponta para um endereço especial que indica o fim da lista. Por exemplo,

TYPE

```

Ponteiro = ^TipoBarra;
TipoBarra = RECORD
                Material : INTEGER;
                Secao     : INTEGER;
                NoInicial : INTEGER;
                NoFinal   : INTEGER;
                AnguloBeta: REAL;
                Proxima   : Ponteiro;
            END;

```

VAR

```

Primeira,ultima,atual : Ponteiro;

```

Quais as vantagens e desvantagens do uso de listas encadeadas em relação ao uso de vetores de ponteiros ?

Vantagens :

- Não é necessário definir um número máximo de barras.
- Não há um gasto inicial de memória devido aos ponteiros do vetor.

Desvantagens :

- O acesso a cada registro só pode ser feito sequencialmente a partir do primeiro elemento da lista.
- Só é indicada para processamento sequencial dos registros.

V.1.3 - Proposta para Análise Estrutural

A decisão de usar o modelo de vetor de ponteiros ou o modelo de listas encadeadas para cada uma das estruturas de dados neste item foi tomada em função da forma de acesso a estas informações pelo programa de análise estrutural. Para as informações que são acessadas sequencialmente, optou-se pelo modelo de lista encadeada e usou-se o modelo de vetor de ponteiros no caso das informações cujo acesso não obedecia a uma ordem sequencial.

Na estruturação abaixo não estão incluídas estruturas internas do programa tais como vetor independente ou matriz de rigidez da estrutura. Estas informações podem ser guardadas em vetores reais, que é a forma natural de tratar tais dados.

const

```
NosPorElemento      =      2;
MaxMateriais        =      5;
MaxSecoes           =     100;
MaxNos              =    1000;
MaxCarregamentos    =     20;
MaxDeslocamentos    =    6000;
MaxRitz             =     10;
```

type

```
TiposdeEstrutura    = (PorticoEspacial,PorticoPlano,
                       TrelicaEspacial,TrelicaPlana,Grelha);
SituacaoCarga       = (LocalXY,LocalXZ,GlobalY,GlobalX,GlobalZ);
ApontaMaterial      = ^TipoMaterial;
ApontaSecao         = ^TipoSecao;
ApontaNo            = ^TipoNo;
ApontaCarga         = ^TipoCarga;
Estado              = record
                       TipoEstrutura:TiposdeEstrutura;
                       Amortecimento:Real;
```

```

end;

TipoMaterial      = record
    E,G,Densidade:Real;
end;

TipoSecao        = record
    Ax,Ay,Az,lx,ly,lz:Real;
end;

TipoNo           = record
    Coord:array[1..3] of Real;
end;

TipoBarra        = record
    Material,Secao      :Integer;
    Beta                :Real;
    No                  :array[1..NosporElemento]
                        of Integer;
    TemLiberacao       :Boolean;
    Liberacao           :String[12];
end;

TipoApoio        = record
    NumeroNo:Integer;
    Codigo   :String[6];
    Condicao:array[1..6] of Real;
end;

TipoCarga        = record
    proxima      :ApontaCarga;
    Barra        :Integer;
    Situacao     :SituacaoCarga;
    case TipoCarga:Integer of
        1:(CargaUnif:Real);
        2:(CargaConc,DistConc:Real);
    end;
end;

```

```
3:(CargaParc,DistParc,Comp:Real);
```

```
4:(CargaEsq,CargaDir,DistTrap,CompTrap:Real);
```

```
5:(DeltaT,Alfa1:Real);
```

```
6:(Alfa2,t1,t2,h:Real);
```

```
7:(Forca:array[1..6] of Real);
```

```
end;
```

```
TipoReacao = record
```

```
NumeroNo:Integer;
```

```
Valor :array[1..6] of Real;
```

```
end;
```

```
TipoEsforcoExtr = record
```

```
Valor :array[1..12] of Real;
```

```
end;
```

```
var
```

```
  Materiais,Secoes,Nos,
```

```
  Barras,Apoios,
```

```
  DeslocamentosporNo,
```

```
  Coordenadas,
```

```
  Carregamentos : Integer;
```

```
  Fim : Boolean;
```

```
  Controle : Estado;
```

```
  drive : Char;
```

```
  Nome : String[8];
```

```
  Material : array[1..MaxMateriais] of ApontaMaterial;
```

```
  Secao : array[1..MaxSecoes] of ApontaSecao;
```

```
  No : array[1..MaxNos] of ApontaNo;
```

```
  PrimeiraCarga,
```

```
  UltimaCarga : ApontaCarga;
```

```
  ArqControle : file of Estado;
```

```
  ArqMaterial : file of TipoMaterial;
```

```
  ArqSecao : file of TipoSecao;
```

ArqNo : file of TipoNo;
ArqBarra : file of TipoBarra;
ArqApoio : file of TipoApoio;
ArqReacao : file of TipoReacao;
ArqEsforcoExtr : file of TipoEsforcoExtr;
ArqCarga : file of TipoCarga;

V.2 - Principais Algoritmos Usados

V.2.1 - Montagem da Matriz de Rigidez Global

Inicialmente considere-se o algoritmo de montagem da matriz de rigidez completa da estrutura.

para i ← 1 até TotalDeBarras faça

 início

 Obtenha(Ke, i)

 direção ← 0

 para j ← 1 até 2 faça

 para k ← 1 até DeslocamentosPorNó faça

 início

 direção ← direção + 1

 direçãoGlobal ← DeslocamentosPorNó*(Incidência[i, j]-1)+k

 JK[direção] ← direçãoGlobal

 fim

 para j ← 1 até DeslocamentosPorBarra faça

 para k ← 1 até DeslocamentosPorBarra faça

 se k >= j então

 K[JK[j], JK[k]] ← -K[JK[j], JK[k]] + Ke[j, k]

 fim

A matriz de rigidez da estrutura foi implementada usando-se a técnica de armazenamento por altura efetiva de coluna, conforme SORIANO (3) 1983. Desta forma, tem-se que a matriz é guardada em um vetor unidimensional cujo acesso se faz segundo a equação

$$K[i, j] = K[i+P[j]-j] \quad \text{V.1}$$

onde P é o vetor apontador descrito na referência supracitada. O algoritmo de montagem da matriz de rigidez fica então da seguinte forma :

```

para i <- 1 até TotalDeBarras faça
  inicio
    Obtenha(Ke,i)
    direção <- 0
    para j <- 1 até 2 faça
      para k <- 1 até DeslocamentosPorNó faça
        inicio
          direção <- direção + 1
          direçãoGlobal <- DeslocamentosPorNó*(Incidência[i,j]-1)+k
          JK[direção]<- direçãoGlobal
        fim
      para j <- 1 até DeslocamentosPorBarra faça
        para k <- 1 até DeslocamentosPorBarra faça
           $K[JK[j]]+P[JK[k]]-JK[k]] <- K[JK[j]]+P[JK[k]]-JK[k]]+Ke[j,k]$ 
        fim
      fim
    fim
  fim
fim

```

V.2.2 - Blocagem da Matriz de Rigidez

Seja uma estrutura reticulada com 6.000 deslocamentos nodais. Considerando-se a simetria e uma esparcidade de 95 % na matriz de rigidez, tem-se, de forma aproximada, $0,5 \times 0,05 \times 6.000 \times 6.000 = 900.000$ termos reais a serem manipulados pelo programa de análise estrutural. No caso do compilador usado, estes termos ocupariam $900.000 \times 6 = 5.400.000$ bytes, quantidade muito superior à memória do equipamento usado, que é de 720.896 bytes. Foi então necessária a utilização de técnicas de blocagem da matriz de rigidez de forma que a análise fosse possível.

A solução adotada utiliza a técnica de armazenamento por altura efetiva de coluna, conforme apresentada em SORIANO (3) 1983 e divide o arranjo unidimensional resultante em blocos de 6.000 termos que são guardados em disco e alocados dinamicamente em memória central por meio de um vetor de ponteiros. O acesso aos termos da matriz é feito por meio de rotinas que têm como parâmetros o número do bloco e o índice do elemento a ser acessado. A rotina testa se o bloco está em memória central. Se estiver, a informação é acessada e fornecida; se não, é chamada uma outra rotina para ler o bloco do disco. Esta outra rotina testa se há memória disponível para alocar o bloco na memória. Se não houver memória suficiente, é chamada uma terceira rotina que selecionará o bloco mais antigo em memória e o deslocará de forma a liberar memória para o bloco a ser lido.

Esta técnica é muito simples e torna o gerenciamento da memória transparente para as rotinas que manipulam a matriz de rigidez. Há também a vantagem do gerenciamento ser genérico e, portanto, utilizável com outras estruturas de dados como, por exemplo, os vetores de Ritz do programa de análise dinâmica.

Da referência SORIANO (3) 1983, definindo-se um arranjo unidimensional V, tem-se que o termo $K[i, j]$ da matriz de rigidez corresponde ao termo $V[i + P[j] - j]$ do arranjo unidimensional utilizado, onde P é o vetor apontador.

A implementação da blocagem da matriz de rigidez se fez por meio de duas rotinas, Ponha e Pegue, de forma que em lugar de ter-se, por exemplo,

$$K[i + P[j] - j] := K[i + P[j] - j] + Aux \quad V.2$$

tem-se

$$\text{Ponha}(\text{Bloco}, \text{Índice}, \text{Pegue}(\text{Bloco}, \text{Índice}) + Aux) \quad V.3$$

onde Aux é uma expressão real qualquer, Bloco é o número do bloco e Índice a posição dentro deste bloco onde está o termo i, j da matriz. A função Pegue possui dois parâmetros, o bloco e o Índice, e retorna o valor do termo. A procedure Ponha possui um terceiro parâmetro que é o valor a ser posto no local especificado por Bloco e Índice.

V.2.3 - Resolução do Sistema de Equações

Escolheu-se o método de Cholesky para a resolução do sistema de equações. É um método direto que utiliza transformações elementares e que se mostra eficiente em estudos realizados por SORIANO (6) 1981. Este método se aplica, em sua forma clássica, a matrizes simétricas positivas definidas e consiste em decompor a matriz do sistema na forma

$$K = U^t U \quad V.4$$

onde U é uma matriz triangular superior denominada Fator de Cholesky, conforme SORIANO (3) 1983. Substituindo-se (V.4) em (II.1), tem-se

$$U^t U d = f \quad V.5$$

seja

$$y = U d \quad V.6$$

logo

$$U^t y = f \quad V.7$$

Solucionando-se inicialmente o sistema (V.6) por meio de um processo de substituição, obtém-se o vetor y . Em seguida resolve-se o sistema (V.7) por um processo de retrossubstituição, obtendo-se o vetor solução d . Apresenta-se a seguir os algoritmos correspondentes às etapas de triangularização, substituição e retrossubstituição conforme SORIANO (3) 1983.

V.2.3.1 - Triangularização

```
K[1] <- SQRT(K[1])
```

```
para colunaj <- 2 até N faça
```

```
  inicio
```

```
    AlturaEfetivaj <- P[colunaj]-P[colunaj-1]
```

```
    termoj <- P[colunaj-1]+1
```

```
    linhaj <- colunaj-AlturaEfetivaj+1
```

```
    se linhaj <> colunaj então
```

```
      inicio
```

```
        K[termoj] <- K[termoj]/K[P[linhaj-1]]
```

```
        para linhai <- linhaj+1 até colunaj faça
```

```
          inicio
```

```
            AlturaEfetivai <- P[linhai] - P[linhai-1]
```

```
            termoi <- linhai - AlturaEfetivai + 1
```

```
            termoj <- termoj + 1
```

```
            se linhaj < termoi então MaiorLinha <- termoi
```

```
              senão MaiorLinha <- linhaj
```

```
            ci <- P[linhai]-linhai+MaiorLinha
```

```
            cj <- P[linhaj]-linhaj+MaiorLinha
```

```
            para linha <- MaiorLinha até linhai-1 faça
```

```
              inicio
```

```
                K[termoj] <- K[termoj] -K[Ci]*K[Cj]
```

```
                Ci <- Ci+1
```

```
                Cj <- Cj +1
```

```
              fim
```

```
            se linhai <> colunaj então K[termoj] <- K[termoj]/K[P[linhai-1]]
```

```
          fim
```

```
    fim
```

```
K[termoj] <- SQRT(K[termoj])
```

```
fim
```

V.2.3.2 - Substituição

```

d[1] <- d[1]/K[1]
para i <- 2 até N faça
  início
    AlturaEfetiva <- P[i] - P[i-1]
    linha <- i - AlturaEfetiva + 1
    termo <- P[i-1]
    aux <- d[termo]
    para j <- linha até termo-1 faça
      início
        termo <- termo + 1
        aux <- aux - d[j]*K[termo]
      fim
    d[termo] <- aux/K[P[termo]]
  fim
fim

```

V.2.3.3 - Retrosubstituição

```

para i <- N até 2 faça
  início
    d[i] <- d[i]/K[P[i]]
    AlturaEfetiva <- P[i] - P[i-1]
    linha <- i - AlturaEfetiva + 1
    termo <- P[i-1]
    para j <- linha até i-1 faça
      início
        termo <- termo + 1
        d[j] <- d[j] - K[termo]*d[i]
      fim
    fim
fim
d[1] <- d[1]/K[1]

```

V.2.4 - Ortogonalização e normalização dos Vetores de Ritz

A montagem dos vetores de Ritz usados para reduzir-se a ordem do sistema de equações diferenciais segue os algoritmos apresentados por WILSON e alii (2). Estes vetores são gerados a partir da seguinte relação de recorrência :

$$K x_i = M x_{i-1} \quad i = 2 \dots N \quad V.8$$

O primeiro vetor é obtido como solução de

$$K x_1 = f \quad V.9$$

Os vetores são então ortogonalizados

$$x_i = x_i^* - c_j x_j \quad j = 1 \dots i-1 \quad V.10$$

onde

$$c_j = x_j^t M x_i^* \quad V.11$$

Após a ortogonalização, o vetor é normalizado de forma que

$$x_j^t M x_j = 1 \quad V.12$$

A seguir, apresenta-se o pseudo-código das rotinas de ortogonalização e normalização :

```

infcio [ ortogonalização do vetor i ]
  para j <- 1 até i-1 faça [ cálculo de cj ]
    infcio
      c[j] <- 0
      para k <- 1 até N faça
        c[j] <- c[j] + Ritz[j,k] * M[k] * Ritz[i,k]
      fim
  para j <- 1 até i-1 faça
    para k <- 1 até N faça
      Ritz[i,k] <- Ritz[i,k] - c[j] * Ritz[j,k]
fim [ fim da rotina de ortogonalização ]

infcio [ infcio da rotina de normalização ]
  aux <- 0
  para j <- 1 até N faça aux <- aux + M[j]*Sqr(Ritz[i,j])
  para j <- 1 até N faça Ritz[i,j] <- Ritz[i,j]/aux
fim

```

V.2.5 - Resolução do Sistema de Equações Diferenciais

Conforme exposto no capítulo 2, uma das etapas da solução proposta para análise dinâmica é a integração do sistema de equações diferenciais desacoplado. Tem-se a equação

$$x_i(t) + 2c_i w_i x_i \dot{x}_i(t) + w_i^2 x_i(t) = p_i g(t) \quad V.13$$

cuja solução é obtida por meio da formulação apresentada por LANDAU (4). Seja t_0 o instante que define o início do intervalo de integração considerado. Tem-se

$$t_1 = t - t_0 \quad V.14$$

$$p(t_1) = a + b * t_1 \quad V.15$$

onde

$$a = p(t_0) \quad V.16$$

$$b = (p(t_1) - p(t_0)) / (t_1 - t_0) \quad V.17$$

figura

A equação (V.13) fica

$$x_i(t_1) + 2c_i w_i x_i \dot{x}_i(t_1) + w_i^2 x_i(t_1) = a + b \cdot t_1 \quad \text{V.18}$$

a solução geral para sistemas sub-amortecidos pode ser escrita como a equação V.19 :

$$x(t_1) = A_0 + A_1 t_1 + A_2 e^{-c w t_1} \cos w_1 t_1 + A_3 e^{-c w t_1} \sin w_1 t_1$$

$$\dot{x}(t_1) = A_1 + e^{-c w t_1} [(A_3 w_1 - A_2 c w) \cos w_1 t_1 - (A_2 w_1 + A_3 c w) \sin w_1 t_1]$$

$$\ddot{x}(t_1) = e^{-c w t_1} [(A_2 c^2 w^2 - A_2 w_1^2 - 2A_3 c w w_1) \cos w_1 t_1 - (A_3 c^2 w^2 + A_3 w_1^2 - 2A_2 c w w_1) \sin w_1 t_1]$$

onde

$$w_1 = w_i \cdot \text{SQRT}(1 - c \cdot c)$$

$$A_0 = (a - 2bc/w_i) / (w_i \cdot w_i)$$

$$A_1 = b/w_i \quad \text{V.20}$$

$$A_2 = x(t_0) - A_0$$

$$A_3 = (x(t_0) + A_2 c w - A_1) / w_1$$

Capítulo VI - Exemplos, Conclusões e Sugestões

VI.1 - Exemplos

Estes exemplos foram analisados pelo programa DINA, cujo projeto encontra-se descrito neste trabalho e pelo Sap IV instalado em um VAX 785 da PROMON. O Sap IV analisou os modelos por meio do método de superposição modal com 8 autovetores. O microcomputador usado foi um SID 501 com clock de 5.47 MHz e disco rígido. Seu desempenho, segundo o programa SI (Norton Utilities) é equivalente ao de um IBM-PC.

VI.1.1 - Exemplo 1

Informações gerais sobre a estrutura analisada :

```

Modelo usado na análise   : Trelica Plana
Numero de Materiais..... :      1
Numero de Secoes.....    :      1
Numero de Nos.....       :      6
Numero de Barras.....    :      9
Numero de Apolos.....    :      2
Numero de Carregamentos  :      1
  
```

Constantes Elasticas e Peso Especifico dos Materiais Utilizados

```

Material 1  E = 210000000000.000
            G = 105000000000.000
            u =                0.801
  
```

Características Geometricas das Secoes

```

Secao 1  Ax = 0.000100
  
```

Coordenadas dos Nos da Estrutura

No	1	X =	0.000000	Y =	5.000000
No	2	X =	4.000000	Y =	5.000000
No	3	X =	0.000000	Y =	2.500000
No	4	X =	4.000000	Y =	2.500000
No	5	X =	0.000000	Y =	0.000000
No	6	X =	4.000000	Y =	0.000000

Informacoes sobre as barras que compoem a estrutura :

Barra	1	No I =	1	No J =	2	Mat =	1	Sec =	1
Barra	2	No I =	1	No J =	4	Mat =	1	Sec =	1
Barra	3	No I =	3	No J =	4	Mat =	1	Sec =	1
Barra	4	No I =	5	No J =	4	Mat =	1	Sec =	1
Barra	5	No I =	5	No J =	6	Mat =	1	Sec =	1
Barra	6	No I =	1	No J =	3	Mat =	1	Sec =	1
Barra	7	No I =	2	No J =	4	Mat =	1	Sec =	1
Barra	8	No I =	3	No J =	5	Mat =	1	Sec =	1
Barra	9	No I =	4	No J =	6	Mat =	1	Sec =	1

Condicoes Prescritas nos Apoios

No	5	desl X =	0.000000000	desl Y =	0.000000000
No	6	desl X =	0.000000000	desl Y =	0.000000000

Carregamento 1

Cargas Nodais :

No	direcao 1	direcao 2
1	0.500	0.000

Variação da amplitude no tempo

t	f(t)
0.00	0.000
0.02	0.070
0.04	0.156
0.06	0.233
0.08	0.309
0.10	0.382
0.12	0.454
0.14	0.522
0.16	0.587
0.18	0.649
0.20	0.707
0.22	0.760
0.24	0.809
0.26	0.852
0.28	0.891
0.30	0.923
0.32	0.951
0.34	0.972
0.36	0.988
0.38	0.998
0.40	1.000

Numero de vetores de Ritz adotados = 8

Tempo de Processamento

SAP IV = 6.57 s

Dinâmica = 178.0 s

Relação Dinâmica/SAP IV = 27

Deslocamentos obtidos

t	No	dir	Dinamica	Sap IV
0.20	1	1	3.195e-07	3.194e-07
0.20	1	2	5.262e-08	5.259e-08
0.20	3	1	1.433e-07	1.432e-07
0.20	3	2	2.631e-08	2.629e-08
0.40	1	1	7.440e-08	7.438e-08
0.40	1	2	4.518e-07	4.517e-07
0.40	3	1	3.720e-08	3.719e-08
0.40	3	2	2.026e-08	2.026e-08

Neste exemplo os resultados estão praticamente idênticos. Isto porque o número de vetores escolhidos é igual à dimensão do espaço vetorial do problema. Podemos então concluir que neste caso a solução por superposição modal tradicional é equivalente à solução usando vetores de Ritz.

