Escola Politécnica da Universidade de São Paulo

Departamento de Engenharia Mecânica

PARAMETRIZAÇÃO:

A automatização do processo de construção do modelo para cálculo por elementos finitos

Trabalho Aula 1

Aluno: João Claudio Cotta Cardoso - Nº USP 2812819

Prof. Orientador: Roberto Ramos Jr.

Data de entrega: 10/12/97

Índice

Prefácio

Este trabalho é totalmente voltado para parametrização das partes estruturais de hidrogeradores. É um tema duplamente interessante nos dias de hoje, visto que por uma lado a parametrização é uma técnica de engenharia bastante inovadora, e por outro lado, que a crescente demanda energética no Brasil promete movimentar (e modernizar) bastante o mercado de hidrogeradores.

Do lado do desenvolvimento pessoal, este trabalho foi muito importante. O método dos elementos finitos, e os softwares especializados nele, eram praticamente uma incógnita para mim quando do início deste trabalho. Ao longo do ano, o desenvolvimento alcançado superou minhas expectativas.

Por fim, gostaria de ressaltar que este trabalho recorre pouco a referências externas: num tema que é relativamente inexplorado, ainda mais aplicado a hidrogeradores e a elementos finitos, ter como referência a experiência e o conhecimento dos engenheiros da Asea Brown Boveri e de meu orientador, Roberto Ramos Jr. Além, é claro, do manual do ANSYS.

1. Introdução

A parametrização é um artifício utilizado para se minimizar o tempo gasto na utilização de computadores para o projeto de máquinas. Para se entender o conceito da parametrização, e o porquê de sua aplicação para os cálculos estruturais de hidrogeradores, devemos observar todo o contexto em que estamos inseridos, ou seja: as condições do mercado do setor de geração de energia no Brasil atualmente; os avanços tecnológicos introduzidos no produto; e as tendências mundiais que levam as empresas a buscar, via padronização, a minimização dos gastos com engenharia de projeto.

Portanto, a primeira parte deste trabalho trará antes a descrição desse contexto, para depois caracterizar o problema enfrentado nas empresas do setor de geração de energia e mostrar que a parametrização é uma solução viável e necessária.

1.1. O Hidrogerador

O hidrogerador é uma máquina elétrica, geralmente de grande porte, que tem como função transformar a energia potencial das águas dos rios em energia elétrica. Seu princípio de funcionamento é inverso ao do motor elétrico: a vazão de água que passa empurra as pás da turbina que, ligada ao eixo da máquina, coloca todo o rotor da máquina em movimento. Esse rotor, dotado de todo o aparato eletromagnético necessário e devidamente excitado eletricamente, cria um campo magnético girante que induz uma tensão nas bobinas do estator. Essa é a energia elétrica gerada e que será utilizada pela população.

As dimensões de um hidrogerador variam dentro de uma faixa bem larga de valores. As chamadas *pequenas centrais hidroelétricas* (PCH's) são utilizadas para aproveitar pequenas alturas de queda e pequenas vazões de água, gerando potências elétricas às vezes menores que 10 MW. Já as grandes usinas hidroelétricas podem possuir dezenas de hidrogeradores e suprir as necessidades de vastas regiões. A Hidroelétrica de Itaipu, por exemplo, conta com 18 (dezoito)

geradores, cada um gerando potências da ordem de 770 MW. A metade dessa energia, que pertence ao Paraguai, excede as necessidades energéticas daquele país, que vende seu excedente de volta para o Brasil.

De toda a energia elétrica produzida no Brasil, mais de 70% provém das usinas hidroelétricas. Na década de 90, com a retomada do crescimento econômico, a demanda de energia no país tem aumentado e a construção de mais usinas tende a se intensificar. Como, no entanto, os maiores potenciais hidroelétricos já foram aproveitados, não há possibilidades de se construir no Brasil outras usinas do porte de Itaipu e Tucuruí. As novas hidroelétricas gerarão potências menores e certamente vão incorporar soluções diferentes para se tornarem viáveis (as máquinas do tipo Bulbo, já bem comuns na Europa e feitas especialmente para menores aproveitamentos, estão começando a chegar ao Brasil).

Não é difícil notar que, em face do porte dos geradores hidroelétricos, eles sejam feitos cada um "sob medida", respeitando as peculiaridades hidráulicas, ambientais e sociais da região onde será instalado. Desse modo, a máquina teria que ser projetada "desde o início" toda vez que um contrato de construção de uma usina hidroelétrica fosse fechado. Na verdade, as empresas utilizam sua experiência anterior no projeto e na fabricação de novas máquinas, e é por isso que as principais concorrentes desse mercado estão nele já a várias décadas.

1.2.Tendências do mercado brasileiro e mundial de geração de energia

A década de 90, principalmente no Brasil, está trazendo grandes mudanças do ambiente de competição entre as empresas na área de geração de energia. Essas mudanças são devidas principalmente a dois importantes fatores de ordem política: A abertura do mercado brasileiro e o programa nacional de privatizações.

A trajetória política do país nos últimos anos tem sido no sentido de se tornar uma nação capitalista e mais inserida no mercado internacional. Assim, o mercado brasileiro, antes fechado às empresas locais, foi sendo gradativamente

aberto à concorrência de empresas estrangeiras. No mercado de geração de energia essa alteração no panorama foi ainda mais sentido, pois as poucas empresas daqui dividiam o mercado entre si, garantindo altas margens de lucro.

Além disso, os clientes dessas empresas, que são as empresas energéticas dos estados (CESP, CEMIG, Light, CPFL, etc.), tendem a passar das mãos do estado às da iniciativa privada. Assim, não há como se fazer *lobbies* políticos nas concorrências para a construção das hidroelétricas, pois o cliente sempre estará visando o bom desempenho de *sua* empresa, e seus fornecedores é que devem se esforçar para baixar seus preços.

Diante dessas novas condições, o que se tem visto no setor é que os preços dos hidrogeradores (e, conseqüentemente, as margens de lucro das empresas que os fabricam) estão caindo continuamente. O preço, que antigamente era determinado pela empresa através da soma de seus custos mais seu lucro, agora é ditado pelos clientes. Assim, se uma empresa quiser aumentar seus lucros, a única maneira é reduzir os custos.

É exatamente isso que tem acontecido nas empresas brasileiras do setor nos últimos anos. Esses dois fatores, mais a recessão do início da década, as forçaram a enxugar seus quadros de funcionários, diminuir as horas de engenharia gastas em projeto, diminuir os gastos com material, evitar a má qualidade (que pode ser traduzida também em custos). Tudo isso enquanto eram obrigadas a se modernizar tecnologicamente, devido à pressão da concorrência externa.

1.3. A padronização como tendência global

Uma das frentes atacadas na redução de custos, como já foi mencionado, é a redução das horas de engenharia gastas em cada projeto. Uma das principais estratégias utilizadas para conseguir esse objetivo é a chamada *padronização*: Fazer as coisas de uma maneira pré-determinada (ou padronizada) para que se perca menos tempo personalizando componentes dessa ou daquela máquina. Assim, a tendência é que as peças e sistemas da máquina sejam

projetadas para atender as necessidades daquela encomenda em particular, mas respeitando (e se baseando em) um *padrão* existente na empresa.

As pequenas centrais hidroelétricas (PCH's) já alcançaram um nível bastante elevado de padronização e os gastos com engenharia nessas máquinas foram bastante reduzidos. As máquinas de maior porte, porém, são mais difíceis de serem padronizadas, já que decisões tomadas na fase de projeto baseadas em qualquer característica especial da máquina podem refletir em economias altas ou prejuízos também altos. É preciso que se façam estudos mais detalhados (e se use a experiência das PCH's) para atingir a padronização das grandes centrais.

A padronização é uma tendência mundial, já que a forte concorrência está presente em quase todos os mercados do mundo. Pode-se concluir daí que, no brasil, as multinacionais do setor já largam com certa vantagem sobre os concorrentes no que diz respeito à padronização. Elas podem se valer de trabalhos desenvolvidos pelas matrizes ou outras filiais de seu próprio grupo.

1.4. A análise estrutural por elementos finitos

Dentre as novas realidades no projeto de máquinas e, em particular, no projeto de hidrogeradores, está a utilização de softwares de elementos finitos. Esses programas auxiliam o engenheiro nas análises e cálculos, proporcionando resultados mais precisos e economizando tempo de cálculo. Esses programas até possibilitam análises que eram inviáveis antes, devido à sua complexidade.