VI.1.2 - Exemplo 2

Informacoes gerais sobre a estrutura analisada :

Modelo usado na analise : Portico Espacial
 Numero de Materiais.... : 1
 Numero de Secoes..... : 2
 Numero de Nos..... : 12
 Numero de Barras..... : 16
 Numero de Apoios..... : 4
 Numero de Carregamentos : 1

Constantes Elasticas e Peso Especifico dos Materiais Utilizados

Material 1 E = 210000000000.000
 G = 63000000000.000
 u = 2.500

Caracteristicas Geometricas das Secoes

Secao 1 Ax = 0.040000 Ay = 0.000000 Az = 0.000000
 Ix = 0.000000 Iy = 0.000133 Iz = 0.000133
 Secao 2 Ax = 0.100000 Ay = 0.000000 Az = 0.000000
 Ix = 0.000000 Iy = 0.000042 Iz = 0.002083

Coordenadas dos Nos da Estrutura

No 1 X = 0.000000 Y = 0.000000 Z = 0.000000
 No 2 X = 5.000000 Y = 0.000000 Z = 0.000000
 No 3 X = 5.000000 Y = 0.000000 Z = 5.000000
 No 4 X = 0.000000 Y = 0.000000 Z = 5.000000
 No 5 X = 0.000000 Y = 3.000000 Z = 0.000000
 No 6 X = 5.000000 Y = 3.000000 Z = 0.000000
 No 7 X = 5.000000 Y = 3.000000 Z = 5.000000
 No 8 X = 0.000000 Y = 3.000000 Z = 5.000000
 No 9 X = 0.000000 Y = 6.000000 Z = 0.000000
 No 10 X = 5.000000 Y = 6.000000 Z = 0.000000

No 11 X = 5.000000 Y = 6.000000 Z = 5.000000
 No 12 X = 0.000000 Y = 6.000000 Z = 5.000000

Informacoes sobre as barras que compoem a estrutura :

Barra 1 No I = 1 No J = 5 Mat = 1 Sec = 1
 Barra 2 No I = 2 No J = 6 Mat = 1 Sec = 1
 Barra 3 No I = 3 No J = 7 Mat = 1 Sec = 1
 Barra 4 No I = 4 No J = 8 Mat = 1 Sec = 1
 Barra 5 No I = 5 No J = 6 Mat = 1 Sec = 2
 Barra 6 No I = 6 No J = 7 Mat = 1 Sec = 2
 Barra 7 No I = 7 No J = 8 Mat = 1 Sec = 2
 Barra 8 No I = 8 No J = 5 Mat = 1 Sec = 2
 Barra 9 No I = 5 No J = 9 Mat = 1 Sec = 1
 Barra 10 No I = 6 No J = 10 Mat = 1 Sec = 1
 Barra 11 No I = 7 No J = 11 Mat = 1 Sec = 1
 Barra 12 No I = 8 No J = 12 Mat = 1 Sec = 1
 Barra 13 No I = 9 No J = 10 Mat = 1 Sec = 2
 Barra 14 No I = 10 No J = 11 Mat = 1 Sec = 2
 Barra 15 No I = 11 No J = 12 Mat = 1 Sec = 2
 Barra 16 No I = 12 No J = 9 Mat = 1 Sec = 2

Condicoes Prescritas nos Apoios

No 1 desl X = 0.000 desl Y = 0.000 desl Z = 0.000
 mola X = 0.000 mola Y = 0.000 mola Z = 0.000
 No 2 desl X = 0.000 desl Y = 0.000 desl Z = 0.000
 mola X = 0.000 mola Y = 0.000 mola Z = 0.000
 No 3 desl X = 0.000 desl Y = 0.000 desl Z = 0.000
 mola X = 0.000 mola Y = 0.000 mola Z = 0.000
 No 4 desl X = 0.000 desl Y = 0.000 desl Z = 0.000
 mola X = 0.000 mola Y = 0.000 mola Z = 0.000

Carregamento 1

Cargas Nodais :

No	direcao 1	direcao 2	direcao 3
9	10.000	0.000	0.000
12	10.000	0.000	0.000

Variacao da amplitude no tempo

t	f(t)
0.00	0.00
0.10	0.10
0.20	0.20
0.30	0.30
0.40	0.40
0.50	0.50
0.60	0.60
0.70	0.70
0.80	0.80
0.90	0.90
1.00	1.00
1.10	0.90
1.20	0.80
1.30	0.70
1.40	0.60
1.50	0.50
1.60	0.40
1.70	0.30
1.80	0.20
1.90	0.10
2.00	0.00

Numero de vetores de Ritz adotados = 8

Tempo de Processamento

SAP IV = 9.69 s

Dinâmica = 630.0 s

Relação Dinâmica/SAP IV = 65

Deslocamentos obtidos

t	No	dir	Dinamica	Sap IV
0.20	9	1	4.4657e-07	5.329e-07
0.40	9	1	8.9386e-07	9.492e-07
0.60	9	1	1.3397e-06	1.374e-06
0.80	9	1	1.7880e-06	1.807e-06
1.00	9	1	2.2328e-06	2.246e-06
1.20	9	1	1.7887e-06	1.622e-06
1.40	9	1	1.3381e-06	1.233e-06
1.60	9	1	8.9640e-07	8.285e-07
1.80	9	1	4.4329e-07	4.075e-07
2.00	9	1	9.4337e-09	8.941e-09

Neste exemplo temos um espaço vetorial bem maior que no exemplo anterior e tanto a solução por superposição modal quanto a por vetores de Ritz são aproximadas. Neste caso os resultados também foram próximos.

VI.1.3 - Exemplo 3

Informacoes gerais sobre a estrutura analisada :

Modelo usado na analise : Portico Plano

Numero de Materiais.... : 1

Numero de Secoes..... : 5

Numero de Nos..... : 49

Numero de Barras..... : 63

Numero de Apoios..... : 4

Numero de Carregamentos : 1

Constantes Elasticas e Peso Especifico dos Materiais Utilizados

Material 1 E = 210000000000.000

G = 105000000000.000

u = 0.000

Caracteristicas Geometricas das Secoes

Secao 1 Ax = 0.175000 Ay = 0.000000 Iz = 0.007150

Secao 2 Ax = 0.500000 Ay = 0.000000 Iz = 0.072000

Secao 3 Ax = 0.240000 Ay = 0.000000 Iz = 0.012800

Secao 4 Ax = 0.100000 Ay = 0.000000 Iz = 0.002080

Secao 5 Ax = 0.360000 Ay = 0.000000 Iz = 0.010800

Coordenadas dos Nos da Estrutura

No 1 X = 0.000000 Y = 0.000000

No 2 X = 3.500000 Y = 0.000000

No 3 X = 7.000000 Y = 0.000000

No 4 X = 10.500000 Y = 0.000000

No 5 X = 14.000000 Y = 0.000000

No 6 X = 17.500000 Y = 0.000000

No 7 X = 21.000000 Y = 0.000000

No 8 X = 0.000000 Y = 2.800000

No 9 X = 3.500000 Y = 2.800000

No	10	X =	7.000000	Y =	2.800000
No	11	X =	10.500000	Y =	2.800000
No	12	X =	14.000000	Y =	2.800000
No	13	X =	17.500000	Y =	2.800000
No	14	X =	21.000000	Y =	2.800000
No	15	X =	0.000000	Y =	7.700000
No	16	X =	2.000000	Y =	7.700000
No	17	X =	5.000000	Y =	7.700000
No	18	X =	7.000000	Y =	7.700000
No	19	X =	9.000000	Y =	7.700000
No	20	X =	12.000000	Y =	7.700000
No	21	X =	14.000000	Y =	7.700000
No	22	X =	16.000000	Y =	7.700000
No	23	X =	19.000000	Y =	7.700000
No	24	X =	21.000000	Y =	7.700000
No	25	X =	0.000000	Y =	11.300000
No	26	X =	3.500000	Y =	11.300000
No	27	X =	7.000000	Y =	11.300000
No	28	X =	10.500000	Y =	11.300000
No	29	X =	14.000000	Y =	11.300000
No	30	X =	17.500000	Y =	11.300000
No	31	X =	21.000000	Y =	11.300000
No	32	X =	0.000000	Y =	14.900000
No	33	X =	7.000000	Y =	14.900000
No	34	X =	14.000000	Y =	14.900000
No	35	X =	21.000000	Y =	14.900000
No	36	X =	0.000000	Y =	18.200000
No	37	X =	3.500000	Y =	18.200000
No	38	X =	7.000000	Y =	18.200000
No	39	X =	10.500000	Y =	18.200000
No	40	X =	14.000000	Y =	18.200000

No 41	X =	17.500000	Y =	18.200000
No 42	X =	21.000000	Y =	18.200000
No 43	X =	0.000000	Y =	22.600000
No 44	X =	3.500000	Y =	22.600000
No 45	X =	7.000000	Y =	22.600000
No 46	X =	10.500000	Y =	22.600000
No 47	X =	14.000000	Y =	22.600000
No 48	X =	17.500000	Y =	22.600000
No 49	X =	21.000000	Y =	22.600000

Informacoes sobre as barras que compoem a estrutura :

Barra 1	No I =	1	No J =	2	Mat =	1	Sec =	1
Barra 2	No I =	2	No J =	3	Mat =	1	Sec =	1
Barra 3	No I =	3	No J =	4	Mat =	1	Sec =	1
Barra 4	No I =	4	No J =	5	Mat =	1	Sec =	1
Barra 5	No I =	5	No J =	6	Mat =	1	Sec =	1
Barra 6	No I =	6	No J =	7	Mat =	1	Sec =	1
Barra 7	No I =	8	No J =	9	Mat =	1	Sec =	1
Barra 8	No I =	9	No J =	10	Mat =	1	Sec =	1
Barra 9	No I =	10	No J =	11	Mat =	1	Sec =	1
Barra 10	No I =	11	No J =	12	Mat =	1	Sec =	1
Barra 11	No I =	12	No J =	13	Mat =	1	Sec =	1
Barra 12	No I =	13	No J =	14	Mat =	1	Sec =	1
Barra 13	No I =	15	No J =	16	Mat =	1	Sec =	2
Barra 14	No I =	16	No J =	17	Mat =	1	Sec =	2
Barra 15	No I =	17	No J =	18	Mat =	1	Sec =	2
Barra 16	No I =	18	No J =	19	Mat =	1	Sec =	2
Barra 17	No I =	19	No J =	20	Mat =	1	Sec =	2
Barra 18	No I =	20	No J =	21	Mat =	1	Sec =	2
Barra 19	No I =	21	No J =	22	Mat =	1	Sec =	2
Barra 20	No I =	22	No J =	23	Mat =	1	Sec =	2

Barra 21	No I =	23	No J =	24	Mat =	1	Sec =	2
Barra 22	No I =	25	No J =	26	Mat =	1	Sec =	3
Barra 23	No I =	26	No J =	27	Mat =	1	Sec =	3
Barra 24	No I =	27	No J =	28	Mat =	1	Sec =	3
Barra 25	No I =	28	No J =	29	Mat =	1	Sec =	3
Barra 26	No I =	29	No J =	30	Mat =	1	Sec =	3
Barra 27	No I =	30	No J =	31	Mat =	1	Sec =	3
Barra 28	No I =	36	No J =	37	Mat =	1	Sec =	4
Barra 29	No I =	37	No J =	38	Mat =	1	Sec =	4
Barra 30	No I =	38	No J =	39	Mat =	1	Sec =	4
Barra 31	No I =	39	No J =	40	Mat =	1	Sec =	4
Barra 32	No I =	40	No J =	41	Mat =	1	Sec =	4
Barra 33	No I =	41	No J =	42	Mat =	1	Sec =	4
Barra 34	No I =	43	No J =	44	Mat =	1	Sec =	3
Barra 35	No I =	44	No J =	45	Mat =	1	Sec =	3
Barra 36	No I =	45	No J =	46	Mat =	1	Sec =	3
Barra 37	No I =	46	No J =	47	Mat =	1	Sec =	3
Barra 38	No I =	47	No J =	48	Mat =	1	Sec =	3
Barra 39	No I =	48	No J =	49	Mat =	1	Sec =	3
Barra 40	No I =	1	No J =	8	Mat =	1	Sec =	5
Barra 41	No I =	3	No J =	10	Mat =	1	Sec =	5
Barra 42	No I =	5	No J =	12	Mat =	1	Sec =	5
Barra 43	No I =	7	No J =	14	Mat =	1	Sec =	5
Barra 44	No I =	8	No J =	15	Mat =	1	Sec =	5
Barra 45	No I =	10	No J =	18	Mat =	1	Sec =	5
Barra 46	No I =	12	No J =	21	Mat =	1	Sec =	5
Barra 47	No I =	14	No J =	24	Mat =	1	Sec =	5
Barra 48	No I =	15	No J =	25	Mat =	1	Sec =	5
Barra 49	No I =	18	No J =	27	Mat =	1	Sec =	5
Barra 50	No I =	21	No J =	29	Mat =	1	Sec =	5

Barra 51	No I = 24	No J = 31	Mat = 1	Sec = 5
Barra 52	No I = 25	No J = 32	Mat = 1	Sec = 5
Barra 53	No I = 27	No J = 33	Mat = 1	Sec = 5
Barra 54	No I = 29	No J = 34	Mat = 1	Sec = 5
Barra 55	No I = 31	No J = 35	Mat = 1	Sec = 5
Barra 56	No I = 32	No J = 36	Mat = 1	Sec = 5
Barra 57	No I = 33	No J = 38	Mat = 1	Sec = 5
Barra 58	No I = 34	No J = 40	Mat = 1	Sec = 5
Barra 59	No I = 35	No J = 42	Mat = 1	Sec = 5
Barra 60	No I = 36	No J = 43	Mat = 1	Sec = 5
Barra 61	No I = 38	No J = 45	Mat = 1	Sec = 5
Barra 62	No I = 40	No J = 47	Mat = 1	Sec = 5
Barra 63	No I = 42	No J = 49	Mat = 1	Sec = 5

Condicoes Prescritas nos Apoios

No 1	desl X = 0.000	desl Y = 0.000	mola Z = 0.000
No 3	mola X = 0.000	desl Y = 0.000	mola Z = 0.000
No 5	mola X = 0.000	desl Y = 0.000	mola Z = 0.000
No 7	mola X = 0.000	desl Y = 0.000	mola Z = 0.000

Carregamento 1

Cargas Nodais :

No	direcao 1	direcao 2	direcao 3
16	5.000	0.000	0.000
17	5.000	0.000	0.000
19	5.000	0.000	0.000
20	5.000	0.000	0.000
22	5.000	0.000	0.000
23	5.000	0.000	0.000

Variacao da amplitude no tempo

t	f(t)
0.00	0.00
0.10	0.10
0.20	0.20
0.30	0.30
0.40	0.40
0.50	0.50
0.60	0.60
0.70	0.70
0.80	0.80
0.90	0.90
1.00	1.00
1.10	0.90
1.20	0.80
1.30	0.70
1.40	0.60
1.50	0.50
1.60	0.40
1.70	0.30
1.80	0.20
1.90	0.10
2.00	0.00

Numero de vetores de Ritz adotados = 8

Tempo de Processamento

SAP IV = 27.60 s

Dinâmica = 1111.0 s

Relação Dinâmica/SAP IV = 40

Deslocamentos obtidos

t	No	dir	Dinamica	Sap IV
1.00	16	1	1.0093e-07	8.820e-08
1.00	16	2	1.1054e-09	9.755e-10
1.00	17	1	1.0098e-07	8.822e-08
1.00	17	2	1.0306e-10	9.028e-11
1.00	19	1	1.0100e-07	8.819e-08
1.00	19	2	6.0664e-10	5.002e-10
1.00	21	1	1.0099e-07	8.813e-08
1.00	21	2	5.9383e-10	6.622e-10

Neste exemplo temos um espaço vetorial ainda maior que no exemplo anterior e tanto a solução por superposição modal quanto a por vetores de Ritz são aproximadas. Neste caso os resultados foram igualmente próximos.

Podemos concluir que para modelos do porte dos analisados acima, o programa DINA, rodando em um micro-computador de 16 bits, fornece resultados equivalentes aos fornecidos pelo Sap IV no VAX 785, o que indica a boa qualidade do produto gerado pela metodologia utilizada.

VI.2 - Conclusões

As técnicas de análise e projeto utilizadas na criação dos programas de análise estática e dinâmica se mostraram extremamente úteis, não só pela facilidade proporcionada na geração dos programas como também pela facilidade existente para eventuais modificações e acréscimos nos programas resultantes. Os Diagramas de Fluxo de Dados e os Diagramas de Estruturas obtidos servem também como documentação dos sistemas, normalmente inexistente ou precária quando não são adotadas sistemáticas adequadas de projeto e programação.

A linguagem Pascal se mostrou muito adequada para programas científicos, superando o FORTRAN e o BASIC com seus recursos de estruturação de código, estruturação de dados, passagem de parâmetros e alocação dinâmica de memória. O compilador usado, Turbo-Pascal da Borland Inc., se revelou muito eficiente, compilando o código fonte com muita rapidez e gerando um programa objeto bastante rápido.

O programa de análise estática é teoricamente capaz de analisar estruturas reticuladas com até 6.000 deslocamentos nodais em um microcomputador de 16 bits com um mínimo de 256 Kbytes de memória e um disco rígido de 10 MegaBytes. O programa de análise dinâmica analisa estruturas do mesmo porte, necessitando de um mínimo de 512 Kbytes de memória central.

VI.3 - Sugestões

Naturalmente que o trabalho desenvolvido inicialmente nesta tese pode ser melhorado e expandido e para isto gostaria de dar algumas sugestões .

Em relação às metodologias usadas, eu particularmente teria detalhado mais a Especificação Funcional, incluindo mais Diagramas de Fluxo de Dados e talvez incluindo um Dicionário de Dados sucinto. O nível de detalhamento do Projeto me parece adequado para o objetivo deste trabalho, porém não seria suficiente para que um programador desse continuidade ao sistema. Minha principal sugestão em relação às metodologias seria a automatização da Análise e do Projeto por meio de ferramentas de CASE(Computer Aided System Engeneering). Tanto a Análise quanto o Projeto são úteis, porém muito trabalhosos sem o uso de uma ferramenta para auxiliar estas tarefas.

Em relação aos programas gerados, seria interessante submetermos estruturas de maior porte aos programas de forma a analisarmos trechos críticos de código a serem otimizados. Provavelmente entre estes trechos se incluem as rotinas de montagem da matriz de rigidez global, triangularização do sistema de equações e resolução do mesmo sistema. O fato de ter sido usada uma rotina genérica para gerenciamento da memória nestas rotinas se por um lado facilitou enormemente o projeto, programação e depuração destas rotinas, por outro lado indica que uma solução mais específica e sofisticada de gerenciamento muito provavelmente nos trará um melhor desempenho, ainda que o esforço para projetar, programar e depurar tais rotinas aumente consideravelmente.

Outra opção interessante seria reaproveitar os projetos e montar a partir deles programas de análise não linear, instabilidade, etc., já que o coração de tais programas é o mesmo dos programas gerados neste trabalho.

A principal sugestão que posso dar a partir deste trabalho é no sentido de que as pessoas que programam sistemas científicos podem e devem se beneficiar muitíssimo com o uso destas metodologias oriundas da informática. São metodologias simples, facilmente aplicáveis se temos ferramentas adequadas e que trazem um benefício sensível em todas as etapas de desenvolvimento, bem como na qualidade do sistema gerado.

Capítulo VII - Bibliografia

- (1) BATHE, K. J.
"Finite Element Procedures in Engineering Analysis" - Prentice Hall, 1982.
- (2) WILSON, E. L.; YUAN, M. W.; DICKENS, J. M.
 "Dynamic Analysis by Direct Superposition of Ritz Vectors" - Earthquake Engineering and Structural Dynamics, vol 10, 813-821, 1982.
- (3) SORIANO, H. L.
"Análise de Estruturas Reticuladas em Computadores de Pequeno Porte" - Escola de Engenharia da Universidade Federal do Rio de Janeiro, 1983.
- (4) LANDAU, LUIZ
 "Comportamento Dinâmico Não-Linear de Estruturas pelo Método de Superposição Modal" - Tese de Doutorado - COPPE-UFRJ - Rio de Janeiro, 1983
- (5) SORIANO, H. L.; PAULA, R. P.
 "Generalização da Técnica do Número Grande no Método dos Deslocamentos" - Revista Escola de Minas, primeiro trimestre de 1985.
- (6) SORIANO, H. L.
"Sistemas de Equações Algébricas Lineares em Problemas Estruturais" - Laboratório Nacional de Engenharia Civil - Lisboa - 1981

- (7) STEVENS, W. B.
"Projeto Estruturado de Sistemas" - Rio de Janeiro, Editora Campus, 1986.
- (8) GANE, C.
"Análise Estruturada de Sistemas" - Rio de Janeiro, LTC, 1983.
- (9) SCHMITZ, E. A.
"Pascal e Técnicas de Programação" - Rio de Janeiro, LTC, 1985.
- (10) SORIANO, H. L.; CARVALHO, S. R. F.
"Técnicas de Estruturação de Dados com a Linguagem Pascal na Análise Estrutural" - VII Congresso Latino-Americano de Métodos Computacionais em Engenharia, São Carlos, SP, 1986.
- (11) GERE, J. M.; WEAVER, W.
"Análise de Estruturas Reticuladas"
Rio de Janeiro, Editora Guanabara Dois, 1981.
- (12) TURBO PASCAL VERSION 3.0 REFERENCE MANUAL-
Borland International Inc. 1985

Apêndice I - Listagem da Rotina Gráfica

```

procedure DesenheEstrutura;
const
  Incremento = 0.0872664;
  Zoom       = 1.1;
var
  Saia      : Boolean;

procedure CalculeCoordenadasIniciais;
var
  DeltaX,DeltaY,DeltaZ,Escala:Real;

function MaiorDistancia:Real;
var
  MaxX,MaxY,MaxZ,
  MinX,MinY,MinZ :Real;
  indice          :1..MaxNos;
begin
  with No[1]^ do
    begin
      MaxX:=Coord[1];MinX:=Coord[1];
      MaxY:=Coord[2];MinY:=Coord[2];
      MaxZ:=Coord[3];MinZ:=Coord[3];
    end;
  for indice:=2 to Nos do with No[indice]^ do
    begin
      if MinX > Coord[1] then MinX:=Coord[1];
      if MaxX < Coord[1] then MaxX:=Coord[1];
      if MinY > Coord[2] then MinY:=Coord[2];
      if MaxY < Coord[2] then MaxY:=Coord[2];
      if MinZ > Coord[3] then MinZ:=Coord[3];
      if MaxZ < Coord[3] then MaxZ:=Coord[3];
    end;
  DeltaX:=MaxX-MinX;DeltaY:=MaxY-MinY;DeltaZ:=MaxZ-MinZ;
  MaiorDistancia:=Sqrt(DeltaX*DeltaX+
                       DeltaY*DeltaY+
                       DeltaZ*DeltaZ);

end;

procedure CalculeEscalaInicial;
begin
  Escala:=180/MaiorDistancia;
end;

procedure ModifiqueCoordenadasIniciais;
var
  indice          :1..MaxNos;
begin
  for indice:=1 to Nos do with No[indice]^ do
    begin
      Coord[1]:= (Coord[1]-DeltaX/2)*Escala;
      Coord[2]:= -(Coord[2]-DeltaY/2)*Escala;
      Coord[3]:= (Coord[3]-DeltaZ/2)*Escala;
    end;
  end;

begin { Calcule Coordenadas Iniciais }
  CalculeEscalaInicial;
  ModifiqueCoordenadasIniciais;
end;

```

```

procedure PrepareTela;
begin
  HIRes;
  HIResColor(15);
end;

procedure PloteEstrutura;
const
  Ajuste = 2.272727; { escala 1:1 }
  DeltaX = 320;
  DeltaY = 95;
var
  x1,x2,y1,y2:Real;
  No1,No2      :1..MaxNos;
  Barra        :TipoBarra;

begin
  PrepareTela;
  Assign(ArqBarra,drive+' '+Nome+'.004');
  Reset(ArqBarra);
  while not Eof(ArqBarra) do
    begin
      Read(ArqBarra,Barra);
      with Barra do
        begin
          No1:=No[1];
          No2:=No[2];
        end;
      with No[No1]^ do
        begin
          x1:=Coord[1]*ajuste+DeltaX;
          y1:=Coord[2]+DeltaY;
        end;
      with No[No2]^ do
        begin
          x2:=Coord[1]*ajuste+DeltaX;
          y2:=Coord[2]+DeltaY;
        end;
      Draw(Round(x1),Round(y1),Round(x2),Round(y2),15);
    end;
  Close(ArqBarra);
end;

procedure CalculeNovasCoordenadas;
var
  R      :array[1..3,1..3] of Real;
  tecla  :char;

procedure MudeEscala(fator:Real);
var
  indice      :1..MaxNos;
begin
  for indice:=1 to Nos do with No[indice]^ do
    begin
      Coord[1]:= Coord[1]*fator;
      Coord[2]:= Coord[2]*fator;
      Coord[3]:= Coord[3]*fator;
    end;
  end;
end;