A utilização de softwares de elementos finitos, além de proporcionar certa redução de horas de engenharia, também entra no corte de custos de uma empresa por outro lado: com base em seus resultados mais precisos, pode-se fazer estruturas mais esbeltas, abrindo mão de altos coeficientes de segurança que eram utilizados para embutir incertezas. Assim, gastos com material também são diminuídos.

É claro que se deve aqui fazer uma ressalva. A utilização de programas desse tipo permite a redução de custos e melhorias no projeto, *desde que utilizado com método*. Devem-se tomar cuidados extras na verificação das hipóteses

aceitas e na construção do modelo que simulará a estrutura nos cálculos. Uma imperfeição nos dados que se entra no programa pode ocasionar uma saída ruim, com resultados ilusórios que não se verificarão quando a estrutura real for construída.

Além disso, apesar de cortar custos por um lado, a utilização desses softwares tem um custo alto de implantação. Programas de elementos finitos são bastante extensos e complexos, portanto são muito caros. Além disso, exigem a compra de computadores potentes e tornam necessário o treinamento do pessoal que irá utilizá-los.

Feitas as ressalvas, vê-se que ainda é um avanço tecnológico muito grande no sentido de se melhorar o produto e aumentar eficiência e produtividade da engenharia das empresas (não só no campo da geração de energia, mas toda firma que realiza projetos de engenharia pode se beneficiar desses sistemas). Ademais, em um ambiente de concorrência pesada como o enfrentado hoje, é vital *ter* e *saber usar* softwares de elementos finitos (pois os concorrentes os têm e sabem usá-los...).

2. Definição do Problema

A utilização dos programas de elementos finitos nos escritórios de engenharia das empresas que fornecem hidrogeradores certamente incorporou grandes ganhos de produtividade e qualidade aos produtos por elas fabricados. Porém, a corrida tecnológica continua e, se implementar sistemas de cálculo por elementos finitos foi uma etapa dessa corrida, a racionalização e otimização do uso desses programas é o próximo passo, e também é bastante complexo.

Além disso, com a cultura de uso de programas de elementos finitos já introduzida em seu dia-a-dia, os engenheiros começam a se perguntar como aproveitar esses programas ao máximo, ou seja, quais os outros usos que podem ser feitos desses programas? Em que outras tarefas, além da verificação da validade de um modelo, como posso utilizar a capacidade de processamento que eles têm?

Temos, assim, problemas e metas advindas do uso de softwares de cálculo por elementos finitos. Os três itens a seguir resumem, de certas formas, o espírito dentro do qual a parametrização se apresenta como melhor solução viável.

2.1. Tempo demandado para a construção dos modelos

No âmbito da engenharia mecânica dos hidrogeradores, os programas de elementos finitos são utilizados para a verificação da parte estrutural da máquina. As principais partes do hidrogerador com função estrutural são: Carcaça, Cruzeta Superior, Cruzeta Inferior e Aranha do Rotor. As cruzetas têm como função suportar o eixo da máquina, abrigando os mancais e o sistema de frenagem da máquina. A carcaça abriga todo o estator da máquina (bobinas e parte magnética), além do sistema de resfriamento (trocadores de calor, tubulações de água e óleo).

Nos memoriais de cálculo que versam sobre essas partes da máquina, sempre estão incluídas as análises feitas através de elementos finitos, mostrando que a estrutura da máquina irá suportar as condições mais adversas de funcionamento previstas no contrato.

O problema que advém daí está na construção do modelo matemático (geometria + apoios + carregamento) que será usado para efetuar o cálculo dentro do programa de elementos finitos. O *tempo* que se gasta nesse processo é muito grande (da ordem de dias). A interface desses programas com o usuário para a confecção dos modelos geralmente não é tão boa, o que atrasa a criação da geometria.

Além disso, para se obter uma boa *malha* (os elementos finitos propriamente ditos que comporão o modelo) devem se tomar cuidados excessivos. A forma dos elementos é vital para a obtenção de bons resultados: elementos com formato inadequado (triangulares, com lados muito grandes ou ângulos muito agudos) geralmente comprometem a confiabilidade da solução. A quantidade de providências que têm de ser tomadas para evitar a formação de maus elementos na malha é relativamente grande, e ocupam muito tempo do usuário.

Importar desenhos de outros softwares (do tipo CAD) também não costuma ser uma boa alternativa por dois motivos: Primeiro, a compatibilidade entre os sistemas CAD e os de elementos finitos nunca é total. Sempre haveriam distorções e erros a serem corrigidos depois da importação, e isso às vezes poderia levar mais tempo do que a própria implementação direta no programa de cálculo. Segundo, os desenhos confeccionados nos sistemas CAD geralmente são desenhos básicos ou de detalhe, e contém partes da geometria que não têm relevância do ponto de vista estrutural. O modelo para cálculo estrutural é *sempre* mais simples que o desenho de fabricação da peça real, e a importação deveria contar com um trabalho posterior de edição e remoção de detalhes que também ocuparia um tempo muito grande.

Visto isso, temos caracterizado um problema: um engenheiro pode perder dias de seu trabalho realizando um serviço que na verdade *não é* de engenheiro, que é a introdução do modelo no computador. O engenheiro deve propor soluções a problemas e analisar resultados.

2.2. Dificuldade para alterações no modelo depois de pronto

Correção de erros cometidos na fase de construção do modelo. Alteração de alguma característica do modelo depois de ter tido uma solução não satisfatória. Verificação de geometrias alternativas, na tentativa de otimizar o projeto. São ocorrências comuns na utilização de programas de elementos finitos. Deveriam ser tarefas fáceis e rápidas. Infelizmente não são.

Alteração de uma geometria já feita geralmente requer um tempo bastante grande. Deve-se desfazer a malha, apagar as partes da geometria que se quer corrigir, refazer essas mesmas partes da maneira que se acha conveniente, tomar novamente as providências necessárias para a obtenção de uma boa malha, gerar a malha, e só aí pode-se rodar a solução novamente.

Algumas vezes esse processo dura até mais do que a obtenção da geometria desde o início, e mais uma vez temos um engenheiro usando suas horas com tarefas que não são, por assim dizer, tão nobres.

2.3. Utilização do software abaixo de suas possibilidades

Outro fato que se verifica no dia-a-dia dos usuários de uma ferramenta de cálculo desse tipo é que, devido a essas grandes perdas de tempo na construção do modelo e na mudança de suas características geométricas depois de obtida uma solução, os programas de cálculo por elementos finitos acabam sendo utilizados apenas para verificação de estruturas com a geometria e as medidas pré-definidas.

Explicando melhor: o engenheiro usa o software *depois* de já ter projetado a estrutura (ou peça) em questão. Ele recorre ao método dos elementos finitos apenas para se assegurar de que seu projeto atende às especificações necessárias.

Acontece que, pela sua capacidade e rapidez de processamento e pela precisão de seus resultados, esses programas deveriam ser utilizados para mais do que apenas verificação. Deveriam eles servir ao engenheiro como ferramenta de *simulação*. Os programas podem ser usados para se testar soluções diferentes, verificar geometrias alternativas. Enfim, pode-se, através deles, realmente *otimizar* o projeto.

3. Parametrização: Conceituação e métodos

Analisando os três itens que foram apresentados como o *problema*, podemos chegar à conclusão de que eles têm, no fundo, a mesma causa: o tempo de processamento do programa é mau utilizado.

Um software de elementos finitos tem seu tempo de utilização dividido em três etapas:

- **Entrada de dados (Pré-processamento):** aqui é feita a confecção do modelo: construção da geometria, definição das restrições, carregamentos, apoios e condições de contorno e obtenção da malha de elementos;

- **Processamento:** é a análise, pelo programa, dos dados entrados pelo usuário, para determinar quais serão as condições do modelo, aplicadas as condições de contorno dadas;

- **Análise da solução(Pós-processamento):** é a fase em que o programa apresenta as condições finais do modelo ao usuário (geometria deformada, tensões, deformações, temperaturas e o que mais for necessário para descrever o estado do modelo), frente às condições impostas.

Podemos aí reconhecer em que fase é o usuário quem trabalha e em que fase é o computador quem trabalha. A parte de processamento é aquela em que toda a capacidade de processamento do software está sendo utilizada. Nas outras duas (principalmente na entrada dos dados), no entanto, o computador apenas auxilia o usuário a introduzir o modelo, e espera as suas tomadas de decisão.