```

```

procedure ModifiqueCoordenadas:
var
  indice      : 1..MaxNos;
  linha,coluna: 1..3;
  aux        : array[1..3] of Real;
begin
  for indice:=1 to Nos do with No[indice]^ do
    begin
      for coluna:=1 to 3 do aux[coluna]:=0;
      for linha:=1 to 3 do
        for coluna:=1 to 3 do
          aux[linha]:=aux[linha]+
            R[linha,coluna]*Coord[coluna];
        for linha:=1 to 3 do Coord[linha]:=aux[linha];
      end;
    end;
end;

procedure RotacaoVertical(Alfa:Real);
var
  linha,coluna: 1..3;
begin
  for linha:=1 to 3 do
    for coluna:=1 to 3 do R[linha,coluna]:=0;
  R[1,1]:=1;
  R[2,2]:=Cos(Alfa);
  R[2,3]:=Sin(Alfa);
  R[3,2]:=-R[2,3];
  R[3,3]:= R[2,2];
  ModifiqueCoordenadas;
end;

procedure RotacaoHorizontal(Alfa:Real);
var
  linha,coluna: 1..3;
begin
  for linha:=1 to 3 do
    for coluna:=1 to 3 do R[linha,coluna]:=0;
  R[1,1]:=Cos(Alfa);
  R[1,3]:=Sin(Alfa);
  R[2,2]:=1;
  R[3,1]:=-R[1,3];
  R[3,3]:= R[1,1];
  ModifiqueCoordenadas;
end;

begin { Novas Coordenadas }
  Saia:=False;
  repeat
    Read(Kbd,Tecla);
    If Keypressed then Read(Kbd,Tecla);
  until tecla in [#13,#43,#45,#75,#77,#72,#80];

  case Ord(Tecla) of
    13:Saia:=True;
    75:RotacaoHorizontal(-Incremento); { esquerda }
    77:RotacaoHorizontal( Incremento); { direita }
    72:RotacaoVertical  (-Incremento); { acima }
    80:RotacaoVertical  ( Incremento); { abaixo }
    43:MudeEscala      (    Zoom    );
    45:MudeEscala      (    1/Zoom  );
  end;
end;

```

```
end;

procedure ConfigureModoTexto;
begin
  TextMode;
  TextBackGround(Black);
  ClrScr;
end;

begin          [ Grafico ]
  LeiaNos;
  CalculeCoordenadasIniciais;
  PrepareTela;
  repeat
    PloteEstrutura;
    CalculeNovasCoordenadas;
  until Sala;
  ApagueNos;
  ConfigureModoTexto;
end;
```

Apêndice II - Listagem da Rotina Estática

```

procedure AnaliseEstrutura;
const
  KApoio      = 1E25;
type
  Matriz3     = array [1..3 , 1..3] of Real;
  Vetor12     = array [1..12] of Real;
  Matriz12    = array [1..12,1..12] of Real;
  VetorReal   = array[1..MaxDeslocamentos] of Real;

procedure ObtenhaMatrizDeRotacao(var R:Matriz3;
                                 Barra:TipoBarra);

const
  Tolerancia = 1E-05;
var
  linha,coluna :1..3;
  NoInicial,NoFinal:1..MaxNos;
  Cx,Cy,Cz,L,Aux,
  SenBeta,CosBeta :Real;

begin
  for linha:=1 to 3 do
    for coluna:=1 to 3 do R[linha,coluna]:=0;
  NoInicial:=Barra.No[1];
  NoFinal :=Barra.No[2];
  Cx:=No[NoFinal]^Coord[1]-No[NoInicial]^Coord[1];
  Cy:=No[NoFinal]^Coord[2]-No[NoInicial]^Coord[2];
  Cz:=No[NoFinal]^Coord[3]-No[NoInicial]^Coord[3];
  L:=Sqrt(Cx*Cx+Cy*Cy+Cz*Cz);
  Cx:=Cx/L;
  Cy:=Cy/L;
  Cz:=Cz/L;
  with Barra do
    begin
      SenBeta:=Sin(Beta);
      CosBeta:=Cos(Beta);
    end;
  R[1,1]:=Cx;R[1,2]:=Cy;R[1,3]:=Cz;
  Aux:=Sqrt(Cx*Cx+Cz*Cz);
  if Abs(Aux) < Tolerancia then
    begin
      R[2,1]:=-Cy*CosBeta;R[2,2]:=0;R[2,3]:=SenBeta;
      R[3,1]:= Cy*SenBeta;R[3,2]:=0;R[3,3]:=CosBeta;
    end
  else
    begin
      R[2,1]:=-(Cx*Cy*CosBeta+Cz*SenBeta)/Aux;
      R[2,2]:= CosBeta*Aux;
      R[2,3]:=-(Cy*Cz*CosBeta-Cx*SenBeta)/Aux;
      R[3,1]:=-(Cz*CosBeta+Cx*Cy*SenBeta)/Aux;
      R[3,2]:=-SenBeta*Aux;
      R[3,3]:= (Cx*CosBeta+Cy*Cz*SenBeta)/Aux;
    end;
  end;

procedure LibereMatrizDeRigidez(var Ke:Matriz12;
                                 Barra:TipoBarra);

var
  linha,coluna,direcao>Total:1..12;
  aux :Real;

```

```

begin
  Total:=DeslocamentosPorNo*NosPorElemento;
  with Barra do
    for direcao:=1 to Total do
      if Liberacao[direcao] = '0' then
        begin
          for linha:=1 to Total do
            if linha <> direcao then
              begin
                if Ke[direcao,direcao] = 0 then Halt;
                aux:=Ke[linha,direcao]/Ke[direcao,direcao];
                for coluna:=1 to Total do
                  Ke[linha,coluna]:=Ke[linha,coluna]-
                    aux*Ke[direcao,coluna];
                end;
                for coluna:=1 to Total do Ke[direcao,coluna]:=0;
              end;
            end;
          end;
        end;
      end;
    end;
end;

```

```

procedure ObtenhaKPE(var Ke:Matriz12;Barra:TipoBarra);

```

```

var
  Aux          :array[1..10] of Real;
  TipoMaterial :1..MaxMaterials;
  TipoSecao    :1..MaxSecoes;
  NoInicial, NoFinal :1..MaxNos;
  linha,coluna,indice :1..12;
  E,G,Ax,Ay,Az,lx,ly,
  lz,Fy,Fz,Cx,Cy,Cz,L :Real;

```

```

begin
  for linha:=1 to 12 do
    for coluna:=1 to 12 do Ke[linha,coluna]:=0;
  end;
  with Barra do
    begin
      TipoMaterial:=Material; TipoSecao :=Secao;
      NoInicial :=No[1]; NoFinal :=No[2];
    end;
  E:=Material[TipoMaterial]^E;
  G:=Material[TipoMaterial]^G;
  Ax:=Secao[TipoSecao]^Ax;
  Ay:=Secao[TipoSecao]^Ay;
  Az:=Secao[TipoSecao]^Az;
  lx:=Secao[TipoSecao]^lx;
  ly:=Secao[TipoSecao]^ly;
  lz:=Secao[TipoSecao]^lz;

  Cx:=No[NoFinal]^Coord[1]-No[NoInicial]^Coord[1];
  Cy:=No[NoFinal]^Coord[2]-No[NoInicial]^Coord[2];
  Cz:=No[NoFinal]^Coord[3]-No[NoInicial]^Coord[3];
  L:=Sqrt(Cx*Cx+Cy*Cy+Cz*Cz);

  if (Ax <> 0) and (G <> 0) then
    begin
      Fy:=12*E*lz*Ay/(G*Ax*Ax*L*L);
      Fz:=12*E*lz*Az/(G*Ax*Ax*L*L);
    end
  else
    begin
      Fy:=0;
      Fz:=0;
    end;
  end;
end;

```

```

Aux[1]:=E*Ax/L ;
Aux[2] :=12*E*Iz/((1+Fy)*L*L*L);
Aux[3] :=12*E*Iy/((1+Fz)*L*L*L);
Aux[4] :=G*Ix/L;
Aux[5] :=(4+Fz)*E*Iy/((1+Fz)*L);
Aux[6] :=(4+Fy)*E*Iz/((1+Fy)*L);
Aux[7] :=(2-Fz)*E*Iy/((1+Fz)*L);
Aux[8] :=(2-Fy)*E*Iz/((1+Fy)*L);
Aux[9] :=6*E*Iz/((1+Fy)*L*L) ;
Aux[10]:=6*E*Iy/((1+Fz)*L*L) ;

```

```

Ke[ 1, 1]:= Aux[ 1];
Ke[ 7, 7]:= Aux[ 1];
Ke[ 1, 7]:=-Aux[ 1];
Ke[ 2, 2]:= Aux[ 2];
Ke[ 8, 8]:= Aux[ 2];
Ke[ 2, 8]:=-Aux[ 2];
Ke[ 3, 3]:= Aux[ 3];
Ke[ 9, 9]:= Aux[ 3];
Ke[ 3, 9]:=-Aux[ 3];
Ke[ 4, 4]:= Aux[ 4];
Ke[10,10]:= Aux[ 4];
Ke[ 4,10]:=-Aux[ 4];
Ke[ 5, 5]:= Aux[ 5];
Ke[11,11]:= Aux[ 5];
Ke[ 5,11]:= Aux[ 7];
Ke[ 6, 6]:= Aux[ 6];
Ke[12,12]:= Aux[ 6];
Ke[ 6,12]:= Aux[ 8];
Ke[ 2, 6]:= Aux[ 9];
Ke[ 2,12]:= Aux[ 9];
Ke[ 3, 5]:=-Aux[10];
Ke[ 3,11]:=-Aux[10];
Ke[ 5, 9]:= Aux[10];
Ke[ 6, 8]:=-Aux[ 9];
Ke[ 8,12]:=-Aux[ 9];
Ke[ 9,11]:= Aux[10];
for linha:=2 to 12 do
  for coluna:=1 to linha-1 do
    Ke[linha,coluna]:=Ke[coluna,linha];
end;

```

```

procedure ObtenhaKPP(var Ke:Matriz12;Barra:TipoBarra);

```

```

var
  Aux :array[1..10] of Real;
  TipoMaterial :1..MaxMateriais;
  TipoSecao :1..MaxSecoes;
  NoInicial,NoFinal :1..MaxNos;
  linha,coluna :1..6;
  E,G,Ax,Ay,Iz,Fy,Cx,Cy,L :Real;

```

```

begin
  for linha:=1 to 12 do
    for coluna:=1 to 12 do Ke[linha,coluna]:=0;
  with Barra do
    begin
      TipoMaterial:=Material;
      TipoSecao :=Secao;
      NoInicial :=No[1];

```

```

        NoFinal      :=No[2];
    end;
E:=Material[TipoMaterial]^E;
G:=Material[TipoMaterial]^G;

Ax:=Secao[TipoSecao]^Ax;
Ay:=Secao[TipoSecao]^Ay;
Iz:=Secao[TipoSecao]^Iz;

Cx:=No[NoFinal]^Coord[1]-No[NoInicial]^Coord[1];
Cy:=No[NoFinal]^Coord[2]-No[NoInicial]^Coord[2];
L:=Sqrt(Cx*Cx+Cy*Cy);

if (Ax <> 0) and (G <> 0) then
    Fy:=12*E*Iz*Ay/(G*Ax*Ax*L*L)
else Fy:=0;

Aux[1]:=E*Ax/L;
Aux[2]:=12*E*Iz/((1+Fy)*L*L*L);
Aux[6:=(4+Fy)*E*Iz/((1+Fy)*L);
Aux[8:=(2-Fy)*E*Iz/((1+Fy)*L);
Aux[9]:=6*E*Iz/((1+Fy)*L*L);

Ke[1,1]:= Aux[1]; Ke[2,3]:= Aux[9];
Ke[2,2]:= Aux[2]; Ke[2,5]:=-Aux[2];
Ke[3,3]:= Aux[6]; Ke[2,6]:= Aux[9];
Ke[4,4]:= Aux[1]; Ke[3,5]:=-Aux[9];
Ke[5,5]:= Aux[2]; Ke[3,6]:= Aux[8];
Ke[6,6]:= Aux[6]; Ke[5,6]:=-Aux[9];
Ke[1,4]:=-Aux[1];
for linha:=2 to 12 do
    for coluna:=1 to linha-1 do
        Ke[linha,coluna]:=Ke[coluna,linha];
    end;
end;

procedure ObtenhaKTE(var Ke:Matriz12;Barra:TipoBarra);
var
    TipoMaterial      :1..MaxMateriais;
    TipoSecao         :1..MaxSecoes;
    NoInicial,NoFinal :1..MaxNos;
    linha,coluna,indice :1..3;
    E,G,Ax,Cx,Cy,Cz,L,Aux :Real;

begin
    for linha:=1 to 12 do
        for coluna:=1 to 12 do Ke[linha,coluna]:=0;
        with Barra do
            begin
                TipoMaterial:=Material;
                TipoSecao   :=Secao;
                NoInicial   :=No[1];
                NoFinal     :=No[2];
            end;
        E:=Material[TipoMaterial]^E;
        G:=Material[TipoMaterial]^G;

        Ax:=Secao[TipoSecao]^Ax;

        Cx:=No[NoFinal]^Coord[1]-No[NoInicial]^Coord[1];
        Cy:=No[NoFinal]^Coord[2]-No[NoInicial]^Coord[2];
        Cz:=No[NoFinal]^Coord[3]-No[NoInicial]^Coord[3];

```

```

L:=Sqrt(Cx*Cx+Cy*Cy+Cz*Cz);

Aux:=E*Ax/L;

Ke[ 1, 1]:= Aux;
Ke[ 4, 4]:= Aux;
Ke[ 1, 4]:=-Aux;
for linha:=2 to 12 do
  for coluna:=1 to linha-1 do
    Ke[linha,coluna]:=Ke[coluna,linha];
end;

procedure ObtenhaKTP(var Ke:Matriz12;Barra:TipoBarra);
var
  TipoMaterial      :1..MaxMaterials;
  TipoSecao         :1..MaxSecoes;
  NoInicial,NoFinal :1..MaxNos;
  E,G,Ax,Cx,Cy,L,Aux :Real;
  linha,coluna,indice :1..2;

begin
  for linha:=1 to 12 do
    for coluna:=1 to 12 do Ke[linha,coluna]:=0;
  with Barra do
    begin
      TipoMaterial:=Material;
      TipoSecao   :=Secao;
      NoInicial   :=No[1];
      NoFinal     :=No[2];
    end;
  E:=Material[TipoMaterial]^E;
  G:=Material[TipoMaterial]^G;

  Ax:=Secao[TipoSecao]^Ax;

  Cx:=No[NoFinal]^Coord[1]-No[NoInicial]^Coord[1];
  Cy:=No[NoFinal]^Coord[2]-No[NoInicial]^Coord[2];
  L:=Sqrt(Cx*Cx+Cy*Cy);

  Aux:=E*Ax/L;

  Ke[ 1, 1]:= Aux;
  Ke[ 3, 3]:= Aux;
  Ke[ 1, 3]:=-Aux;
  for linha:=2 to 12 do
    for coluna:=1 to linha-1 do
      Ke[linha,coluna]:=Ke[coluna,linha];
    end;
end;

procedure ObtenhaKG(var Ke:Matriz12;Barra:TipoBarra);
var
  Aux                :array[1..10] of Real;
  TipoMaterial      :1..MaxMaterials;
  TipoSecao         :1..MaxSecoes;
  NoInicial,NoFinal :1..MaxNos;
  linha,coluna      :1..12;
  E,G,Ax,Ay,Ix,Iz,Fy,Cx,Cz,L :Real;

begin
  for linha:=1 to 12 do
    for coluna:=1 to 12 do Ke[linha,coluna]:=0;

```

```

with Barra do
begin
  TipoMaterial:=Material;
  TipoSecao   :=Secao;
  NoInicial   :=No[1];
  NoFinal     :=No[2];
end;
E:=Material[TipoMaterial]^E;
G:=Material[TipoMaterial]^G;

Ax:=Secao[TipoSecao]^Ax;
Ay:=Secao[TipoSecao]^Ay;
Ix:=Secao[TipoSecao]^Ix;
Iz:=Secao[TipoSecao]^Iz;

Cx:=No[NoFinal]^Coord[1]-No[NoInicial]^Coord[1];
Cz:=No[NoFinal]^Coord[3]-No[NoInicial]^Coord[3];
L:=Sqrt(Cx*Cx+Cz*Cz);

if (Ax <> 0) and (G <> 0) then
  Fy:=12*E*Iz*Ay/(G*Ax*Ax*L*L)
else Fy:=0;

Aux[2]:=12*E*Iz/((1+Fy)*L*L*L);
Aux[4]:=G*Ix/L;
Aux[6]:=(4+Fy)*E*Iz/((1+Fy)*L);
Aux[8]:=(2-Fy)*E*Iz/((1+Fy)*L);
Aux[9]:=6*E*Iz/((1+Fy)*L*L);

Ke[1,1]:= Aux[4];Ke[2,2]:= Aux[2];Ke[3,3]:= Aux[6];
Ke[4,4]:= Aux[4];Ke[5,5]:= Aux[2];Ke[6,6]:= Aux[6];
Ke[2,3]:= Aux[9];Ke[2,5]:=-Aux[2];Ke[2,6]:= Aux[9];
Ke[3,6]:= Aux[8];Ke[3,5]:=-Aux[9];
Ke[1,4]:=-Aux[4];Ke[5,6]:=-Aux[9];
for linha:=2 to 12 do
  for coluna:=1 to linha-1 do
    Ke[linha,coluna]:=Ke[coluna,linha];
  end;
end;

procedure MonteVetoresIndependentes;
type
  Vetor3 = array[1..3] of Real;
var
  f          :VetorReal;
  Carregamento :1..MaxCarregamentos;
  NovaCarga   :ApontaCarga;

procedure InicializeVetorIndependente;
var
  direcao:1..MaxDeslocamentos;
begin
  for direcao:=1 to Nos*DeslocamentosPorNo do
    f[direcao]:=0;
  end;
end;

procedure LeiaCarregamento(Numero:Integer);
var
  NovaCarga:ApontaCarga;

procedure LeiaCargaAux;

```

```

begin
  New(NovaCarga);
  Read(ArqCarga,NovaCarga^);
  if PrimeiraCarga = nil
    then PrimeiraCarga :=NovaCarga
    else UltimaCarga^.proxima:=NovaCarga;
  NovaCarga^.proxima:=nil;
  UltimaCarga:=NovaCarga;
end;

begin { LeiaCarregamento}
  ApagueCarregamento;
  Assign(ArqCarga,drive+' '+Nome+'.C'+Complemento(Numero));
  Reset(ArqCarga);
  PrimeiraCarga:=nil;
  while not Eof(ArqCarga) do LeiaCargaAux;
  Close(ArqCarga);
  NovaCarga:=PrimeiraCarga;
end;

procedure ConsidereCargasNasBarras;
var
  Elemento          : Integer;
  Barra             : TipoBarra;
  L                 : Real;
  ForcaLocal       : array[1..12] of Real;
  NoInicial,NoFinal: 1..MaxNos;
  R                 : Matriz3;

procedure CalculeParametrosBarra;
var
  Cx,Cy,Cz:Real;
begin
  NoInicial:=Barra.No[1];
  NoFinal  :=Barra.No[2];
  Cx:=No[NoInicial]^Coord[1]-No[NoFinal]^Coord[1];
  Cy:=No[NoInicial]^Coord[2]-No[NoFinal]^Coord[2];
  Cz:=No[NoInicial]^Coord[3]-No[NoFinal]^Coord[3];
  L:=Sqrt(Cx*Cx+Cy*Cy+Cz*Cz);
  ObtenhaMatrizDeRotacao(R,Barra);
end;

procedure ProcesseCargas;
var
  direcao:1..12;
  parcela:Vetor3;

  procedure ConsidereSituacaoDaCarga;

  procedure TransformeEmLocal;
  var
    Aux:Vetor3;
  begin
    Aux:=parcela;
    parcela[1]:=R[1,1]*Aux[1]+
                R[1,2]*Aux[2]+R[1,3]*Aux[3];
    parcela[2]:=R[2,1]*Aux[1]+
                R[2,2]*Aux[2]+R[2,3]*Aux[3];
    parcela[3]:=R[3,1]*Aux[1]+
                R[3,2]*Aux[2]+R[3,3]*Aux[3];
  end;

```

```

begin [ ConsidereSituacaoDaCarga ]
  parcela[1]:=0;
  parcela[2]:=0;
  parcela[3]:=0;
  case NovaCarga^.Situacao of
    LocalXY: parcela[2]:=1;
    LocalXZ: parcela[3]:=1;
    GlobalX: begin
      parcela[1]:=1;
      TransformeEmLocal;
    end;
    GlobalY: begin
      parcela[2]:=1;
      TransformeEmLocal;
    end;
    GlobalZ: begin
      parcela[3]:=1;
      TransformeEmLocal;
    end;
  end;
end;

procedure ConsidereCargaUniforme;
var
  q,qx,qy,qz,a1,a2:Real;
begin
  q:=NovaCarga^.CargaUnif;
  ConsidereSituacaoDaCarga;
  qx:=q*parcela[1];
  qy:=q*parcela[2];
  qz:=q*parcela[3];
  a1:=L/2;
  a2:=L*L/12;
  ForcaLocal[ 1]:= ForcaLocal[ 1]-qx*a1;
  ForcaLocal[ 7]:= ForcaLocal[ 7]-qx*a1;
  ForcaLocal[ 2]:= ForcaLocal[ 2]-qy*a1;
  ForcaLocal[ 8]:= ForcaLocal[ 8]-qy*a1;
  ForcaLocal[ 3]:= ForcaLocal[ 3]-qz*a1;
  ForcaLocal[ 9]:= ForcaLocal[ 9]-qz*a1;
  ForcaLocal[ 5]:= ForcaLocal[ 5]+qz*a2;
  ForcaLocal[11]:= ForcaLocal[11]-qz*a2;
  ForcaLocal[ 6]:= ForcaLocal[ 6]-qy*a2;
  ForcaLocal[12]:= ForcaLocal[12]+qy*a2;
  with Controle do
    if TipoEstrutura in [TrellicaEspacial,TrellicaPlana]
    then begin
      ForcaLocal[ 5]:= 0;ForcaLocal[11]:= 0;
      ForcaLocal[ 6]:= 0;ForcaLocal[12]:= 0;
    end
  end;
end;

procedure ConsidereCargaConcentrada;
var
  q,qx,qy,qz,a,b:Real;
begin
  with NovaCarga^ do
    begin
      q:=CargaConc;
      a:=DistConc;
    end;
end;