Um exemplo ilustrativo é o seguinte: criar uma linha. Enquanto o usuário planeja sua linha (determinando seu comprimento, sua inclinação, e às vezes até fazendo alguns cálculo para determinar sua posição exata), o computador está ocioso, apenas aguardando ordens do usuário. Uma vez que o comando para gerar a linha é dado, o programa a executa em uma fração de segundo e está novamente na espera para um novo comando.

Quando se gasta uma manhã, ou um dia, ou uma semana na construção de um modelo dentro do software, na verdade está se jogando fora um

precioso tempo *de processamento* da máquina. Tempo esse que, se fosse utilizado, permitiria o uso do software como ferramenta de simulação e verificação de alternativa, como era do nosso desejo.

A conclusão a que se chega, portanto, é que o caminho para resolver os problemas apresentados anteriormente é reduzir ao máximo o tempo que o usuário interage com o programa, para que este passe o maior tempo possível utilizando sua plena capacidade de processamento.

3.1. O Conceito de Parametrização

O nome *parametrização* vem de parâmetro, e é exatamente essa a idéia. A meta é aliviar o usuário do trabalho de ter que construir o modelo, transferindo essa tarefa ao próprio computador. Para isso, deve-se encontrar no modelo os *parâmetros-chave* que determinam as características do modelo, e *ensinar* ao computador como construí-lo a partir desses parâmetros.

A escolha dos parâmetros-chave deve ser muito bem feita, pois em função deles o programa determinará todas as outras medidas da geometria e carregamento do modelo. Quanto menor for a quantidade de parâmetros que devem ser entrados pelo usuário, maior agilidade terá o programa. Porém, um número muito pequeno de parâmetros tira a liberdade de se variar as características do modelo.

A construção do modelo pelo computador também deve ser alvo de cuidados especiais, por motivos óbvios. Deve-se estabelecer faixas de variação dos valores entrados como parâmetros, para que a geometria seja possível de ser obtida e corresponda ao esperado pelo usuário. Deve-se fazer testes preliminares antes de colocar a parametrização em uso, para que nos certifiquemos de que o computador está realmente fazendo o que nós faríamos se estivéssemos construindo o modelo em seu lugar.

3.2. Avaliação das melhorias com a implementação da parametrização

Não é difícil imaginar que, depois de implementada, a parametrização traz um ganho de produtividade altíssimo aos engenheiros de uma empresa do ramo de construção de hidrogeradores. É só observar que todos os dias gastos para introduzir os modelos para cálculo nos programas de elementos finitos estará reduzido a quase nada. Assim, quase todo o tempo o computador estará operando com sua capacidade máxima de processamento.

É claro que, antes de entrar os parâmetros no programa, o usuário deve ter uma idéia, ao menos preliminar, de quais serão os valores desses parâmetros. Mas isso não traz grandes diferenças em relação ao processo anterior, pois neste o usuário também já deveria ter uma geometria pré-determinada antes de se utilizar dos softwares de cálculo.

Os problemas apresentados estarão solucionados, pois:

- O tempo demandado para a criação do modelo dentro do programa foi reduzido a níveis ínfimos. O computador "saberá", a partir dos parâmetros entrados pelo usuário, como construir a geometria adequada, como obter uma malha de elementos de boa qualidade e como distribuir os carregamentos e as condições de contorno;

- A correção de erros e modificação do modelo após a obtenção de uma solução será, agora, rápida e fácil. Isso porque, como é o computador que constrói o modelo (e faz isso rapidamente), para efetuar modificações basta apagar tudo e fazer o modelo de novo. Isso sem contar o fato de que, com a participação do usuário reduzida drasticamente, a incidência de erros nos dados de entrada também será reduzida, na mesma proporção.

- A utilização do software em sua máxima capacidade também é possível agora. Obter os modelos e as soluções fica muito mais rápido, portanto temos mais tempo livre para testar geometrias alternativas, alterar parâmetros para verificar a sensibilidade do modelo a cada parâmetro, tentar utilizar um conjunto de parâmetros que possibilitem que a estrutura real seja construída com

menor custo. Enfim, agora o software é uma *ferramenta de simulação* e auxílio para a tomada de decisões do engenheiro.

3.3. A padronização como pré-requisito da parametrização

Se a parametrização envolve a escolha de parâmetros que sirvam para descrever o modelo, e o "ensinamento" ao computador de como fazer para obter esse modelo através dos parâmetros dados como entrada, é claro que se deve já saber de antemão como será a geometria do modelo em função dos parâmetros. Em outras palavras, é necessário que se utilize uma geometria *padrão* para as peças com a mesma função dentro da máquina.