```

```

b:=L-a;
ConsidereSituacaoDaCarga;
qx:=q*parcela[1];
qy:=q*parcela[2];
qz:=q*parcela[3];
if Controle.TipoEstrutura in
  [TrelicaEspacial,TrelicaPlana] then
  begin
    ForcaLocal[ 1]:= ForcaLocal[ 1]-qx*b/L;
    ForcaLocal[ 7]:= ForcaLocal[ 7]-qx*a/L;
    ForcaLocal[ 2]:= ForcaLocal[ 2]-qy*b/L;
    ForcaLocal[ 8]:= ForcaLocal[ 8]-qy*a/L;
    ForcaLocal[ 3]:= ForcaLocal[ 3]-qz*b/L;
    ForcaLocal[ 9]:= ForcaLocal[ 9]-qz*a/L;
    ForcaLocal[ 5]:= 0;
    ForcaLocal[11]:= 0;
    ForcaLocal[ 6]:= 0;
    ForcaLocal[12]:= 0;
  end
else
  begin
    ForcaLocal[ 1]:= ForcaLocal[ 1]-qx*b/L;
    ForcaLocal[ 2]:= ForcaLocal[ 2]-
      qy*b*b*(2*a+L)/(L*L*L);
    ForcaLocal[ 3]:= ForcaLocal[ 3]-
      qz*b*b*(2*a+L)/(L*L*L);
    ForcaLocal[ 5]:= ForcaLocal[ 5]+qz*a*b*b/(L*L);
    ForcaLocal[ 6]:= ForcaLocal[ 6]-qy*a*b*b/(L*L);
    ForcaLocal[ 7]:= ForcaLocal[ 7]-qx*a/L;
    ForcaLocal[ 8]:= ForcaLocal[ 8]-q
      y*(1-b*b*(2*a+L)/(L*L*L));
    ForcaLocal[ 9]:= ForcaLocal[ 9]-
      qz*(1-b*b*(2*a+L)/(L*L*L));
    ForcaLocal[11]:= ForcaLocal[11]-qz*a*a*b/(L*L);
    ForcaLocal[12]:= ForcaLocal[12]+qy*a*a*b/(L*L);
  end;
end;

procedure ConsidereCargaUniformeParcial;
var
  q,qx,qy,qz,a,b,c,a1,a2,a3:Real;
begin
  with NovaCarga^ do
    begin
      q:=CargaParc;
      c:=comp;
      a:=DistParc+c/2;
      b:=L-a;
    end;
  ConsidereSituacaoDaCarga;
  qx:=q*parcela[1];
  qy:=q*parcela[2];
  qz:=q*parcela[3];
  if Controle.TipoEstrutura in
    [TrelicaEspacial,TrelicaPlana] then
    begin
      a1:=b*c/L;
      a2:=0;
      a3:=0;
    end
  else

```

```

begin
  a1:=b*c/L+c*(4*(a*b*b-a*a*b)+c*c*(a-b))/(4*L*L*L);
  a2:=c*(12*a*b*b+c*c*(L-3*b))/(12*L*L);
  a3:=c*(12*a*a*b+c*c*(L-3*a))/(12*L*L);
  end;
ForcaLocal[ 1]:= ForcaLocal[ 1]-qx*c+qx*a*c/L;
ForcaLocal[ 7]:= ForcaLocal[ 7]-qx*a*c/L ;
ForcaLocal[ 2]:= ForcaLocal[ 2]-qy*a1 ;
ForcaLocal[ 8]:= ForcaLocal[ 8]-qy*(c-a1);
ForcaLocal[ 3]:= ForcaLocal[ 3]-qz*a1 ;
ForcaLocal[ 9]:= ForcaLocal[ 9]-qz*(c-a1);
ForcaLocal[ 5]:= ForcaLocal[ 5]+qz*a2 ;
ForcaLocal[11]:= ForcaLocal[11]-qz*a3 ;
ForcaLocal[ 6]:= ForcaLocal[ 6]-qy*a2 ;
ForcaLocal[12]:= ForcaLocal[12]+qy*a3 ;
end;

procedure ConsiderereCargaTrapezoidal;
var
  q1,q2,qx1,qx2,qy1,qy2,qz1,qz2,a,b,c,
  a1,a2,a3,a4,a5,a6,a7,a8,a9,a10 :Real;
begin
  with NovaCarga^ do
    begin
      q1:=CargaEsq;
      q2:=CargaDir-CargaEsq;
      a:=DistTrap;
      c:=CompTrap;
      b:=L-a-c;
    end;
  ConsiderereSituacaoDaCarga;
  qx1:=q1*parcela[1];qx2:=q2*parcela[1];
  qy1:=q1*parcela[2];qy2:=q2*parcela[2];
  qz1:=q1*parcela[3];qz2:=q2*parcela[3];

  if Controle.TipoEstrutura in
    [TrelicaEspacial,TrelicaPlana] then
    begin
      a1 := c-c*(a+c/2)/L;
      a2 := c*(b+c/2)/L;
      a3 := 0;
      a4 := 0;
      a7 := 0;
      a8 := 0;
      a5 := c*(0.5+(a/2+c/3)/L);
      a6 := c*(b+c/3)/(2*L);
    end
  else
    begin
      a1 := c-c*(a+c/2)/L;
      a2 := c*(b+c/2)/L+3*c*c*c*(a-b)/(12*L*L*L);
      a3 := c*(12*(a+c/2)*(b+c/2)*(b+c/2)+
        c*c*(L-3*(b+c/2)))/(12*L*L);
      a4 := c*(12*(a+c/2)*(a+c/2)*(b+c/2)+
        c*c*(L-3*(a+c/2)))/(12*L*L);
      a7 := c*(10*b*b*(3*a+2*c)+c*c*(10*b+ 5*a+2*c)+
        20*a*b*c)/(60*L*L);
      a8 := c*(10*a*a*(3*b+ c)+c*c*(10*a+15*b+3*c)+
        40*a*b*c)/(60*L*L);
      a5 := c*(0.5+(a/2+c/3)/L);
      a6 := c*(b+c/3)/(2*L)+(a7-a8)/L;
    end;
  end;
end;

```

```

end;

ForcaLocal[ 1]:= ForcaLocal[ 1]-qx1*a1-qx2*a5;
ForcaLocal[ 2]:= ForcaLocal[ 2]-qy1*a2-qy2*a6;
ForcaLocal[ 3]:= ForcaLocal[ 3]-qz1*a2-qz2*a6;
ForcaLocal[ 5]:= ForcaLocal[ 5]+qz1*a3+qz2*a7;
ForcaLocal[ 6]:= ForcaLocal[ 6]-qy1*a3-qy2*a7;
ForcaLocal[ 7]:= ForcaLocal[ 7]-(2*qx1+qx2)*c/2+
qx1*a1+qx2*a5;
ForcaLocal[ 8]:= ForcaLocal[ 8]-(2*qy1+qy2)*c/2+
qy1*a2+qy2*a6;
ForcaLocal[ 9]:= ForcaLocal[ 9]-(2*qz1+qz2)*c/2+
qz1*a2+qz2*a6;
ForcaLocal[11]:= ForcaLocal[11]-qz1*a4-qz2*a8;
ForcaLocal[12]:= ForcaLocal[12]+qy1*a4+qy2*a8;
end;

procedure ConsiderereVariacaoDeTemperatura:
var
  E,Ax,Forca:Real;
begin
  LeiaMaterials;
  LeiaSecoes;
  E      :=Material[Barra.Material]^E;
  Ax     :=Secao [Barra.Secao ]^Ax;
  Forca :=NovaCarga^.Alfa1*E*Ax*NovaCarga^.DeltaT;
  ForcaLocal[1]:= ForcaLocal[1]+Forca;
  ForcaLocal[7]:= ForcaLocal[7]-Forca;
end;

procedure ConsiderereVariacaoDiferencialDeTemperatura:
var
  E,Iy,Iz,DeltaT,altura,DeltaTxy,DeltaTxz,Alfa:Real;
begin
  LeiaMaterials;
  LeiaSecoes;
  E:= Material[Barra.Material]^E;
  Iy:=Secao [Barra.Secao ]^Iy;
  Iz:=Secao [Barra.Secao ]^Iz;
  with NovaCarga^ do
    begin
      DeltaT:=t2-t1;
      altura:=h;
      Alfa :=Alfa2;
    end;
  ConsidereSituacaoDaCarga;
  DeltaTxy:=DeltaT*parcela[2];
  DeltaTxz:=DeltaT*parcela[3];
  ForcaLocal[ 5]:=ForcaLocal[ 5]+
Alfa*E*Iy*DeltaTxz/Altura;
  ForcaLocal[ 6]:=ForcaLocal[ 6]-
Alfa*E*Iz*DeltaTxy/Altura;
  ForcaLocal[11]:=ForcaLocal[11]-
Alfa*E*Iy*DeltaTxz/Altura;
  ForcaLocal[12]:=ForcaLocal[12]+
Alfa*E*Iz*DeltaTxy/Altura;
end;

begin { Processe Cargas }
  for direcao:=1 to 12 do ForcaLocal[direcao]:=0;
  NovaCarga:=PrimeiraCarga;

```

```

while NovaCarga <> nil do
  with NovaCarga^ do
    begin
      if (Elemento = Barra) and (TipoCarga <> 7) then
        case TipoCarga of
          1:ConsiderereCargaUniforme;
          2:ConsiderereCargaConcentrada;
          3:ConsiderereCargaUniformeParcial;
          4:ConsiderereCargaTrapezoidal;
          5:ConsiderereVariacaoDeTemperatura;
          6:ConsiderereVariacaoDiferencialDeTemperatura;
        end;
        NovaCarga:=proxima;
      end;
    end;
end;

procedure ConsiderereLiberacoes;
var
  linha,coluna,direcao>Total:1..12;
  aux                    :Real;
  Ke                     :Matriz12;
  libere                 :Boolean;

begin
  libere:=false;
  Total:=DeslocamentosPorNo*NosPorElemento;
  for linha:=1 to total do
    libere:=(libere or (ForcaLocal[linha] <> 0));
  if libere then
    begin
      LeiaMaterials;
      LeiaSecoes;
      case Controle.TipoEstrutura of
        Grelha           :ObtenhaKG (Ke,Barra);
        PorticoEspacial :ObtenhaKPE(Ke,Barra);
        PorticoPlano     :ObtenhaKPP(Ke,Barra);
        TrelicaEspacial :ObtenhaKTE(Ke,Barra);
        TrelicaPlana     :ObtenhaKTP(Ke,Barra);
      end;

      with Barra do
        for direcao:=1 to Total do
          if Liberacao[direcao] = '0' then
            begin
              for linha:=1 to Total do
                if ((linha <> direcao) and
                    not((Liberacao[linha] = '0') and
                        (linha < direcao))) then
                  begin
                    aux:=Ke[linha,direcao]/
                        Ke[direcao,direcao];
                    for coluna:=1 to total do
                      Ke[linha,coluna]:=
                        Ke[linha,coluna]-
                        aux*Ke[direcao,coluna];
                    ForcaLocal[linha]:=
                      ForcaLocal[linha]-
                      aux*ForcaLocal[direcao];
                  end;
                ForcaLocal[direcao]:=0;
              end;
            end;
          end;
        end;
      end;
    end;
  end;
end;

```

```

        end;
    end;

procedure GraveForcasEquivalentesNodais;
var
    direcao      :1..12;
    EsforcoExtr:TipoEsforcoExtr;
begin
    with EsforcoExtr do
        for direcao:=1 to 12 do
            Valor[direcao]:=-ForcaLocal[direcao];
            Write(ArqEsforcoExtr,EsforcoExtr);
        end;
    end;

procedure AtualizeVetorIndependente;
var
    direcao:1..12;
    i,j,k   :1..4;
    Fg      :Vetor12;

begin
    for direcao:=1 to 12 do Fg[direcao]:=0;
    for k:=1 to 4 do
        for i:=1 to 3 do
            for j:=1 to 3 do
                Fg[3*(k-1)+i]:=Fg[3*(k-1)+i]+
                    R(i,j)*ForcaLocal[3*(k-1)+j];

            case Controle.TipoEstrutura of
                PorticoEspacial:begin end;
                PorticoPlano    :begin
                    Fg[3]:=Fg[ 6];Fg[4]:=Fg[ 7];
                    Fg[5]:=Fg[ 8];Fg[6]:=Fg[12];
                end;
                TrelicaEspacial:begin
                    Fg[4]:=Fg[ 7];
                    Fg[5]:=Fg[ 8];
                    Fg[6]:=Fg[ 9];
                end;
                TrelicaPlana    :begin
                    Fg[3]:=Fg[ 7];
                    Fg[4]:=Fg[ 8];
                end;
                Grelha          :begin
                    Fg[1]:=Fg[ 4];Fg[3]:=Fg[ 6];
                    Fg[5]:=Fg[ 8];Fg[4]:=Fg[10];
                    Fg[6]:=Fg[12];
                end;
            end;
        end;
    end;

    for direcao:=1 to DeslocamentosPorNo do
        begin
            f[(NoInicial-1)*DeslocamentosPorNo+direcao]:=
                f[(NoInicial-1)*DeslocamentosPorNo+direcao]-
                Fg[direcao];
            f[(NoFinal-1)*DeslocamentosPorNo+direcao]:=
                f[(NoFinal-1)*DeslocamentosPorNo+direcao]-
                Fg[direcao+DeslocamentosPorNo];
        end;
    end;
end;

```

```

begin { Considere Cargas Nas Barras }
  Assign(ArqEsforcoExtr,drive+':'+Nome+'.N'+
        complemento(carregamento));
  Rewrite(ArqEsforcoExtr);

  Assign(ArqBarra,drive+':'+Nome+'.004');
  Reset(ArqBarra);
  Elemento:=0;
  while not EOF(ArqBarra) do
    begin
      Elemento:=Elemento+1;
      Read(ArqBarra,Barra);
      CalculeParametrosBarra;
      ProcesseCargas;
      if Barra.TemLiberacao then ConsidereLiberacoes;
      GraveForcasEquivalentesNodais;
      AtualizeVetorIndependente;
    end;
  Close(ArqBarra);
  Close(ArqEsforcoExtr);
end;

procedure ConsidereCargasNodais;

procedure ConsidereCargaNodal;
  var
    direcao:1..6;
    pos      :Integer;
  begin
    with NovaCarga^ do
      begin
        pos:=(barra-1)*DeslocamentosporNo;
        for direcao:=1 to DeslocamentosPorNo do
          f[pos+direcao]:=f[pos+direcao]+Forca[direcao];
        end;
      end;
  end;

begin { Considere Cargas Nodais }
  NovaCarga:=PrimeiraCarga;
  while NovaCarga <> nil do
    begin
      if NovaCarga^.TipoCarga = 7 then ConsidereCargaNodal;
      NovaCarga:=NovaCarga^.proxima;
    end;
  end;

procedure ConsidereApoios;
  var
    Numero,Direcao:Integer;
    DirecaoNoNo    :1..6;
    Apoio          :TipoApoio;
  begin
    Numero:=0;
    Assign(ArqApoio,drive+':'+Nome+'.005');
    Reset (ArqApoio);
    while not EOF(ArqApoio) do
      begin
        Read(ArqApoio,Apoio);
        Numero:=Numero+1;
        direcao:=(Apoio.NumeroNo-1)*DeslocamentosporNo;
        for direcaoNoNo:=1 to DeslocamentosporNo do

```

```

begin
  direcao:=direcao+1;
  if (Apoio.Codigo[direcaoNo] = '1') then
    f[direcao]:=f[direcao]+
      KApoio*Apoio.Condicao[direcaoNo];
  end;
end;
Close(ArqApoio);
end;

procedure GraveVetorIndependente;
var
  contador:integer;
  ArqR      :file of Real;
begin
  Assign(ArqR,drive+' '+Nome+'.F'+
    complemento(Carregamento));
  Rewrite(ArqR);
  for contador:=1 to Nos*DeslocamentosPorNo do
    Write(ArqR,f[contador]);
  Close(ArqR);
end;

begin { Monte Vetores Independentes }
  LeiaNos;
  for Carregamento:=1 to Carregamentos do
    begin
      InicializeVetorIndependente;
      LeiaCarregamento(Carregamento);
      ConsidereCargasNasBarras;
      ConsidereCargasNodais;
      ConsidereApoios;
      GraveVetorIndependente;
      ApagueCarregamento;
    end;
  ApagueNos;
end;

procedure CalculeDeslocamentos;
const
  MaxBlocos      = 500;
type
  HoraString     = String[8];
  ApontaVetor   = ^VetorReal;
  VetorPonteiros = array[1..MaxBlocos] of ApontaVetor;
  VetorEntrada   = array[1..MaxBlocos] of HoraString;
var
  Apontador      : VetorReal;
  Bloco          : VetorPonteiros;
  Entrada        : VetorEntrada;
  ArqBlocos      : file of VetorReal;
  BlocosNecessarios : 0..MaxBlocos;
  Carregamento  : 1..MaxCarregamentos;

function Hora:HoraString;
type
  regpack        = record
    ax,bx,cx,dx,di,
    si,ds,es,flags:integer;
  end;
var

```

```

recpack          : regpack;
ah,al,ch,cl,dh  : Byte;
hour,min,seg    : String[2];

```

```

begin
  ah:=$2c;
  with recpack do
    begin
      ax:=ah shl 8 + al;
    end;
  intr($21,recpack);
  with recpack do
    begin
      str(cx shr 8:2,hour);
      str(cx mod 256:2,min);
      str(dx shr 8:2,seg);
    end;
    if min[1] = ' ' then min[1]:='0';
    if seg[1] = ' ' then seg[1]:='0';
    Hora:=hour+':'+min+':'+seg;
  end;

```

```
{.PA}
```

```
procedure MonteVetorApontador;
```

```
var
```

```

Indice,TotalDeslocamentos,Elemento,
MaiorDirecao,MenorDirecao,Dif      : Integer;
NoInicial,NoFinal,MenorNo,MaiorNo  : 1..MaxNos;
NodoElemento,direcao              : 1..6;
Barra                              : TipoBarra;

```

```
begin { Monte Vetor Apontador }
```

```

TotalDeslocamentos:=Nos*DeslocamentosPorNo;
for Indice:=1 to TotalDeslocamentos do
  Apontador[Indice]:=0;

```

```

Assign(ArqBarra,drive+' '+Nome+'.004');
Reset(ArqBarra);

```

```
Elemento:=0;
```

```
while not EOF(ArqBarra) do
```

```
begin
```

```

  Elemento:=Elemento+1;
  Read(ArqBarra,Barra);
  NoInicial:=Barra.No[1];
  NoFinal :=Barra.No[2];
  if NoInicial < NoFinal then MenorNo:=NoInicial
  else MenorNo:=NoFinal;

```

```
MenorDirecao:=(MenorNo-1)*DeslocamentosporNo+1;
```

```
for NodoElemento:=1 to NosporElemento do
  for direcao:=1 to DeslocamentosporNo do
```

```
begin
```

```

  MaiorDirecao:=(Barra.No[NodoElemento]-1)*
  DeslocamentosporNo+direcao;
  Dif:=MaiorDirecao-MenorDirecao+1;
  if Apontador[MaiorDirecao] < Dif then
    Apontador[MaiorDirecao]:=Dif;

```

```
end;
```

```
end;
```

```
Close(ArqBarra);
```

```
for Indice:=2 to TotalDeslocamentos do
```

```

                Apontador[Indice]:=Apontador[Indice]+
                    Apontador[Indice-1];
    end;
{.PA}
procedure SalveBlocos;
var
    indice:1..MaxBlocos;
begin
    Assign (ArqBlocos,drive+' '+Nome+'.007');
    Reset(ArqBlocos);
    for indice:=1 to BlocosNecessarios do
        if Bloco[indice] <> nil then
            begin
                Seek(ArqBlocos,indice-1);
                Write(ArqBlocos,Bloco[indice]^);
            end;
    Close(ArqBlocos);
end;

procedure DesaloqueBlocos;
var
    indice:1..MaxBlocos;
begin
    for indice:=1 to BlocosNecessarios do
        if Bloco[indice] <> nil then
            begin
                Dispose(Bloco[indice]);
                Bloco[indice]:=nil;
            end;
    end;

end;

procedure PrepareBlocos;
var
    indice                :1..MaxBlocos;
    deslocamentos,direcao :1..MaxDeslocamentos;
    TermosDaMatriz        :Real;
    Aux                   :ApontaVetor;
begin
    deslocamentos:=Nos*DeslocamentosPorNo;
    TermosDaMatriz:=Apontador[deslocamentos];
    BlocosNecessarios:=Trunc((TermosDaMatriz-1)/
        MaxDeslocamentos)+1;

    for indice:=1 to MaxBlocos do
        begin
            Bloco[indice]:=nil;
            Entrada[indice]:=Hora;
        end;

    New(Aux);
    for direcao:=1 to MaxDeslocamentos do Aux^[direcao]:=0;

    Assign(ArqBlocos,drive+' '+Nome+'.007');
    Rewrite(ArqBlocos);
    for indice:=1 to BlocosNecessarios do
        Write(ArqBlocos,Aux^);
    Close(ArqBlocos);

    Dispose(Aux);
end;
{.PA}

```

```

procedure SalveBloco(n:integer);
begin
  Assign(ArqBlocos,drive+' '+Nome+'.007');
  Reset(ArqBlocos);
  Seek(ArqBlocos,n-1);
  Write(ArqBlocos,Bloco[n]^);
  Close(ArqBlocos);
end;

function Memoria:Real;
begin
  if MemAvail > 0 then Memoria:=MemAvail*16.0
  else Memoria:=(MemAvail*16.0+1048576.0);
end;

procedure ElimineUmBloco;
var
  MaisAntigo:1..MaxBlocos;

  procedure EscolhaBloco;
  var
    indice:1..MaxBlocos;
    aux :HoraString;
  begin { EscolhaBloco }
    MaisAntigo:=1;
    aux:='99:99:99';
    for indice:=1 to BlocosNecessarios do
      if (Bloco[indice] <> nil) and
        (Entrada[indice] < aux) then
        begin
          aux :=Entrada[indice];
          MaisAntigo:=indice;
        end;
    end;
  end;

begin { ElimineUmBloco }
  EscolhaBloco;
  SalveBloco(MaisAntigo);
  Dispose(Bloco[MaisAntigo]);
  Bloco[MaisAntigo]:=nil;
end;

procedure ChameBloco(n:integer);

  procedure LeiaBloco;
  begin { LeiaBloco }
    New(Bloco[n]);
    Entrada[n]:=Hora;
    Assign(ArqBlocos,drive+' '+Nome+'.007');
    Reset(ArqBlocos);
    Seek(ArqBlocos,n-1);
    Read(ArqBlocos,Bloco[n]^);
    Close(ArqBlocos);
  end;

begin { ChameBloco }
  if ((Memoria - 5000) < (SizeOf(VetorReal))) then
    ElimineUmBloco;
  LeiaBloco;
end;

```

```

function Pegue(QualBloco, indice: Integer): Real;
begin
  if Bloco[QualBloco] = nil then ChameBloco(QualBloco);
  Pegue := Bloco[QualBloco]^(indice);
end;

procedure Ponha(QualBloco, indice: Integer; conteudo: Real);
begin
  if Bloco[QualBloco] = nil then ChameBloco(QualBloco);
  Bloco[QualBloco]^(indice) := conteudo;
end;

procedure MonteMatrizDeRigidez;
var
  Barra           : TipoBarra;
  indice          : array[1..12, 1..12] of Integer;
  correspondencia : array[1..12] of Integer;
  Elemento       : Integer;
  Deslocamentos  : 1..MaxDeslocamentos;
  Bloco          : 1..MaxBlocos;
  considere      : Boolean;

procedure MonteCorrespondencia;
var
  DirecaoElemento : 0..12;
  NodoElemento    : 1..NosPorElemento;
  DirecaoNo       : 1..6;
  DirecaoAnterior : Integer;
begin
  with Barra do
    begin
      DirecaoElemento := 0;
      for NodoElemento := 1 to NosPorElemento do
        begin
          DirecaoAnterior := (No[NodoElemento]-1)*
                               DeslocamentosporNo;
          for DirecaoNo := 1 to DeslocamentosporNo do
            begin
              DirecaoElemento := DirecaoElemento+1;
              Correspondencia[DirecaoElemento] :=
                DirecaoAnterior+DirecaoNo;
            end;
          end;
        end;
      end;
    end;

[ .PA ]
procedure CalculeIndices;
var
  linha, coluna : 0..12;
  aux           : Real;
begin
  considere := false;
  for linha := 1 to Deslocamentos do
    for coluna := 1 to Deslocamentos do
      if not (Correspondencia[linha] >
              Correspondencia[coluna]) then
        begin
          aux := Correspondencia[linha]-
                 Correspondencia[coluna]+
                 Apontador[Correspondencia[coluna]]-
                 1.0*(Bloco-1)*MaxDeslocamentos;

```

```

        if Abs(aux) > MaxDeslocamentos then aux:=0.
        indice[linha,coluna]:=Trunc(aux);
        considere:=considerere or (aux > 0);
    end;
end;

```

```

procedure ConsiderereBarra;

```

```

var
    Ke           :Matriz12;
    linha,coluna :0..12;

```

[.PA]

```

procedure MudeParaReferencialGlobal;

```

```

var
    i,j,k,l,m,n:1..12;
    R,S         :Matriz3;
    Aux         :Real;
    MaxM,Maxl:1..4;
begin
    case Controle.TipoEstrutura of
        PorticoEspacial:begin
            MaxM:=4;
            Maxl:=3;
        end;
        PorticoPlano   :begin
            MaxM:=2;
            Maxl:=3;
        end;
        TrelicaEspacial:begin
            MaxM:=2;
            Maxl:=3;
        end;
        TrelicaPlana   :begin
            MaxM:=2;
            Maxl:=2;
        end;
        Grelha         :begin
            MaxM:=2;
            Maxl:=3;
        end;
    end;
end;

```

```

end;

```

```

ObtenhaMatrizDeRotacao(R,Barra).

```

```

for m:=1 to MaxM do
    for n:=m to MaxM do
        begin
            for i:=1 to Maxl do
                for j:=1 to Maxl do S[i,j]:=0;
            for i:=1 to Maxl do
                for j:=1 to Maxl do
                    begin
                        Aux:=0;
                        for k:=1 to Maxl do
                            for l:=1 to Maxl do
                                Aux:=Aux+R[i,i]*R[k,j]*
                                Ke[Maxl*(m-1)+i,Maxl*(n-1)+k];
                            S[i,j]:=Aux;
                        end;
                    for i:=1 to Maxl do
                        for j:=1 to Maxl do
                            Ke[Maxl*(m-1)+i,Maxl*(n-1)+j]:=S[i,j];
                        end;
                    end;
                for i:=1 to 12 do

```

```

        for j:=i+1 to 12 do Ke(j,i):=Ke(i,j);
    end;
{.PA}
begin { Considere Barra }
    case Controle.TipoEstrutura of
        Greiha      :ObtenhaKG (Ke,Barra);
        PorticoEspacial:ObtenhaKPE(Ke,Barra);
        PorticoPlano  :ObtenhaKPP(Ke,Barra);
        TrelicaEspacial:ObtenhaKTE(Ke,Barra);
        TrelicaPlana  :ObtenhaKTP(Ke,Barra);
    end;
    if Barra.TemLiberacao then
        LibereMatrizDeRigidez(Ke,Barra);
        MudeParaReferencialGlobal;

        for linha:=1 to Deslocamentos do
            for coluna:=1 to Deslocamentos do
                if not (Correspondencia[linha] >
                    Correspondencia[coluna]) then
                    if indice[linha,coluna] > 0 then
                        Ponha(Bloco,indice[linha,coluna],Pegue(Bloco,
                            indice[linha,coluna])+Ke[linha,coluna]);
                    end;
            end;
        end;

begin { Monte Matriz De Rigidez }
    LeiaMaterials;
    LeiaSecoes;
    LeiaNos;
    Deslocamentos:=NosporElemento*DeslocamentosporNo;
    for Bloco:=1 to BlocosNecessarios do
        begin
            Assign(ArqBarra,drive+' '+Nome+'.004');
            Reset(ArqBarra);
            Elemento:=0;
            while not EOF(ArqBarra) do
                begin
                    Elemento:=Elemento+1;
                    Read(ArqBarra,Barra);
                    MonteCorrespondencia;
                    CalculeIndices;
                    if Considere then ConsidereBarra;
                end;
            Close(ArqBarra);
        end;
    ApagueNos;
    ApagueSecoes;
    ApagueMaterials;
end;
{.PA}
procedure IntroduzaApoios;
var
    Apoio      :TipoApoio;
    Numero,Direcao:integer;
    DirecaoNo  :1..6;
    Posicao     :1..MaxDeslocamentos;
    Bloco      :1..MaxBlocos;
    aux       :Real;

begin
    Assign(ArqApoio,drive+' '+Nome+'.005');
    Reset(ArqApoio);

```

```

Numero:=0;
while not EOF(ArqApoio) do
  with Apoio do
    begin
      Read(ArqApoio,Apoio);
      Numero:=Numero+1;
      direcao:=(NumeroNo-1)*DeslocamentosporNo;
      for direcaoNo:=1 to DeslocamentosporNo do
        begin
          direcao:=direcao+1;
          Bloco :=Trunc((Apontador[direcao]-1)/
                        MaxDeslocamentos)+1;
          posicao:=Trunc(Apontador[direcao]-
                        1.0*(Bloco-1)*MaxDeslocamentos);
          aux:=Pegue(Bloco,posicao);
          ifCodigo[direcaoNo] = '1'
            then Ponha(Bloco,posicao,aux+KApoio)
            else Ponha(Bloco,posicao,aux+
                        Condicao[direcaoNo])
        end;
      end;
    Glose(ArqApoio);
  end;
end;

```

```

procedure TriangularizeMatrizDeRigidez;

```

```

const

```

```

  Tolerancia = 1E-10;

```

```

var

```

```

  termo : Real;

```

```

  BlocoTermo, BlocoLinha : 0..MaxBlocos;

```

```

  IndiceTermo, IndiceLinha : Integer;

```

```

  (linha, coluna);

```

```

  AlturaEfetiva, deslocamentos : 1..MaxDeslocamentos;

```

```

{.PA}

```

```

procedure ProcesseColuna;

```

```

var

```

```

  BlocoI, BlocoJ : 1..MaxBlocos;

```

```

  CoeficienteI, CoeficienteJ, Aux : Real;

```

```

  IndiceI, IndiceJ : Integer;

```

```

  LinhaI, AlturaEfetivaI,

```

```

  Linha, MaiorLinha : 0..MaxDeslocamentos;

```

```

begin

```

```

  Ponha(BlocoTermo, IndiceTermo,
        Pegue(BlocoTermo, indiceTermo)/
        Pegue(BlocoLinhaI, IndiceLinhaI));

```

```

  for LinhaI:=LinhaI+1 to ColunaI do

```

```

    begin

```

```

      AlturaEfetivaI:=Trunc(Apontador[LinhaI]-
                            Apontador[LinhaI-1]);

```

```

      IndiceTermo :=IndiceTermo+1;

```

```

      if indiceTermo>MaxDeslocamentos then

```

```

        begin

```

```

          indiceTermo:=1;

```

```

          BlocoTermo:=BlocoTermo+1;

```

```

        end;

```

```

      Aux:=Pegue(BlocoTermo, IndiceTermo);

```

```

      if LinhaI < LinhaI-AlturaEfetivaI+1

```

```

        then MaiorLinha:=LinhaI-AlturaEfetivaI+1

```

```

        else MaiorLinha:=LinhaI;

```

```

Coeficientei := Apontador[Linha] -
                (Linha - MaiorLinha);
Blocoi        := Trunc((Coeficientei - 1) /
                MaxDeslocamentos) + 1;
IndiceI       := Trunc(Coeficientei -
                1.0 * (Blocoi - 1) * MaxDeslocamentos);

Coeficientej := Apontador[Coluna] -
                (Coluna - MaiorLinha);
Blocoj        := Trunc((Coeficientej - 1) /
                MaxDeslocamentos) + 1;
Indicej       := Trunc(Coeficientej - 1.0 * (Blocoj - 1) *
                MaxDeslocamentos);

```

```

for Linha := MaiorLinha to Linha - 1 do
begin
    Aux := Aux - Pegue(Blocoi, Indicei) *
                Pegue(Blocoj, Indicej);
    Indicei := Indicei + 1;
    if Indicei > MaxDeslocamentos then
    begin
        Indicei := 1;
        Blocoi := Blocoi + 1;
    end;
    Indicej := Indicej + 1;
    if Indicej > MaxDeslocamentos then
    begin
        Indicej := 1;
        Blocoj := Blocoj + 1;
    end;
end;

```

```

if Linha <> Coluna then
begin
    Blocoi := Trunc((Apontador[Linha] - 1) /
                    MaxDeslocamentos) + 1;
    Indicei := Trunc(Apontador[Linha] -
                    1.0 * (Blocoi - 1) * MaxDeslocamentos);
    Aux := Aux / Pegue(Blocoi, Indicei);
end;
Ponha(BlocoTermo, IndiceTermo, Aux);
end;
end;

```

{ .PA }

```

begin [ Triangularize Matriz De Rigidez ]
deslocamentos := Nos * DeslocamentosPorNo;
Ponha(1, 1, Sqrt(Pegue(1, 1)));

```

```

for Coluna := 2 to deslocamentos do
begin
    AlturaEfetiva := Trunc(Apontador[Coluna] -
                            Apontador[Coluna - 1]);
    termo          := Apontador[Coluna - 1] + 1;
    BlocoTermo     := Trunc((termo - 1) / MaxDeslocamentos) + 1;
    IndiceTermo    := Trunc(termo - 1.0 * (BlocoTermo - 1) *
                            MaxDeslocamentos);

    linha         := Coluna - AlturaEfetiva + 1;
    BlocoLinha    := Trunc((Apontador[Linha] - 1) /
                            MaxDeslocamentos) + 1;
    IndiceLinha   := Trunc(Apontador[Linha] -

```

```
1.0*(BlocoInhaj-1)*MaxDeslocamentos;
```

```
if (linha) <> Coluna) then ProcresseColuna;
```

```
  Ponha(BlocoTermo,IndiceTermo,
        Sqrt(Pegue(BlocoTermo,IndiceTermo)));
```

```
end;
```

```
end;
```

```
procedure ResolvaSistemasDeEquacoes;
```

```
var
```

```
  Deslocamento:ApontaVetor;
```

```
procedure LeiaVetorIndependente;
```

```
var
```

```
  contador:Integer;
```

```
  ArqR      :file of Real;
```

```
begin
```

```
  Assign(ArqR,drive+' '+Nome+'.F'+
        complemento(Carregamento));
```

```
  Reset(ArqR);
```

```
  for contador:=1 to Nos*DeslocamentosPorNo do
    Read(ArqR,Deslocamento^[contador]);
```

```
  Close(ArqR);
```

```
end;
```

```
{.PA}
```

```
procedure Substituicao;
```

```
var
```

```
  termo,Aux      :Real;
```

```
  BlocoTermo,BlocoDirecao  :1..MaxBlocos;
```

```
  IndiceTermo,IndiceDirecao:Integer;
```

```
  direcao,primeiralinha,
```

```
  AlturaEfetiva,linha      :1..MaxDeslocamentos;
```

```
begin
```

```
  Deslocamento^[1]:=Deslocamento^[1]/Pegue(1,1);
```

```
  for direcao:=2 to Nos*DeslocamentosPorNo do
```

```
    begin
```

```
      AlturaEfetiva:=Trunc(Apontador[direcao]-
                          Apontador[direcao-1]);
```

```
      primeiralinha:=direcao-AlturaEfetiva+1;
```

```
      termo:=Apontador[direcao-1];
```

```
      BlocoTermo :=Trunc((termo-1)/MaxDeslocamentos)+1;
```

```
      IndiceTermo:=Trunc(termo-1.0*(BlocoTermo-1)*
                          MaxDeslocamentos);
```

```
      Aux:=Deslocamento^[direcao];
```

```
      for linha:=primeiralinha to direcao-1 do
```

```
        begin
```

```
          IndiceTermo:=IndiceTermo+1;
```

```
          if IndiceTermo > MaxDeslocamentos then
```

```
            begin
```

```
              IndiceTermo:=1;
```

```
              BlocoTermo:=BlocoTermo+1;
```

```
            end;
```

```
          Aux:=Aux-Deslocamento^[linha]*
```

```
            Pegue(BlocoTermo,IndiceTermo);
```

```
        end;
```

```
      Blocodirecao :=Trunc((Apontador[direcao]-1)/
                          MaxDeslocamentos)+1;
```

```
      Indicedirecao:=Trunc(Apontador[direcao]-1.0*
                          (Blocodirecao-1)*MaxDeslocamentos);
```

```
      Deslocamento^[direcao]:=
```

```

                Aux/Pegue(Blocodirecao,Indicedirecao).
            end;
        end;
    end;
(.PA)
procedure RetroSubstituicao;
var
    termo,Aux                                :Real;
    BlocoTermo,BlocoDirecao  :1..MaxBlocos;
    IndiceTermo,IndiceDirecao:integer;
    direcao,primeiralinha,
    AlturaEfetiva,linha      :1..MaxDeslocamentos;
begin
    for direcao:=Nos*DeslocamentosPorNo downto 2 do
        begin
            Blocodirecao :=Trunc((Apontador[direcao]-1)/
                                   MaxDeslocamentos)+1;
            Indicedirecao:=Trunc(Apontador[direcao]-1.0*
                                   (Blocodirecao-1)*MaxDeslocamentos);

            Deslocamento^[direcao]:=Deslocamento^[direcao]/
                Pegue(Blocodirecao,Indicedirecao);
            AlturaEfetiva:=Trunc(Apontador[direcao]-
                                   Apontador[direcao-1]);
            primeiralinha:=direcao-AlturaEfetiva+1;
            termo:=Apontador[direcao-1];
            BlocoTermo :=Trunc((termo-1)/MaxDeslocamentos)+1;
            IndiceTermo:=Trunc(termo-1.0*(BlocoTermo-1)*
                                   MaxDeslocamentos);

            for linha:=primeiralinha to direcao-1 do
                begin
                    IndiceTermo:=IndiceTermo+1;
                    if IndiceTermo > MaxDeslocamentos then
                        begin
                            IndiceTermo:=1;
                            BlocoTermo:=BlocoTermo+1;
                        end;
                    Deslocamento^[linha]:=Deslocamento^[linha]-
                        Pegue(BlocoTermo,IndiceTermo)*
                        Deslocamento^[direcao];
                end;
            end;
            Deslocamento^[1]:=Deslocamento^[1]/Pegue(1,1).
        end;
    end;

procedure GraveDeslocamentos;
var
    contador:integer;
    ArqR      :file of Real;
begin
    Assign(ArqR,drive+' '+Nome+'.D'+
            complemento(Carregamento));
    Rewrite(ArqR);
    for contador:=1 to Nos*DeslocamentosPorNo do
        Write(ArqR,Deslocamento^[contador]);
    Close(ArqR);
end;

begin { Resolva Sistemas De Equacoes }
    New(Deslocamento);
    for Carregamento:=1 to Carregamentos do
        begin

```

```

        LeiaVetorIndependente.
        Substituicao;
        RetroSubstituicao;
        GraveDeslocamentos.
    end;
    Dispose(Deslocamento);
    Assign(ArqBlocos,drive+':'+Nome+'.007');
    Erase (ArqBlocos);
end;
{ .PA }
begin { Calcule Deslocamentos }
    MonteVetorApontador;
    PrepareBlocos;
    MonteMatrizDeRigidez;
    IntroduzaApoios;
    TriangularizeMatrizDeRigidez;
    SalveBlocos;
    DesaloqueBlocos;
    ResolvaSistemasDeEquacoes;
    DesaloqueBlocos;
end.

procedure LeiaDeslocamentos(Carregamento: Integer;
                            var Deslocamento: VetorReal);

var
    contador: Integer;
    ArqR      : file of Real;
begin
    Assign(ArqR,drive+':'+Nome+'.D'+complemento(Carregamento));
    Reset(ArqR);
    for contador:=1 to Nos*DeslocamentosPorNo do
        Read(ArqR,Deslocamento[contador]);
    Close(ArqR);
end.

procedure CalculeReacoesDeApoio;
var
    Carregamento: 1..MaxCarregamentos;
    Reacao        : TipoReacao;
    Deslocamento: VetorReal;
    direcao       : 1..8;
    posicao        : Integer;
    Apoio         : TipoApoio;
begin
    for Carregamento:=1 to Carregamentos do
        begin
            LeiaDeslocamentos(Carregamento,Deslocamento);
            Assign(ArqReacao,drive+':'+Nome+'.R'+
                    Complemento(Carregamento));
            Rewrite(ArqReacao);

            Assign(ArqApoio,drive+':'+Nome+'.005');
            Reset(ArqApoio);

            while not EOF(ArqApoio) do
                begin
                    Read(ArqApoio,Apoio);
                    Reacao.NumeroNo:=Apoio.NumeroNo;
                    posicao:=(Apoio.NumeroNo-1)*DeslocamentosporNo;
                    for direcao:=1 to DeslocamentosporNo do
                        begin

```

```

        if Apoio.Codigo[direcao] = '1'
            then Reacao.Valor[direcao] := -
                Deslocamento[posicao+direcao]*KApoio
            else Reacao.Valor[direcao] := -
                Deslocamento[posicao+direcao]*
                    Apoio.Condicao[direcao].
        end;
        Write(ArqReacao, Reacao);
    end;
    Close(ArqApoio);
    Close(ArqReacao);
end.
end.
[.PA]
procedure CalculeEsforcosDeExtremidade;
var
    Deslocamento      : VetorReal;
    Carregamento      : 1..MaxCarregamentos;
    EsforcoExtr       : TipoEsforcoExtr;
    ArqAux             : file of TipoEsforcoExtr;
    Elemento, posicao : Integer;
    DeslNo, DeslBarra : 1..12;

procedure CalculeEsforcosDaBarra;
var
    Barra              : TipoBarra;
    Ke                 : Matriz12;
    f, d               : Vetor12;
    direcao, linha, coluna : 1..12;

procedure PreMultipliqueMatrizDeRigidez;
var
    Aux                : Real;
    MaxM, MaxI : 1..4;
    i, j, k, m, n : 1..12;
    R, S              : Matriz3;
begin
    case Controle.TipoEstrutura of
        PorticoEspacial : begin
            MaxM := 4;
            MaxI := 3;
        end;
        PorticoPlano     : begin
            MaxM := 2;
            MaxI := 3;
        end;
        TrelicaEspacial : begin
            MaxM := 2;
            MaxI := 3;
        end;
        TrelicaPlana    : begin
            MaxM := 2;
            MaxI := 2;
        end;
        Greiha           : begin
            MaxM := 2;
            MaxI := 3;
        end;
    end;
end.
ObtenhaMatrizDeRotacao(R, Barra);
for m := 1 to MaxM do

```

```

for n:=1 to MaxM do
  begin
    for i:=1 to MaxI do
      for j:= 1 to MaxI do S[i,j]:=0;
    for i:=1 to MaxI do
      for j:=1 to MaxI do
        begin
          Aux:=0;
          for k:= 1 to MaxI do
            Aux:=Aux+
              R[k,j]*Ke[MaxI*(m-1)+i,MaxI*(n-1)+k];
          S[i,j]:=Aux;
        end;
      for i:=1 to MaxI do
        for j:=1 to MaxI do
          Ke[MaxI*(m-1)+i,MaxI*(n-1)+j]:=S[i,j];
        end;
      end;
    end;
end;
{.GP59}
procedure ConsidereForcasEquivalentesNodais;
begin
  Read(ArqAux,EsforcoExtr);
  with EsforcoExtr do
    case Controle.TipoEstrutura of
      PorticoEspacial:for direcao:=1 to 12 do
        Valor[direcao]:=
          -Valor[direcao]+f[direcao];
      PorticoPlano :begin
        Valor[ 1]:=-Valor[ 1]+f[ 1];
        Valor[ 2]:=-Valor[ 2]+f[ 2];
        Valor[ 3]:=-Valor[ 6]+f[ 3];
        Valor[ 4]:=-Valor[ 7]+f[ 4];
        Valor[ 5]:=-Valor[ 8]+f[ 5];
        Valor[ 6]:=-Valor[12]+f[ 6];
      end;
      TrelicaEspacial:begin
        Valor[ 1]:=-Valor[ 1]+f[ 1];
        Valor[ 2]:=-Valor[ 2]+f[ 2];
        Valor[ 3]:=-Valor[ 3]+f[ 3];
        Valor[ 4]:=-Valor[ 7]+f[ 4];
        Valor[ 5]:=-Valor[ 8]+f[ 5];
        Valor[ 6]:=-Valor[ 9]+f[ 6];
      end;
      TrelicaPlana :begin
        Valor[ 1]:=-Valor[ 1]+f[ 1];
        Valor[ 2]:=-Valor[ 2]+f[ 2];
        Valor[ 3]:=-Valor[ 7]+f[ 3];
        Valor[ 4]:=-Valor[ 8]+f[ 4];
      end;
      Greiha :begin
        Valor[ 1]:=-Valor[ 4]+f[ 1];
        Valor[ 2]:=-Valor[ 2]+f[ 2];
        Valor[ 3]:=-Valor[ 6]+f[ 3];
        Valor[ 4]:=-Valor[10]+f[ 4];
        Valor[ 5]:=-Valor[ 8]+f[ 5];
        Valor[ 6]:=-Valor[12]+f[ 6];
      end;
    end;
  end;
end;
begin { CalculeEsforcosDaBarra }