Portanto, antes que se desenvolva a parametrização, é preciso que a própria padronização dos componentes já esteja adiantada, de modo a fazer com que o esforço empregado para se conseguir a parametrização produza resultados que possam ser utilizados no maior número de projetos possível.

Como já foi dito anteriormente, no caso dos hidrogeradores, a padronização das máquinas de pequeno porte já alcançou níveis consideráveis, e a das de maior porte está avançando. Quanto mais baixo o nível de padronização de um determinado componente (ou seja, mais personalizado seja o seu projeto para cada máquina), mais flexível (e portanto de mais difícil implementação) terá de ser sua parametrização para elementos finitos. E como conseqüência aumenta o tempo gasto pelo usuário para determinação e entrada dos parâmetros, e aumenta também a probabilidade da ocorrência de erros.

3.4. Método de implementação da parametrização

Foi muito falado anteriormente que um dos passos da parametrização é *ensinar* ao computador como se constrói o modelo dados os parâmetros-chave como entrada. Será mostrado agora como se "ensina" algo ao computador.

Linguagens de programação são usadas quando se quer que o computador execute uma tarefa específica. Existem várias linguagens de programação disponíveis hoje no mercado, para diversos tipos de aplicação e ambiente operacional. Os grandes aplicativos, hoje em dia, também trazem incorporada uma linguagem de programação interna, para que se possa automatizar tarefas repetitivas ou que tomam muito esforço do usuário dentro dos aplicativos. Essas linguagens geralmente são bem próximas à maneira que o próprio aplicativo opera. As rotinas geradas nessas linguagens geralmente só podem ser executadas *dentro* dos aplicativos, e recebem o nome de *macros*.

As *macros* trazem, em geral, seqüências de operações do aplicativo, que podiam ser executadas diretamente pelo usuário. Porém, existem 2 elementos que tornam as macros bem mais flexíveis: a possibilidade da utilização de parâmetros ou variáveis (aumentando assim o campo de utilização das macros); e a existência de comandos de laço (do tipo *for ... next*) e comandos condicionais (do tipo *if ... then ... else*). Desse modo o computador pode, de dentro das macros, tomar decisões e realizar repetidas tarefas.

Os softwares de elementos finitos também possuem essas linguagens internas, e é através de macros que a parametrização toma forma. As macros em questão repetiriam a ordem das operações que um usuário executaria se estivesse sentado em frente ao computador.

Uma macro de parametrização deve englobar todo o conhecimento necessário para a obtenção de um modelo fiel ao que é esperado e de uma malha de elementos de boa qualidade. Para isso, o programador da macro deve escrever cada linha de comando para que o modelo seja construído da forma mais objetiva (sem comandos inúteis, que tenham seus efeitos anulados por comandos posteriores) e possibilite elementos bons.

Para se utilizar uma macro de parametrização, deve-se realmente conhecer *apenas* os parâmetros chave. Todas as preocupações no sentido de se construir um bom modelo já foram tidas, todas as decisões necessárias para isso já foram tomadas, e todos os problemas que surgiriam no caminho já foram

enfrentados. É por isso que é correto dizer que a macro realmente engloba todo o conhecimento em torno da construção daquele tipo de modelos.

3.5. Tempo de implantação da parametrização

Todo problema que tínhamos antes da parametrização gira em torno do tempo: o tempo de construção do modelo é muito grande, o tempo gasto para se fazer alterações em uma geometria consolidada é muito grande, não sobra tempo para que o software possa ser utilizado como ferramenta de simulação e otimização dos projetos.

Da mesma forma, todas as vantagens que a parametrização promete também são relativas ao tempo: o engenheiro não gasta seu tempo construindo o modelo, tem mais tempo para se dedicar a tarefas mais importantes, sobra tempo de utilização do computador para que os programas de elementos finitos se tornem auxiliares do engenheiro na tomada de decisões de projeto.

Falta considerar ainda um fator: qual é o *tempo de implementação* da parametrização? Quanto tempo se gastará para se escrever aquelas macros, antes que elas possam abreviar tanto o tempo que é gasto em frente aos computadores, introduzindo os modelos para o cálculo? Em quantos projetos deve-se utilizar essas macros para que elas se tornem viáveis, ou seja, o tempo gasto na sua implementação é menor que o tempo economizado nos projetos em que ela foi utilizada?