```

```
Elemento:=Elemento+1;
Read(ArqBarra,Barra);
```

```
case Controle.TipoEstrutura of
  Greiha           :ObtenhaKG (Ke,Barra);
  PorticoEspacial :ObtenhaKPE(Ke,Barra);
  PorticoPlano     :ObtenhaKPP(Ke,Barra);
  TrelicaEspacial :ObtenhaKTE(Ke,Barra);
  TrelicaPlana    :ObtenhaKTP(Ke,Barra);
end;
```

```
if Barra.TemLiberacao then
  LibereMatrizDeRigidez(Ke,Barra);
PreMultipliqueMatrizDeRigidez;
```

```
for direcao:=1 to DeslNo do
  begin
    d[direcao] :=deslocamento[(Barra.No[1]-1)*
                               DeslNo+direcao];
    d[direcao+DeslNo]:=deslocamento[(Barra.No[2]-1)*
                                       DeslNo+direcao];
  end;
```

```
for direcao:=1 to DeslBarra do f[direcao]:=0;
```

```
for linha:=1 to DeslBarra do
  for coluna:=1 to DeslBarra do
    f[linha]:=f[linha]+Ke[linha,coluna]*d[coluna];
```

```
  ConsidereForcasEquivalentesNodais;
  Write(ArqEsforcoExtr,EsforcoExtr);
end;
```

```
{.PA}
```

```
begin { Calcule EsforçosExtremidade }
  DeslNo:=DeslocamentosporNo;
  DeslBarra:=DeslNo*NosporElemento;
  LeiaMaterials;
  LeiaSecoes;
  LeiaNos;
  for Carregamento:=1 to Carregamentos do
    begin
      LeiaDeslocamentos(Carregamento,Deslocamento);

      Assign(ArqAux,drive+': '+Nome+'.N'+
             complemento(carregamento));
      Reset(ArqAux);

      Assign(ArqEsforcoExtr,drive+': '+Nome+'.E'+
             complemento(Carregamento));
      Rewrite(ArqEsforcoExtr);

      Assign(ArqBarra,drive+': '+Nome+'.004');
      Reset(ArqBarra);

      Elemento:=0;
      while not EOF(ArqBarra) do CalculeEsforçosDaBarra;

      Glose(ArqBarra);
      Glose(ArqEsforcoExtr);
      Glose(ArqAux);
      Erase(ArqAux);
```

```
    end;  
    ApagueNos;  
    ApagueSecoes;  
    ApagueMateriais;  
end;
```

```
begin { Analise Estrutura }  
    MonteVetoresIndependentes;  
    CalculeDeslocamentos;  
    CalculeReacoesDeApoio;  
    CalculeEsforcosDeExtremidade;  
end;
```

Apêndice III - Listagem da Rotina Dinâmica

```

procedure ExecuteAnaliseDinamica;
const
  MaxBlocos          = 500;
type
  HoraString         = String[8];
  ApontaVetor        = ^VetorReal;
  VetorPonteiros     = array[1..MaxBlocos] of ApontaVetor;
  VetorEntrada       = array[1..MaxBlocos] of HoraString;
var
  Apontador          : VetorReal;
  BlocoRigidez       : VetorPonteiros;
  EntradaRigidez     : VetorEntrada;
  BlocosNecessarios : 0..MaxBlocos;
  Carregamento      : 1..MaxCarregamentos;

function Memoria:Real;
begin
  if MemAvail > 0 then Memoria:=MemAvail*16.0
    else Memoria:=(MemAvail*16.0+1048576.0);
end;

function Hora:HoraString;
type
  repack            = record
    ax,bx,cx,dx,
    di,si,ds,es,flags:Integer;
  end;
var
  recpack           : repack;
  ah,al,ch,ci,dh    : Byte;
  hour,min,seg      : String[2];

begin
  ah:=$2c;
  with repack do
    begin
      ax:=ah shl 8 + al;
    end;
  intr($21,repack);
  with repack do
    begin
      str(cx shr 8:2, hour);
      str(cx mod 256:2, min);
      str(dx shr 8:2, seg);
    end;
  if min[1] = ' ' then min[1]:='0';
  if seg[1] = ' ' then seg[1]:='0';
  Hora:=hour+' '+min+' '+seg;
end;

procedure DesaloqueMatrizRigidez;
var
  indice:1..MaxBlocos;
begin
  for indice:=1 to BlocosNecessarios do
    if BlocoRigidez[indice] <> nil then
      begin
        Dispose(BlocoRigidez[indice]);
        BlocoRigidez[indice]:=nil;
      end;
end;

```

```

        end;
    end;

procedure ElimineBlocoMatrizDeRigidez;
var
    MaisAntigo: 1..MaxBlocos;

procedure EscolhaBloco;
var
    indice: 1..MaxBlocos;
    aux    : HoraString;
begin { EscolhaBloco }
    MaisAntigo:=1;
    aux:='99:99:99';
    for indice:=1 to BlocosNecessarios do
        if (BlocoRigidez[indice] <> nil) and
            (EntradaRigidez[indice] < aux) then
            begin
                aux      :=EntradaRigidez[indice];
                MaisAntigo:=indice;
            end;
    end;

procedure SalveBloco(n: Integer);
var
    ArqMatrizRigidez : file of VetorReal;
begin
    Assign(ArqMatrizRigidez, drive+' : '+Nome+'.007');
    Reset(ArqMatrizRigidez);
    Seek(ArqMatrizRigidez, n-1);
    Write(ArqMatrizRigidez, BlocoRigidez[n]^);
    Close(ArqMatrizRigidez);
end;

begin { ElimineBlocoMatrizDeRigidez }
    EscolhaBloco;
    if BlocoRigidez[MaisAntigo] <> nil then
        begin
            SalveBloco(MaisAntigo);
            Dispose(BlocoRigidez[MaisAntigo]);
            BlocoRigidez[MaisAntigo]:=nil;
        end;
end;

procedure ChameBlocoMatrizDeRigidez(n: Integer);

procedure LeiaBloco;
var
    ArqMatrizRigidez : file of VetorReal;
begin { LeiaBloco }
    New(BlocoRigidez[n]);
    EntradaRigidez[n]:=Hora;
    Assign(ArqMatrizRigidez, drive+' : '+Nome+'.007');
    Reset(ArqMatrizRigidez);
    Seek(ArqMatrizRigidez, n-1);
    Read(ArqMatrizRigidez, BlocoRigidez[n]^);
    Close(ArqMatrizRigidez);
end;

begin { ChameBlocoMatrizDeRigidez }
    if ((Memoria - 5000) < (SizeOf(VetorReal))) then

```

```

        ElimineBlocoMatrizDeRigidez;
    LeiaBloco;
end;

function PegueMatrizRigidez(QualBloco, indice: Integer): Real;
begin
    if BlocoRigidez[QualBloco] = nil then
        ChameBlocoMatrizDeRigidez(QualBloco);
    PegueMatrizRigidez := BlocoRigidez[QualBloco]^[indice];
end;

procedure PonhaMatrizRigidez(QualBloco, indice: Integer;
                             conteudo: Real);
begin
    if BlocoRigidez[QualBloco] = nil then
        ChameBlocoMatrizDeRigidez(QualBloco);
    BlocoRigidez[QualBloco]^[indice] := conteudo;
end;

procedure PrepareSistema;

procedure MonteVetorApontador;
var
    Indice, TotalDeslocamentos, Elemento,
    MaiorDirecao, MenorDirecao, Dif      : Integer;
    NoInicial, NoFinal, MenorNo, MaiorNo  : 1..MaxNos;
    NodoElemento, direcao                 : 1..6;
    Barra                                  : TipoBarra;

begin { Monte Vetor Apontador }
    TotalDeslocamentos := Nos*DeslocamentosPorNo;
    for Indice := 1 to TotalDeslocamentos do
        Apontador[Indice] := 0;

    Assign(ArqBarra, drive+' : '+Nome+'.004');
    Reset(ArqBarra);

    Elemento := 0;
    while not EOF(ArqBarra) do
        begin
            Elemento := Elemento + 1;
            Read(ArqBarra, Barra);
            NoInicial := Barra.No[1];
            NoFinal   := Barra.No[2];
            if NoInicial < NoFinal then
                MenorNo := NoInicial else MenorNo := NoFinal;

            MenorDirecao := (MenorNo - 1) * DeslocamentosPorNo + 1;
            for NodoElemento := 1 to NosPorElemento do
                for direcao := 1 to DeslocamentosPorNo do
                    begin
                        MaiorDirecao := (Barra.No[NodoElemento] - 1) *
                            DeslocamentosPorNo + direcao;
                        Dif := MaiorDirecao - MenorDirecao + 1;
                        if Apontador[MaiorDirecao] < Dif then
                            Apontador[MaiorDirecao] := Dif;
                    end;
                end;
            end;
        Close(ArqBarra);
        for Indice := 2 to TotalDeslocamentos do
            Apontador[Indice] := Apontador[Indice] +

```

```

                                Apontador[Indice-1]);
end;

procedure SalveBlocos;
var
  ArqMatrizRigidez: file of VetorReal;
  indice           : 1..MaxBlocos;
begin
  Assign (ArqMatrizRigidez, drive+' : '+Nome+'.007');
  Reset(ArqMatrizRigidez);
  for indice:=1 to BlocosNecessarios do
    if BlocoRigidez[indice] <> nil then
      begin
        Seek(ArqMatrizRigidez, indice-1);
        Write(ArqMatrizRigidez, BlocoRigidez[indice]^);
      end;
  Close(ArqMatrizRigidez);
end;

procedure PrepareMatrizDeRigidez;
var
  ArqMatrizRigidez      : file of VetorReal;
  indice                : 1..MaxBlocos;
  deslocamentos, direcao : 1..MaxDeslocamentos;
  TermosDaMatriz        : Real;
  Aux                   : ApontaVetor;
begin
  deslocamentos:=Nos*DeslocamentosPorNo;
  TermosDaMatriz:=Apontador[deslocamentos];
  BlocosNecessarios:=Trunc((TermosDaMatriz-1)/
                             MaxDeslocamentos)+1;

  for indice:=1 to MaxBlocos do
    begin
      BlocoRigidez[indice]:=nil;
      EntradaRigidez[indice]:=Hora;
    end;

  New(Aux);
  for direcao:=1 to MaxDeslocamentos do
    Aux^[direcao]:=0;

  Assign(ArqMatrizRigidez, drive+' : '+Nome+'.007');
  Rewrite(ArqMatrizRigidez);
  for indice:=1 to BlocosNecessarios do
    Write(ArqMatrizRigidez, Aux^);
  Close(ArqMatrizRigidez);

  Dispose(Aux);
end;

procedure MonteMatrizDeRigidez;
var
  Barra           : TipoBarra;
  indice          : array[1..12, 1..12] of Integer;
  correspondencia : array[1..12] of Integer;
  Elemento        : Integer;
  Deslocamentos   : 1..MaxDeslocamentos;
  NumeroBloco     : 1..MaxBlocos;
  considere       : Boolean;

```

```

procedure MonteCorrespondencia;
var
  DirecaoElemento : 0..12;
  NodoElemento    : 1..NosPorElemento;
  DirecaoNo       : 1..6;
  DirecaoAnterior : Integer;
begin
  with Barra do
  begin
    DirecaoElemento:=0;
    for NodoElemento:=1 to NosPorElemento do
    begin
      DirecaoAnterior:=(No[NodoElemento]-1)*
        DeslocamentosporNo;
      for DirecaoNo:=1 to DeslocamentosporNo do
      begin
        DirecaoElemento:=DirecaoElemento+1;
        Correspondencia[DirecaoElemento]:=
          DirecaoAnterior+DirecaoNo;
      end;
    end;
  end;
end;

```

```

procedure CalculeIndices;
var
  linha,coluna : 0..12;
  aux          : Real;
begin
  considere:=false;
  for linha:=1 to Deslocamentos do
    for coluna:=1 to Deslocamentos do
      if not (Correspondencia[linha] >
        Correspondencia[coluna]) then
        begin
          aux:=Correspondencia[linha]-
            Correspondencia[coluna]+
            Apontador[Correspondencia[coluna]]-
            1.0*(NumeroBloco-1)*MaxDeslocamentos;
          if Abs(aux) > MaxDeslocamentos then aux:=0;
          indice[linha,coluna]:=Trunc(aux);
          considere:=considerere or (aux > 0);
        end;
    end;
  end;

```

```

procedure ConsidereBarra;
var
  Ke          : Matriz12;
  linha,coluna : 0..12;

```

```

procedure MudeParaReferenciaIGlobal;
var
  i,j,k,l,m,n: 1..12;
  R,S        : Matriz3;
  Aux        : Real;
  MaxM,MaxI : 1..4;
begin
  case Controle.TipoEstrutura of
    PorticoEspacial:begin
      MaxM:=4;
      MaxI:=3;

```

```

end;
PorticoPlano :begin
    MaxM:=2;
    MaxI:=3;
end;
TrelicaEspacial:begin
    MaxM:=2;
    MaxI:=3;
end;
TrelicaPlana :begin
    MaxM:=2;
    MaxI:=2;
end;
Grelha :begin
    MaxM:=2;
    MaxI:=3;
end;
end;
ObtenhaMatrizDeRotacao(R,Barra);
for m:=1 to MaxM do
    for n:=m to MaxM do
        begin
            for i:=1 to MaxI do
                for j:= 1 to MaxI do S[i,j]:=0;
                    for l:=1 to MaxI do
                        for j:=1 to MaxI do
                            begin
                                Aux:=0;
                                for k:= 1 to MaxI do
                                    for l:=1 to MaxI do
                                        Aux:=Aux+R[l,i]*R[k,j]*
                                            Ke[MaxI*(m-1)+l,MaxI*(n-1)+k];
                                        S[i,j]:=Aux;
                                    end;
                                for i:=1 to MaxI do
                                    for j:=1 to MaxI do
                                        Ke[MaxI*(m-1)+i,MaxI*(n-1)+j]:=S[i,j];
                                    end;
                                for i:=1 to 12 do
                                    for j:=i+1 to 12 do Ke[j,i]:=Ke[i,j];
                                end;
                            end;
                        end;
                    end;
                end;
            end;
        end;
end;
begin { Considere Barra }
case Controle.TipoEstrutura of
    Grelha :ObtenhaKG (Ke,Barra);
    PorticoEspacial:ObtenhaKPE(Ke,Barra);
    PorticoPlano :ObtenhaKPP(Ke,Barra);
    TrelicaEspacial:ObtenhaKTE(Ke,Barra);
    TrelicaPlana :ObtenhaKTP(Ke,Barra);
end;
if Barra.TemLiberacao then
    LibereMatrizDeRigidez(Ke,Barra);
MudeParaReferencialGlobal;

for linha:=1 to Deslocamentos do
    for coluna:=1 to Deslocamentos do
        if not (Correspondencia[linha] >
            Correspondencia[coluna]) then
            if indice[linha,coluna] > 0 then
                PonhaMatrizRigidez(NumeroBloco,
                    indice[linha,coluna],

```

```

                PegueMatrizRigidez(NumeroBloco,
                                    indice[linha,coluna])+
                                    Ke[linha,coluna]);
    end;

begin { Monte Matriz De Rigidez }
  LeiaMateriais;
  LeiaSecoes;
  LeiaNos;
  Deslocamentos:=NosporElemento*DeslocamentosporNo;
  for NumeroBloco:=1 to BlocosNecessarios do
    begin
      Assign(ArqBarra,drive+' '+Nome+'.004');
      Reset(ArqBarra);
      Elemento:=0;
      while not EOF(ArqBarra) do
        begin
          Elemento:=Elemento+1;
          Read(ArqBarra,Barra);
          MonteCorrespondencia;
          CalculeIndices;
          if Considere then ConsidereBarra;
        end;
      Close(ArqBarra);
    end;
  ApagueNos;
  ApagueSecoes;
  ApagueMateriais;
end;

procedure IntroduzaApoios;
var
  Apoio           :TipoApoio;
  Numero,Direcao  :Integer;
  DirecaoNo       :1..6;
  Posicao          :1..MaxDeslocamentos;
  NumeroBloco     :1..MaxBlocos;
  aux             :Real;

begin
  Assign(ArqApoio,drive+' '+Nome+'.005');
  Reset(ArqApoio);
  Numero:=0;
  while not EOF(ArqApoio) do
    with Apoio do
      begin
        Read(ArqApoio,Apoio);
        Numero:=Numero+1;
        direcao:=(NumeroNo-1)*DeslocamentosporNo;
        for direcaoNo:=1 to DeslocamentosporNo do
          begin
            direcao:=direcao+1;
            NumeroBloco :=Trunc((Apontador[direcao]-1)/
                                MaxDeslocamentos)+1;
            posicao:=Trunc(Apontador[direcao]-1.0*
                          (NumeroBloco-1)*MaxDeslocamentos);
            aux:=PegueMatrizRigidez(NumeroBloco,posicao);
            if Codigo[direcaoNo] = '1'
              then PonhaMatrizRigidez(NumeroBloco,
                                       posicao,aux+KApoio)
              else PonhaMatrizRigidez(NumeroBloco,

```

```

                                posicao,aux+Condicao(direcaoNo))
        end;
    end;
    Close(ArqApoio);
end;

procedure TriangularizeMatrizDeRigidez;
const
    Tolerancia = 1E-10;
var
    termo                                :Real;
    BlocoTermo,BlocoLinhaJ              :0..MaxBlocos;
    IndiceTermo,IndiceLinhaJ            :Integer;
    linhaJ,colunaJ,
    AlturaEfetivaJ,deslocamentos:1..MaxDeslocamentos;

procedure ProcesseColuna;
var
    BlocoI,BlocoJ                        :1..MaxBlocos;
    CoeficienteI,CoeficienteJ,Aux:Real;
    IndiceI,IndiceJ                      :Integer;
    linhaI,AlturaEfetivaI,
    linha,MaiorLinha                    :0..MaxDeslocamentos;
begin
    PonhaMatrizRigidez(BlocoTermo,IndiceTermo,
        PegueMatrizRigidez(BlocoTermo,IndiceTermo)/
        PegueMatrizRigidez(BlocoLinhaJ,IndiceLinhaJ));

    for linhaI:=linhaJ+1 to ColunaJ do
        begin
            AlturaEfetivaI:=Trunc(Apontador[linhaI]-
                Apontador[linhaI-1]);
            IndiceTermo :=IndiceTermo+1;
            if IndiceTermo>MaxDeslocamentos then
                begin
                    IndiceTermo:=1;
                    BlocoTermo:=BlocoTermo+1;
                end;
            Aux:=PegueMatrizRigidez(BlocoTermo,IndiceTermo);
            if linhaJ < linhaI-AlturaEfetivaI+1
                then MaiorLinha:=linhaI-AlturaEfetivaI+1
                else MaiorLinha:=linhaJ;

            CoeficienteI:=Apontador[linhaI ]-
                (linhaI-MaiorLinha);
            BlocoI      :=Trunc((CoeficienteI-1)/
                MaxDeslocamentos)+1;
            IndiceI     :=Trunc(CoeficienteI-1.0*(BlocoI-1)*
                MaxDeslocamentos);

            CoeficienteJ:=Apontador[ColunaJ]-
                (ColunaJ-MaiorLinha);
            BlocoJ      :=Trunc((CoeficienteJ-1)/
                MaxDeslocamentos)+1;
            IndiceJ     :=Trunc(CoeficienteJ-1.0*(BlocoJ-1)*
                MaxDeslocamentos);

            for linha:=MaiorLinha to linhaI-1 do
                begin
                    Aux:= Aux-PegueMatrizRigidez(BlocoI,IndiceI)*
                        PegueMatrizRigidez(BlocoJ,IndiceJ);
                end;
            end;
        end;
    end;
end;

```

```

        Indicei:=Indicei+1;
        if Indicei>MaxDeslocamentos then
            begin
                Indicei:=1;
                Blocoi:=Blocoi+1;
            end;
        Indicej:=Indicej+1;
        if Indicej>MaxDeslocamentos then
            begin
                Indicej:=1;
                Blocoj:=Blocoj+1;
            end;
        end;

        if linhaj <> Colunaj then
            begin
                Blocoi :=Trunc((Apontador[linhaj]-1)/
                    MaxDeslocamentos)+1;
                Indicei:=Trunc(Apontador[linhaj]-
                    1.0*(Blocoi-1)*MaxDeslocamentos);
                Aux:=Aux/PegueMatrizRigidez(Blocoi,Indicei);
            end;
            PonhaMatrizRigidez(BlocoTermo,IndiceTermo,Aux);
        end;
    end;

begin { Triangularize Matriz De Rigidez }
    deslocamentos:=Nos*DeslocamentosPorNo;
    PonhaMatrizRigidez(1,1,Sqrt(PegueMatrizRigidez(1,1) ));

    for Colunaj:=2 to deslocamentos do
        begin
            AlturaEfetivaj:=Trunc(Apontador[Colunaj]-
                Apontador[Colunaj-1]);
            termo           :=Apontador[Colunaj-1] + 1;
            BlocoTermo      :=Trunc((termo-1)/MaxDeslocamentos)+1;
            IndiceTermo     :=Trunc(termo-1.0*(BlocoTermo-1)*
                MaxDeslocamentos);

            linhaj          :=Colunaj-AlturaEfetivaj+1;
            Blocolinhaj     :=Trunc((Apontador[linhaj]-1)/
                MaxDeslocamentos)+1;
            Indicelinhaj    :=Trunc(Apontador[linhaj]-
                1.0*(Blocolinhaj-1)*MaxDeslocamentos);

            if linhaj <> Colunaj then ProcesseColuna;

            PonhaMatrizRigidez(BlocoTermo,IndiceTermo,
                Sqrt(PegueMatrizRigidez(BlocoTermo,IndiceTermo)));
        end;
    end;

begin { Prepare Sistema }
    MonteVetorApontador;
    PrepareMatrizDeRigidez;
    MonteMatrizDeRigidez;
    IntroduzaApoios;
    TriangularizeMatrizDeRigidez;
    SalveBlocos;
    DesaloqueMatrizRigidez;
end;

```

```

procedure ObtenhaResposta;
type
  TipoBaseRitz = array[1..MaxRitz] of ApontaVetor;
  TipoRitz     = array[1..MaxRitz] of Real;
var
  Ritz,Kx           : TipoBaseRitz;
  ArqRitz          : file of VetorReal;
  AutoValor,f      : TipoRitz;
  Nv,Carregamento : Integer;
  EntradaRitz,EntradaKx : VetorEntrada;

procedure LeiaNv;
begin
  ClrScr;
  Write('Entre com o numero de vetores :');
  repeat
    Read(Nv);
  until Nv in [1..MaxRitz];
end;

procedure DesaloqueBaseDeRitz;
var
  indice:1..MaxRitz;
begin
  for indice:=1 to Nv do
    if Ritz[indice] <> nil then
      begin
        Dispose(Ritz[indice]);
        Ritz[indice]:=nil;
      end;
  end;
end;

procedure ElimineBlocoRitz;
var
  MaisAntigo:1..MaxRitz;

  procedure EscolhaBloco;
  var
    indice:1..MaxRitz;
    aux    :HoraString;
  begin { EscolhaBloco }
    MaisAntigo:=1;
    aux:='99:99:99';
    for indice:=1 to Nv do
      if (Ritz[indice] <> nil) and
        (EntradaRitz[indice] < aux) then
        begin
          aux           :=EntradaRitz[indice];
          MaisAntigo:=indice;
        end;
    end;
  end;

procedure SalveBloco(n:Integer);
var
  ArqRitz : file of Real;
  indice  : 1..MaxDeslocamentos;
begin
  Assign(ArqRitz,drive+' : '+Nome+'.Z'+Carregamento);
  Reset(ArqRitz);
  Seek(ArqRitz,(n-1)*Nos*DeslocamentosPorNo);

```

```

    for indice:=1 to Nos*DeslocamentosPorNo do
      Write(ArqRitz,Ritz[n]^[indice]);
    Close(ArqRitz);
  end;

begin { ElimineBlocoRitz }
  EscolhaBloco;
  if Ritz[MaisAntigo] <> nil then
    begin
      SalveBloco(MaisAntigo);
      Dispose(Ritz[MaisAntigo]);
      Ritz[MaisAntigo]:=nil;
    end;
end;

procedure ElimineBlocoKx;
var
  MaisAntigo:1..MaxRitz;

  procedure EscolhaBloco;
  var
    indice:1..MaxRitz;
    aux    :HoraString;
  begin { EscolhaBloco }
    MaisAntigo:=1;
    aux:='99:99:99';
    for indice:=1 to Nv do
      if (Kx[indice] <> nil) and
        (EntradaKx[indice] < aux) then
        begin
          aux      :=EntradaKx[indice];
          MaisAntigo:=indice;
        end;
    end;
  end;

  procedure SalveBloco(n:Integer);
  var
    ArqKx   : file of Real;
    indice  : 1..MaxDeslocamentos;
  begin
    Assign(ArqKx,drive+' '+Nome+'.$'+
           Garregamento);
    Reset(ArqKx);
    Seek(ArqKx,(n-1)*Nos*DeslocamentosPorNo);
    for indice:=1 to Nos*DeslocamentosPorNo do
      Write(ArqKx,Kx[n]^[indice]);
    Close(ArqKx);
  end;

begin { ElimineBlocoKx }
  EscolhaBloco;
  if Kx[MaisAntigo] <> nil then
    begin
      SalveBloco(MaisAntigo);
      Dispose(Kx[MaisAntigo]);
      Kx[MaisAntigo]:=nil;
    end;
end;

procedure ChameBlocoRitz(n:Integer);

```

```

procedure LeiaBloco;
var
  ArqRitz   : file of Real;
  indice    : 1..MaxDeslocamentos;
begin
  New(Ritz[n]);
  EntradaRitz[n]:=Hora;
  Assign(ArqRitz,drive+' '+Nome+'.Z'+
         Carregamento);
  Reset(ArqRitz);
  Seek(ArqRitz,(n-1)*Nos*DeslocamentosPorNo);
  for indice:=1 to Nos*DeslocamentosPorNo do
    Read(ArqRitz,Ritz[n]^[indice]);
  Close(ArqRitz);
end;

begin { ChameBlocoRitz}
  if ((Memoria - 5000) < (2*SizeOf(VetorReal))) then
    ElimineBlocoMatrizDeRigidez;
  if ((Memoria - 5000) < (2*SizeOf(VetorReal))) then
    ElimineBlocoKx;
  if ((Memoria - 5000) < (2*SizeOf(VetorReal))) then
    ElimineBlocoRitz;
  LeiaBloco;
end;

function PegueRitz(QualBloco,indice:Integer):Real;
begin
  if Ritz[QualBloco] = nil then ChameBlocoRitz(QualBloco);
  PegueRitz:=Ritz[QualBloco]^[indice];
end;

procedure PonhaRitz(QualBloco,indice:Integer;
                   conteudo:Real);
begin
  if Ritz[QualBloco] = nil then
    ChameBlocoRitz(QualBloco);
  Ritz[QualBloco]^[indice]:=conteudo;
end;

procedure ChameBlocoKx(n:Integer);

procedure LeiaBloco;
var
  ArqKx     : file of Real;
  indice    : 1..MaxDeslocamentos;
begin
  New(Kx[n]);
  EntradaKx[n]:=Hora;
  Assign(ArqKx,drive+' '+Nome+'.K'+
         Carregamento);
  Reset(ArqKx);
  Seek(ArqKx,(n-1)*Nos*DeslocamentosPorNo);
  for indice:=1 to Nos*DeslocamentosPorNo do
    Read(ArqKx,Kx[n]^[indice]);
  Close(ArqKx);
end;

begin { ChameBlocoKx}
  if ((Memoria - 5000) < (2*SizeOf(VetorReal))) then
    ElimineBlocoMatrizDeRigidez;

```

```

    if ((Memoria - 5000) < (2*SizeOf(VetorReal))) then
      ElimineBlocoRitz;
    if ((Memoria - 5000) < (2*SizeOf(VetorReal))) then
      ElimineBlocoKx;
  LeiaBloco;
end;

function PegueKx(QualBloco, indice: Integer): Real;
begin
  if Kx[QualBloco] = nil then ChameBlocoKx(QualBloco);
  PegueKx := Kx[QualBloco]^[indice];
end;

procedure PonhaKx(QualBloco, indice: Integer; conteudo: Real);
begin
  if Kx[QualBloco] = nil then ChameBlocoKx(QualBloco);
  Kx[QualBloco]^[indice] := conteudo;
end;

procedure MonteBaseDeRitz;
var
  AutoVetor, KR: array[1..MaxRitz, 1..MaxRitz] of Real;
  deslocamentos: Integer;

procedure LeiaVetorIndependente(Carregamento: Integer;
                                var Deslocamento: VetorReal);
var
  contador: Integer;
  ArqR      : file of Real;
begin
  Assign(ArqR, drive+' : '+Nome+'.F'+
        Carregamento);
  Reset(ArqR);
  for contador := 1 to Nos*DeslocamentosPorNo do
    Read(ArqR, Deslocamento[contador]);
  Close(ArqR);
end;

procedure MonteSistemaReduzido;
var
  M : VetorReal;

procedure DesaloqueKx;
var
  ArqKx : file;
  indice: 1..MaxRitz;
begin
  for indice := 1 to Nv do
    if Kx[indice] <> nil then
      begin
        Dispose(Kx[indice]);
        Kx[indice] := nil;
      end;
  Assign(ArqKx, drive+' : '+Nome+'.F'+
        Carregamento);
  Erase(ArqKx);
end;

procedure MonteMatrizDeMassa;
var
  Barra: TipoBarra;

```

```

Me      :VetorReal;
i       :1..MaxDeslocamentos;
NumeroBarra:Integer;

```

```

procedure ConsidereMassasDiscretas;

```

```

var
  ArqR : file of Real;
  i,j   : 1..MaxNos;
  Aux   : Real;
begin
  Assign(ArqR,drive+' '+Nome+'.006');
  Reset(ArqR);
  for i:=1 to Nos do
    begin
      Read(ArqR,Aux);
      case Controle.TipoEstrutura of
        Grelha:M[3*(i-1)+2]:=M[3*(i-1)+2]+Aux;
        PorticoEspacial:for j:=1 to 3 do
          M[6*(i-1)+j]:=M[6*(i-1)+j]+Aux;
        PorticoPlano   :for j:=1 to 2 do
          M[3*(i-1)+j]:=M[3*(i-1)+j]+Aux;
        TrelicaEspacial:for j:=1 to 3 do
          M[3*(i-1)+j]:=M[3*(i-1)+j]+Aux;
        TrelicaPlana   :for j:=1 to 2 do
          M[2*(i-1)+j]:=M[2*(i-1)+j]+Aux;
      end;
    end;
  Close(ArqR);
end;

```

```

procedure CalculeMatrizDeMassaDaBarra;

```

```

var
  TipoMaterial           :1..MaxMateriais;
  TipoSecao              :1..MaxSecoes;
  NoInicial,NoFinal      :1..MaxNos;
  Peso,cx,cy,cz,L,aux1,aux2,aux3:Real;

```

```

procedure CalculeMatrizDeMassaGrelha;

```

```

var
  R:Matriz3;
begin { CalculeMatrizDeMassaGrelha }
  ObtenhaMatrizDeRotacao(R,Barra);
  Me[ 1]:=aux2*Sqr(R[1,1])+aux3*Sqr(R[3,1]);
  Me[ 2]:=aux1;
  Me[ 3]:=aux2*Sqr(R[1,3])+aux3*Sqr(R[3,3]);
  Me[ 4]:=Me[1];
  Me[ 5]:=Me[2];
  Me[ 6]:=Me[3];
end;

```

```

procedure CalculeMatrizDeMassaPorticoEspacial;

```

```

var
  R:Matriz3;
begin { CalculeMatrizDeMassaPorticoEspacial }
  ObtenhaMatrizDeRotacao(R,Barra);
  Me[ 1]:=aux1;
  Me[ 2]:=aux1;
  Me[ 3]:=aux1;
  Me[ 4]:=aux2*Sqr(R[1,1])+aux3*Sqr(R[2,1])+
    aux3*Sqr(R[3,1]);
  Me[ 5]:=aux2*Sqr(R[1,2])+aux3*Sqr(R[2,2])+

```

```

        aux3*Sqr(R[3,2]);
Me[ 6] :=aux2*Sqr(R[1,3])+aux3*Sqr(R[2,3])+
        aux3*Sqr(R[3,3]);
Me[ 7] :=aux1;
Me[ 8] :=aux1;
Me[ 9] :=aux1;
Me[10] :=Me[4];
Me[11] :=Me[5];
Me[12] :=Me[6];
end;

procedure CalculeMatrizDeMassaPorticoPlano;
begin { CalculeMatrizDeMassaPorticoPlano }
Me[ 1] :=aux1;
Me[ 2] :=aux1;
Me[ 3] :=aux3;
Me[ 4] :=aux1;
Me[ 5] :=aux1;
Me[ 6] :=aux3;
end;

procedure CalculeMatrizDeMassaTrelicaPlana;
begin { CalculeMatrizDeMassaTrelicaPlana }
Me[ 1] :=aux1;
Me[ 2] :=aux1;
Me[ 3] :=aux1;
Me[ 4] :=aux1;
end;

procedure CalculeMatrizDeMassaTrelicaEspacial;
begin { CalculeMatrizDeMassaTrelicaEspacial }
Me[ 1] :=aux1;
Me[ 2] :=aux1;
Me[ 3] :=aux1;
Me[ 4] :=aux1;
Me[ 5] :=aux1;
Me[ 6] :=aux1;
end;

begin { CalculeMatrizDeMassaDaBarra }
with Barra do
begin
TipoMaterial :=Material;
TipoSecao :=Secao;
NoInicial :=No[1];
NoFinal :=No[2];
end;
Peso :=Material[TipoMaterial]^Densidade*
        Secao[TipoSecao]^Ax;
cx :=No[NoFinal]^Coord[1]-
        No[NoInicial]^Coord[1];
cy :=No[NoFinal]^Coord[2]-
        No[NoInicial]^Coord[2];
cz :=No[NoFinal]^Coord[3]-
        No[NoInicial]^Coord[3];
L :=Sqr(cx*cx+cy*cy+cz*cz);
aux1 :=Peso*L/2;
aux2 :=Peso*L*Secao[TipoSecao]^Ix/
        (2*Secao[TipoSecao]^Ax);
aux3 :=Peso*L*L*L/36;
case Controle.TipoEstrutura of

```

```

Grelha           :CalculeMatrizDeMassaGrelha;
PorticoEspacial :CalculeMatrizDeMassaPorticoEspacial;
PorticoPlano     :CalculeMatrizDeMassaPorticoPlano;
TrelicaEspacial :CalculeMatrizDeMassaTrelicaEspacial;
TrelicaPlana    :CalculeMatrizDeMassaTrelicaPlana;
end;
end;

procedure ConsidereBarra;
var
  DirecaoNoNo      :1..6;
  DirecaoNoElemento:0..12;
  NoDoElemento     :1..NosPorElemento;
  i,Direcaoanterior:0..MaxDeslocamentos;
  Correspondencia  :array[1..12] of Integer;

begin { ConsidereBarra }
  with Barra do
  begin
    DirecaoNoElemento:=0;
    for NoDoElemento:=1 to NosPorElemento do
      begin
        DirecaoAnterior:=(No[NoDoElemento]-1)*
          DeslocamentosPorNo;
        for DirecaoNoNo:=1 to DeslocamentosPorNo do
          begin
            DirecaoNoElemento:=DirecaoNoElemento+1;
            Correspondencia[DirecaoNoElemento]:=
              DirecaoAnterior+DirecaoNoNo;
          end;
        end;
      end;
    for i:=1 to NosPorElemento*DeslocamentosPorNo do
      M[Correspondencia[i]]:=M[Correspondencia[i]]+Me[i];
    end;
  end;

begin { Monte Matriz De Massa }
  LeiaMateriais;
  LeiaSecoes;
  LeiaNos;
  for i:=1 to MaxDeslocamentos do M[i]:=0;
  ConsidereMassasDiscretas;
  Assign(ArqBarra,drive+' '+Nome+'.004');
  Reset(ArqBarra);
  NumeroBarra:=0;
  while not EOF(ArqBarra) do
    begin
      NumeroBarra:=NumeroBarra+1;
      Read(ArqBarra,Barra);
      CalculeMatrizDeMassaDaBarra;
      ConsidereBarra;
    end;
  Glose(ArqBarra);
  ApagueNos;
  ApagueSecoes;
  ApagueMateriais;
end;

procedure CalculeVetoresDeRitz;
var
  i:1..MaxRitz;

```

```

j:=1..MaxDeslocamentos;

procedure PrepareVetoresDeRitz;
var
  aux      :Real;
  ArqRitz  :file of Real;
  indice   :1..MaxRitz;
  direcao  :Integer;
begin
  for indice:=1 to MaxRitz do
    begin
      Ritz[indice]:=nil;
      EntradaRitz[indice]:=Hora;
    end;

  Aux:=0;
  Assign(ArqRitz,drive+':'+Nome+'.Z'+
        Carregamento);
  Rewrite(ArqRitz);
  for direcao:=1 to Nv*Nos*DeslocamentosPorNo do
    Write(ArqRitz,Aux);
  Close(ArqRitz);
end;

procedure PrepareKx;
var
  Aux      :Real;
  ArqKx    :file of Real;
  indice   :1..MaxRitz;
  direcao  :Integer;
begin
  for indice:=1 to MaxRitz do
    begin
      Kx[indice]:=nil;
      EntradaKx[indice]:=Hora;
    end;

  Aux:=0;
  Assign(ArqKx,drive+':'+Nome+'.\$'+
        Carregamento);
  Rewrite(ArqKx);
  for direcao:=1 to Nv*Nos*DeslocamentosPorNo do
    Write(ArqKx,Aux);
  Close(ArqKx);
end;

procedure ResolvaSistemaDeEquacoes(var deslocamento:
                                     VetorReal);

procedure Substituicao;
var
  termo,Aux      :Real;
  BlocoTermo,BlocoDirecao  :1..MaxBlocos;
  IndiceTermo,IndiceDirecao :Integer;
  direcao,primeiralinha,
  AlturaEfetiva,linha      :1..MaxDeslocamentos;
begin
  Deslocamento[1]:=Deslocamento[1]/
    PegueMatrizRigidez(1,1);
  for direcao:=2 to Nos*DeslocamentosPorNo do
    begin

```

```

AlturaEfetiva:=Trunc(Apontador[direcao]-
                    Apontador[direcao-1]);
primeiralinha:=direcao-AlturaEfetiva+1;
termo:=Apontador[direcao-1];
BlocoTermo :=Trunc((termo-1)/
                    MaxDeslocamentos)+1;
IndiceTermo:=Trunc(termo-1.0*(BlocoTermo-1)*
                    MaxDeslocamentos);
Aux:=Deslocamento[direcao];
for linha:=primeiralinha to direcao-1 do
begin
    IndiceTermo:=IndiceTermo+1;
    if IndiceTermo > MaxDeslocamentos then
begin
        IndiceTermo:=1;
        BlocoTermo:=BlocoTermo+1;
    end;
    Aux:=Aux-Deslocamento[linha]*
        PegueMatrizRigidez(BlocoTermo,
                            IndiceTermo);
end;
Blocodirecao :=Trunc((Apontador[direcao]
                    -1)/MaxDeslocamentos)+1;
Indicedirecao:=Trunc(Apontador[direcao]
                    -1.0*(Blocodirecao-1)*MaxDeslocamentos);
Deslocamento[direcao]:=Aux/
    PegueMatrizRigidez(Blocodirecao,Indicedirecao)
end;
end;

procedure RetroSubstituicao;
var
    termo,Aux                                     :Real;
    BlocoTermo,Blocodirecao :1..MaxBlocos;
    IndiceTermo,Indicedirecao:Integer;
    direcao,primeiralinha,
    AlturaEfetiva,linha :1..MaxDeslocamentos;
begin
    for direcao:=Nos*DeslocamentosPorNo downto 2 do
begin
    Blocodirecao :=Trunc((Apontador[direcao]-1)/
                        MaxDeslocamentos)+1;
    Indicedirecao:=Trunc(Apontador[direcao]-1.0*
                        (Blocodirecao-1)*MaxDeslocamentos);

    Deslocamento[direcao]:=Deslocamento[direcao]/
    PegueMatrizRigidez(Blocodirecao,Indicedirecao)
    AlturaEfetiva:=Trunc(Apontador[direcao]-
                        Apontador[direcao-1]);
    primeiralinha:=direcao-AlturaEfetiva+1;
    termo:=Apontador[direcao-1];
    BlocoTermo :=Trunc((termo-1)/MaxDeslocamentos)+1;
    IndiceTermo:=Trunc(termo-1.0*(BlocoTermo-1)*
                        MaxDeslocamentos);
    for linha:=primeiralinha to direcao-1 do
begin
        IndiceTermo:=IndiceTermo+1;
        if IndiceTermo > MaxDeslocamentos then
begin
            IndiceTermo:=1;
            BlocoTermo:=BlocoTermo+1;
        end;
    end;
end;
end;

```

```

        end;
        Deslocamento[Linha]:= Deslocamento[Linha]
        -PegueMatrizRigidez(BlocoTermo,
        IndiceTermo)*Deslocamento[direcao];
    end;
end;
Deslocamento[1]:=Deslocamento[1]/
    PegueMatrizRigidez(1,1);
end;

begin { Resolva Sistema De Equacoes }
    Substituicao;
    RetroSubstituicao;
end;

procedure NormalizeVetor(i: Integer);
var
    aux: Real;
    j : 1..MaxDeslocamentos;
begin { Normalize Vetor }
    aux:=0;
    for j:=1 to deslocamentos do
        aux:=aux+M[j]*Sqr(PegueRitz(i,j));
    aux:=Sqr(aux);
    for j:=1 to deslocamentos do
        begin
            PonhaRitz(i,j,PegueRitz(i,j)/aux);
            PonhaKx(i,j,PegueKx(i,j)/aux);
        end;
    end;
end;

procedure OrtogonalizeVetor(i: Integer);
var
    j : 1..MaxRitz;
    k : 1..MaxDeslocamentos;
    C : array[1..MaxRitz] of Real;
begin { Ortogonalize Vetor }

    for j:=1 to i-1 do { Calculo de Cj }
        begin
            C[j]:=0;
            for k:=1 to deslocamentos do
                C[j]:=C[j]+PegueRitz(j,k)*M[k]*PegueRitz(i,k);
            end;

        for j:=1 to i-1 do
            for k:=1 to deslocamentos do
                begin
                    PonhaRitz(i,k,PegueRitz(i,k)-
                        C[j]*PegueRitz(j,k));
                    PonhaKx(i,k,PegueKx(i,k)-
                        C[j]*PegueKx(j,k));
                end;
            end;
        end;
end;

begin { Calcule Vetores De Ritz }
    PrepareVetoresDeRitz;
    PrepareKx;
    New(Ritz[1]);
    LeiaVetorIndependente(Carregamento,Ritz[1]^);
    for j:=1 to deslocamentos do

```

```

    PonhaKx(1,j,PegueRitz(1,j));
    ResolvaSistemaDeEquacoes(Ritz[1]^);
    NormalizeVetor(1);
    for i:=2 to Nv do { Calculando demais vetores }
    begin
        for j:=1 to deslocamentos do
            PonhaRitz(i,j,PegueRitz(i-1,j)*M[j]);
        for j:=1 to deslocamentos do
            PonhaKx(i,j,PegueRitz(i,j));
        ResolvaSistemaDeEquacoes(Ritz[i]^);
        OrtogonalizeVetor(i);
        NormalizeVetor(i);
    end;
    DesaloqueMatrizRigidez;
end;

```

```

procedure MonteMatrizReduzida:

```

```

    var
        i,j:1..MaxRitz;
        k :1..MaxDeslocamentos;
    begin { Monte Matriz Reduzida }
        for i:=1 to Nv do { KR = Ritz t Kx }
            for j:=1 to i do
                begin
                    KR[i,j]:=0;
                    for k:=1 to deslocamentos do
                        KR[i,j]:=KR[i,j]+PegueRitz(i,k)*PegueKx(j,k);
                    end;
                for i:=1 to Nv do for j:=i+1 to Nv do KR[i,j]:=KR[j,i];
            end;

```

```

begin { Monte Sistema Reduzido }

```

```

    MonteMatrizDeMassa;
    CalculeVetoresDeRitz;
    MonteMatrizReduzida;
    DesaloqueKx;
end;

```

```

procedure DesacopleSistemaReduzido:

```

```

procedure InicializeAutoValores:

```

```

    var
        i:1..MaxRitz;
    begin { Inicialize AutoValores }
        for i:=1 to Nv do AutoValor[i]:=KR[i,i];
    end;

```

```

procedure InicializeAutovetores:

```

```

    var
        i,j:1..MaxRitz;
    begin { Inicialize AutoVetores }
        for i:=1 to Nv do
            begin
                for j:=1 to Nv do AutoVetor[i,j]:=0;
                Autovetor[i,i]:=1;
            end;
        end;

```

```

procedure CalculeAutoVetores:

```

```

    const
        MaxIteracoes = 20;

```

```

Tolerancia      = 1e-20;
var
  Convergiao    : Boolean;
  Maior         : Real;
  i,j,k         : 1..MaxRitz;
  iteracao      : 0..MaxIteracoes;
  Lambda        : array[1..MaxRitz] of Real;

procedure ERRO;
begin
  Write(†007);
end;

procedure AtualizeAutoVetores;
var
  i : 1..MaxRitz;
  aux,aux1,aux2,cx,cy:Real;
begin
  aux:=KR[j,j]-KR[k,k];
  if Abs(aux) < Tolerancia then
    begin
      cx:=1/Sqrt(2);
      cy:=cx;
    end
  else
    begin
      aux:=ArcTan(2*KR[j,k]/aux)/2;
      cx:=Cos(Aux);
      cy:=Sin(Aux);
    end;

  for i:=1 to Nv do { K O }
    begin
      aux1:=KR[i,j]*cx+KR[i,k]*cy;
      aux2:=KR[i,k]*cx-KR[i,j]*cy;
      KR[i,j]:=aux1;
      KR[i,k]:=aux2;
    end;

  for i:=1 to Nv do { Ot K O }
    begin
      aux1:=KR[j,i]*cx+KR[k,i]*cy;
      aux2:=KR[k,i]*cx-KR[j,i]*cy;
      KR[j,i]:=aux1;
      KR[k,i]:=aux2;
    end;

  for i:=1 to Nv do { autovetores }
    begin
      aux1:=AutoVetor[i,j]*cx+AutoVetor[i,k]*cy;
      aux2:=AutoVetor[i,k]*cx-AutoVetor[i,j]*cy;
      AutoVetor[i,j]:=aux1;
      AutoVetor[i,k]:=aux2;
    end;
  KR[j,k]:=KR[k,j];
end;

begin { Calcule autoVetores }
  iteracao:=0;
  repeat
    Convergiao:=True;

```

```

iteracao:=iteracao+1;
for i:=1 to Nv do Lambda[i]:=AutoValor[i];

for j:=1 to Nv-1 do
  for k:=j+1 to Nv do
    begin
      if KR[j,j] > KR[k,k] then Maior:=KR[j,j]
      else Maior:=KR[k,k];
      if Abs(KR[j,k]/Maior) > Tolerancia then
        AtualizeAutoVetores;
    end;

for i:=1 to Nv do { Atualize AutoValores }
  begin
    if KR[i,i] < 0 then ERRO;
    AutoValor[i]:=KR[i,i];
  end;

i:=1;
while Convergiu and ( i <= Nv ) do
  begin
    Convergiu:=Abs(AutoValor[i]-Lambda[i]) <
      Tolerancia ;
    i:=i+1;
  end;

j:=1;
while Convergiu and ( j < Nv ) do
  begin
    k:=j+1;
    while Convergiu and ( k <= Nv ) do
      begin
        Convergiu:=Sqr(KR[j,k])/(KR[j,j]*KR[k,k])
          < Tolerancia;
        k:=k+1;
      end;
    j:=j+1;
  end;
until Convergiu or (iteracao = MaxIteracoes);
end;

procedure NormalizeAutoVetores;
var
  i,j:1..MaxRitz;
  aux:Real;
begin { Normalize AutoVetores }
  for i:=1 to Nv do
    begin
      aux:=0;
      for j:=1 to Nv do aux:=aux+Sqr(AutoVetor[i,j]);
      aux:=Sqrt(aux);
      for j:=1 to Nv do
        AutoVetor[i,j]:=AutoVetor[i,j]/aux;
    end;
  end;

procedure DesacopleBaseDeRitz;
var
  ArqRitz : file of Real;
  i,j      :1..MaxRitz;
  k        :1..MaxDeslocamentos;

```

```

Aux      : Real;
VetorAux: VetorReal;

begin { Desacople Base De Ritz }
  Assign(ArqRitz,drive+':'+Nome+'.g'+
        Carregamento);
  Rewrite(ArqRitz);
  for i:=1 to Nv do
    begin
      for k:=1 to deslocamentos do VetorAux[k]:=0;
      for j:=1 to Nv do
        for k:=1 to deslocamentos do
          VetorAux[k]:=VetorAux[k]+PegueRitz(i,k)*
            AutoVetor[i,j];
        for k:=1 to deslocamentos do
          Write(ArqRitz,VetorAux[k]);
        end;
      Close(ArqRitz);
      DesaloqueBaseDeRitz;
      Assign(ArqRitz,drive+':'+Nome+'.Z'+
            Carregamento);
      Erase(ArqRitz);
      Assign(ArqRitz,drive+':'+Nome+'.g'+
            Carregamento);
      Rename(ArqRitz,drive+':'+Nome+'.Z'+
            Carregamento);
    end;
end;

procedure DesacopleVetorDeCargas;
var
  i:1..MaxRitz;
  j:1..MaxDeslocamentos;
  Deslocamento:VetorReal;
  ArqR:file of Real;
begin { Desacople Vetor de Cargas }
  LeiaVetorIndependente(Carregamento,Deslocamento);
  for i:=1 to Nv do
    begin
      f[i]:=0;
      for j:=1 to deslocamentos do
        f[i]:=f[i]+PegueRitz(i,j)*Deslocamento[j];
      end;
      Assign(ArqR,drive+':'+Nome+'.c'+
            Carregamento);
      Rewrite(ArqR);
      for i:=1 to Nv do Write(ArqR,f[i]);
      Close(ArqR);
    end;
end;

procedure GraveAutoValores;
var
  ArqR      : file of Real;
  i         : 1..MaxRitz;
begin
  Assign(ArqR,drive+':'+Nome+'.j'+
        Carregamento);
  Rewrite(ArqR);
  for i:=1 to Nv do Write(ArqR,AutoValor[i]);
  Close(ArqR);
end;

```

```

begin { Desacople Sistema Reduzido }
  InicializeAutoValores;
  InicializeAutoVetores;
  CalculeAutoVetores;
  NormalizeautoVetores;
  DesacopleBaseDeRitz;
  DesacopleVetorDeCargas;
  GraveAutoValores;
end;

begin { Monte Base De Ritz }
  deslocamentos:=Nos*DeslocamentosPorNo;
  MonteSistemaReduzido;
  DesacopleSistemaReduzido;
  DesaloqueBaseDeRitz;
end;

procedure CalculeParametrosResposta;
var
  Intervalo,gt,gt1,deltaT : Real;
  ArqA,ArqV,ArqD,ArqT : file of Real;
  a0,a1,a2,a3,a4,w,w1,a,v,d: TipoRitz;

procedure InicializeVariaveis;
var
  i : 1..MaxRitz;
  aux,expoente:Real;

begin { InicializeVariaveis }
  aux:=Sqrt(1-Sqr(Amortecimento));
  for i:=1 to Nv do
    begin
      d[i]:=0;
      v[i]:=0;
      a[i]:=0;
      w[i]:=Sqrt(AutoValor[i]);
      w1[i]:=w[i]*Aux;
      a0[i]:=0;
      a1[i]:=0;
      a2[i]:=0;
      a3[i]:=0;
      expoente:=-Amortecimento*w[i]*deltaT;
      if expoente < -88 then a4[i]:=0
        else a4[i]:=exp(expoente);
    end;
  end;

procedure CalculeParametros;
var
  i : 1..MaxRitz;
  aux1,aux2 : Real;
begin { Calcule Parametros }
  for i:=1 to Nv do
    begin
      aux1:=gt*f[i];
      aux2:=(gt1*f[i]-aux1)/deltaT;
      a0[i]:=(aux1-2*aux2*
        Amortecimento/w[i])/AutoValor[i];
      a1[i]:=aux2/AutoValor[i];
      a2[i]:=d[i]-a0[i];
      a3[i]:=(v[i]+a2[i]*w[i]*

```

```

                                Amortecimento-a1[i])/w1[i];
    end;
end;

procedure CalculeAceleracoes;
var
    i:1..MaxRitz;
    aux1,aux2:Real;
begin { Calcule Aceleracoes }
    for i:=1 to Nv do
        begin
            aux1:=a2[i]*(Sqr(Amortecimento*w[i])-w1[i]*w1[i])
                -2*a3[i]*Amortecimento*w[i]*w1[i];
            aux2:=a3[i]*(Sqr(Amortecimento*w[i])+w1[i]*w1[i])
                -2*a2[i]*Amortecimento*w[i]*w1[i];
            a[i]:=a4[i]*(aux1*Cos(w1[i]*deltaT)-
                aux2*Sin(w1[i]*deltaT));
            Write(ArqA,a[i]);
        end;
    end;

procedure CalculeVelocidades;
var
    i:1..MaxRitz;
    aux1,aux2:Real;
begin { Calcule Velocidades }
    for i:=1 to Nv do
        begin
            aux1:=w1[i]*a3[i]-Amortecimento*w[i]*a2[i];
            aux2:=w1[i]*a2[i]+Amortecimento*w[i]*a3[i];
            v[i]:=a1[i]+a4[i]*(aux1*Cos(w1[i]*deltaT)-
                aux2*Sin(w1[i]*deltaT));
            Write(ArqV,v[i]);
        end;
    end;

procedure CalculeDeslocamentos;
var
    i:1..MaxRitz;
begin { Calcule Deslocamentos }
    for i:=1 to Nv do
        begin
            d[i]:=a0[i]+a1[i]*deltaT+a4[i]*(a2[i]*
                Cos(w1[i]*deltaT)+a3[i]*Sin(w1[i]*deltaT));
            Write(ArqD,d[i]);
        end;
    end;

begin { Resolva Historia Discreta }
    Assign(ArqA,drive+' '+Nome+'.'+' '+Carregamento);
    Rewrite(ArqA);

    Assign(ArqV,drive+' '+Nome+'.'+'a'+Carregamento);
    Rewrite(ArqV);

    Assign(ArqD,drive+' '+Nome+'.'+'b'+Carregamento);
    Rewrite(ArqD);

    Assign(ArqT,drive+' '+Nome+'.'+'T'+Carregamento);
    Reset(ArqT);

```

```

intervalo:=0;
Read(ArqT,deltaT,gt1);
InicializeVariaveis;
while not EOF(ArqT) do
  begin
    intervalo:=intervalo+DeltaT;
    gt:=gt1;
    Read(ArqT,gt1);
    CalculeParametros;
    CalculeAceleracoes;
    CalculeVelocidades;
    CalculeDeslocamentos;
  end;
Close(ArqT);
Close(ArqA);
Close(ArqV);
Close(ArqD);
end;

```

```

procedure CalculeResposta;

```

```

var
  ArqAg,ArqVg,ArqDg,
  ArqA,ArqV,ArqD,ArqT:file of Real;
  ArqR           :file of TipoReacao;
  ArqE           :file of TipoEsforcoExtr;
  deslocamentos :integer;
  DeltaT,Amplitude,intervalo :Real;
  deslocamento  :VetorReal;

```

```

procedure AbraArquivos;

```

```

begin
  Assign(ArqAg,drive+' '+Nome+'.'+'+Carregamento);
  Reset (ArqAg);
  Assign(ArqVg,drive+' '+Nome+'.'+'a'+Carregamento);
  Reset (ArqVg);
  Assign(ArqDg,drive+' '+Nome+'.'+'b'+Carregamento);
  Reset (ArqDg);
  Assign(ArqA,drive+' '+Nome+'.'+'A'+Carregamento);
  Rewrite(ArqA);
  Assign(ArqV,drive+' '+Nome+'.'+'V'+Carregamento);
  Rewrite(ArqV);
  Assign(ArqD,drive+' '+Nome+'.'+'D'+Carregamento);
  Rewrite(ArqD);
  Assign(ArqR,drive+' '+Nome+'.'+'R'+Carregamento);
  Rewrite(ArqR);
  Assign(ArqE,drive+' '+Nome+'.'+'E'+Carregamento);
  Rewrite(ArqE);
  Assign(ArqT,drive+' '+Nome+'.'+'T'+Carregamento);
  Reset (ArqT);
end;

```

```

procedure MudeReferencial(var d:VetorReal;var u:TipoRitz);

```

```

var
  i:1..MaxDeslocamentos;
  j:1..MaxRitz;
begin { Mude Referencial }
  for j:=1 to Nv do
    for i:=1 to deslocamentos do
      d[i]:=d[i]+PegueRitz(j,i)*u[j];
    end;
end;

```

```
procedure ObtenhaAceleracoes:
```

```
var
  i:1..MaxDeslocamentos;
  j:1..MaxRitz;
  u:TipoRitz;
  A:VetorReal;
begin { Obtenha Aceleracoes }
  for j:=1 to Nv do Read(ArqAg,u[j]);
  for i:=1 to deslocamentos do A[i]:=0;
  MudeReferencial(A,u);
  for i:=1 to deslocamentos do Write(ArqA,A[i]);
end;
```

```
procedure ObtenhaVelocidades:
```

```
var
  i:1..MaxDeslocamentos;
  j:1..MaxRitz;
  u:TipoRitz;
  V:VetorReal;
begin { Obtenha Velocidades }
  for j:=1 to Nv do Read(ArqVg,u[j]);
  for i:=1 to deslocamentos do V[i]:=0;
  MudeReferencial(V,u);
  for i:=1 to deslocamentos do Write(ArqV,V[i]);
end;
```

```
procedure ObtenhaDeslocamentos:
```

```
var
  i:1..MaxDeslocamentos;
  j:1..MaxRitz;
  u:TipoRitz;
begin { Obtenha Deslocamentos }
  for j:=1 to Nv do Read(ArqDg,u[j]);
  for i:=1 to deslocamentos do deslocamento[i]:=0;
  MudeReferencial(deslocamento,u);
  for i:=1 to deslocamentos do
    Write(ArqD,deslocamento[i]);
end;
```

```
procedure ObtenhaReacoes:
```

```
var
  Reacao      :TipoReacao;
  direcao     :1..6;
  posicao      :Integer;
  Apoio       :TipoApoio;
begin
  Assign(ArqApoio,drive+' '+Nome+'.005');
  Reset(ArqApoio);

  while not EOF(ArqApoio) do
    begin
      Read(ArqApoio,Apoio);
      Reacao.NumeroNo:=Apoio.NumeroNo;
      posicao:=(Apoio.NumeroNo-1)*DeslocamentosporNo;
      for direcao:=1 to DeslocamentosporNo do
        if Apoio.Codigo[direcao] = '1'
          then Reacao.Valor[direcao]:=
            -Deslocamento[posicao+direcao]*KApoio
          else Reacao.Valor[direcao]:=
            -Deslocamento[posicao+direcao]*
              Apoio.Condicao[direcao];
```

```

    Write(ArqR,Reacao);
end;
Close(ArqApoio);
end;

```

```

procedure ObtenhaEsforçosDeExtremidade;

```

```

var
  EsforçoExtr      :TipoEsforçoExtr;
  ArqF              :file of TipoEsforçoExtr;
  Elemento, posicao:Integer;
  DesiNo, DesiBarra:1..12;

```

```

procedure CalculeEsforçosDaBarra;

```

```

var
  Barra              :TipoBarra;
  Ke                 :Matriz12;
  f,d                :Vetor12;
  direcao,linha,coluna:1..12;

```

```

procedure PreMultipliqueMatrizDeRigidez;

```

```

var
  Aux                :Real;
  MaxM,MaxI:1..4;
  i,j,k,m,n:1..12;
  R,S                :Matriz3;
begin
  case Controle.TipoEstrutura of
    PorticoEspacial:begin
                        MaxM:=4;
                        MaxI:=3;
                      end;
    PorticoPlano     :begin
                        MaxM:=2;
                        MaxI:=3;
                      end;
    TrelicaEspacial:begin
                        MaxM:=2;
                        MaxI:=3;
                      end;
    TrelicaPlana     :begin
                        MaxM:=2;
                        MaxI:=2;
                      end;
    Greiha           :begin
                        MaxM:=2;
                        MaxI:=3;
                      end;
  end;
  ObtenhaMatrizDeRotacao(R,Barra);
  for m:=1 to MaxM do
    for n:=1 to MaxM do
      begin
        for i:=1 to MaxI do
          for j:= 1 to MaxI do S[i,j]:=0;
        for i:=1 to MaxI do
          for j:=1 to MaxI do
            begin
              Aux:=0;
              for k:= 1 to MaxI do
                Aux:=Aux+R[k,j]*
                  Ke[MaxI*(m-1)+i,MaxI*(n-1)+k];
            end;
          end;
        end;
      end;
    end;
  end;
end;

```

```

        S[i,j]:=Aux;
    end;
    for i:=1 to Maxi do
        for j:=1 to Maxi do
            Ke[Maxi*(m-1)+i,Maxi*(n-1)+j]:=S[i,j];
        end;
    end;
end;

procedure AtualizeEsforco;
begin
    Read(ArqF,EsforcoExtr);
    with EsforcoExtr do
        for direcao:=1 to 12 do
            Valor[direcao]:=Amplitude*Valor[direcao];
        end;
    with EsforcoExtr do
        case Controle.TipoEstrutura of
            PorticoEspacial:
                for direcao:=1 to 12 do
                    Valor[direcao]:=-Valor[direcao]+f[direcao];
                end;
            PorticoPlano:
                begin
                    Valor[ 1]:=-Valor[ 1]+f[ 1];
                    Valor[ 2]:=-Valor[ 2]+f[ 2];
                    Valor[ 3]:=-Valor[ 6]+f[ 3];
                    Valor[ 4]:=-Valor[ 7]+f[ 4];
                    Valor[ 5]:=-Valor[ 8]+f[ 5];
                    Valor[ 6]:=-Valor[12]+f[ 6];
                end;
            TrelicaEspacial:
                begin
                    Valor[ 1]:=-Valor[ 1]+f[ 1];
                    Valor[ 2]:=-Valor[ 2]+f[ 2];
                    Valor[ 3]:=-Valor[ 3]+f[ 3];
                    Valor[ 4]:=-Valor[ 7]+f[ 4];
                    Valor[ 5]:=-Valor[ 8]+f[ 5];
                    Valor[ 6]:=-Valor[ 9]+f[ 6];
                end;
            TrelicaPlana:
                begin
                    Valor[ 1]:=-Valor[ 1]+f[ 1];
                    Valor[ 2]:=-Valor[ 2]+f[ 2];
                    Valor[ 3]:=-Valor[ 7]+f[ 3];
                    Valor[ 4]:=-Valor[ 8]+f[ 4];
                end;
            Grelha:
                begin
                    Valor[ 1]:=-Valor[ 4]+f[ 1];
                    Valor[ 2]:=-Valor[ 2]+f[ 2];
                    Valor[ 3]:=-Valor[ 6]+f[ 3];
                    Valor[ 4]:=-Valor[10]+f[ 4];
                    Valor[ 5]:=-Valor[ 8]+f[ 5];
                    Valor[ 6]:=-Valor[12]+f[ 6];
                end;
        end;
    end;
    Write(ArqE,EsforcoExtr);
end;

begin { CalculeEsforcosDaBarra }
    Elemento:=Elemento+1;
    Read(ArqBarra,Barra);
end;

```

```

case Controle.TipoEstrutura of
  Grelha           :ObtenhaKG (Ke,Barra);
  PorticoEspacial :ObtenhaKPE(Ke,Barra);
  PorticoPlano     :ObtenhaKPP(Ke,Barra);
  TrelicaEspacial :ObtenhaKTE(Ke,Barra);
  TrelicaPlana    :ObtenhaKTP(Ke,Barra);
end;

if Barra.TemLiberacao then
  LibereMatrizDeRigidez(Ke,Barra);
PreMultipliqueMatrizDeRigidez;

for direcao:=1 to DeslNo do
  begin
    d[direcao]           :=deslocamento[(Barra.No[1]-1)*
                                          DeslNo+direcao];
    d[direcao+DeslNo] :=deslocamento[(Barra.No[2]-1)*
                                          DeslNo+direcao];
  end;

  for direcao:=1 to DeslBarra do f[direcao]:=0;

  for linha:=1 to DeslBarra do
    for coluna:=1 to DeslBarra do
      f[linha]:=f[linha]+Ke[linha,coluna]*d[coluna];

  AtualizeEsforco;
end;

begin { Obtenha Esforços De Extremidade }
  DeslNo:=DeslocamentosporNo;
  DeslBarra:=DeslNo*NosporElemento;

  Assign(ArqF,drive+' '+Nome+'.N'+Carregamento);
  Reset(ArqF);

  Assign(ArqBarra,drive+' '+Nome+'.004');
  Reset(ArqBarra);

  Elemento:=0;
  while not EOF(ArqBarra) do CalculeEsforçosDaBarra;

  Close(ArqBarra);
  Close(ArqF);
end;

procedure FechaArquivos;
begin
  Close(ArqAg);
  Close(ArqVg);
  Close(ArqDg);
  Close(ArqA);
  Close(ArqV);
  Close(ArqD);
  Close(ArqR);
  Close(ArqE);
  Close(ArqT);
end;

begin { Calcule Resposta }

```

```

deslocamentos:=Nos*DeslocamentosPorNo;
LeiaMateriais;
LeiaSecoes;
LeiaNos;
AbraArquivos;
Read(ArqT,DeltaT);
intervalo:=0;
while Not (EOF(ArqT ) or
           EOF(ArqAg) or
           EOF(ArqVg) or
           EOF(ArqDg))do
  begin
    Read(ArqT,Amplitude);
    ObtenhaAceleracoes;
    ObtenhaVelocidades;
    ObtenhaDeslocamentos;
    ObtenhaReacoes;
    ObtenhaEsforçosDeExtremidade;
    intervalo:=intervalo+DeltaT;
  end;
  FecheArquivos;
end;

begin { Obtenha Resposta }
  LeiaNv;
  for Carregamento:=1 to Carregamentos do
    begin
      MonteBaseDeRitz;
      CalculeParametrosResposta;
      CalculeResposta;
    end;
  end;

begin { Execute Analise Dinamica }
  PrepareSistema;
  ObtenhaResposta;
end;

begin { Analise Estrutura }
  MonteVetoresIndependentes;
  ExecuteAnaliseDinamica;
end;

```