Como foi já mostrado no item anterior, as macros da parametrização devem conter todo o conhecimento para a construção de um determinado modelo. Isso significa que em suas linhas de comando existem fórmulas matemáticas e tomadas de decisão. Tudo isso, para ser determinado, requer tempo e dedicação. As fórmulas, por exemplo, não podem se aplicar apenas a um caso particular, como seria feita a conta se o modelo fosse ser construído diretamente. E, dentre as tomadas de decisão, estão também as que não existiriam se a construção do modelo fosse direta: a verificação da validade dos parâmetros (os parâmetros, para

possibilitarem a construção do modelo, devem estar em faixas determinadas e respeitar diversas relações entre si).

Para se somar a esse gasto de tempo direto, existem ainda as perdas indiretas: o tempo que se ocupa o programa de elementos finitos para implementar e testar as macros da parametrização é um tempo que não pode ser usado para efetuar cálculos. Portanto o tempo de processamento do computador na fase de implementação da parametrização é ainda menor que esse mesmo tempo antes da implementação.

Apesar disso, para empresas que têm grande número de projetos em carteira e pretendem permanecer no mercado ainda por muitos anos, é claro que a parametrização é viável. Uma vez implantado um programa de parametrização, é só se utilizar dele quantas vezes for necessário. As macros, como são programas de computador, não exigem manutenção física nenhuma. Elas podem, isso sim, ficar obsoletas face a algum avanço tecnológico que venha a acontecer e mudar o padrão de projeto de determinada peça. Mas para isso podem-se fazer alterações na macro que incorporem os novos métodos, ou até mesmo fazer uma nova macro, se essa inovação for mais revolucionária.

3.6. Um pequeno exemplo de parametrização

Vamos considerar um exemplo bem simples de parametrização, para que se possa observar a aplicação dos conceitos vistos até aqui e sair um pouco da campo da abstração.

Vamos imaginar uma firma que construa trampolins de piscina de vários tamanhos, e use um software de elementos finitos para auxiliar no projeto desses trampolins.

O primeiro passo da parametrização é se definir um modelo padrão. No caso dessa empresa, foi adotado que o modelo correto seria uma viga engastada em uma extremidade e em balanço na outra, com uma carga pontual aplicada na extremidade livre.

O próximo passo é a escolha dos parâmetros que serão usados como dados de entrada da macro. No caso, naturalmente se escolhe o comprimento da viga, L, sua espessura, e, a carga aplicada em sua extremidade livre, P, e as características do material da viga e de sua seção transversal.

A macro da parametrização deve seguir os seguintes passos:

- Escolher o tipo de elemento utilizado - no caso, seria um elemento de viga;
- Associar os parâmetros de material e de geometria (espessura, área e momentos de inércia da seção transversal) aos valores entrados pelo usuário;
- Gerar uma linha de comprimento L que represente a viga;
- Gerar os elementos que comporão este modelo;
- Colocar as restrições que correspondem ao engastamento em um nó em um dos extremos da viga;
- Aplicar, no nó do outro extremo, a carga P;
- Rodar a solução.

Todas essas tarefas, que deveriam ser feitas novamente a cada vez que se quisesse calcular um trampolim diferente, com a macro ficam reduzidas apenas à entrada dos dados.

4. Objetivos deste trabalho

4.1. Escolha das partes do hidrogerador que sofrerão a parametrização

Vistos os conceitos que levam à adoção da parametrização, e os métodos para implementá-la, vamos agora partir para o escopo deste trabalho em si: quais as partes do hidrogerador que sofrerão esse processo.

Seria possível se parametrizar *todos* os componentes do hidrogerador, as isso tomaria o trabalho de anos de um profissional. É necessário que se faça uma análise da relação custo/benefício aí envolvida. Valerá mais a pena a implementação da parametrização de componentes pesados e caros, pois a economia trazida pela parametrização e seus impactos nos custos finais seriam maiores.

É natural que um processo deve começar a ser implementado com as partes mais importantes, para que se sintam seus efeitos e para que os resultados de sua implementação sejam logo sentidos, como um incentivo para se estender a implementação do processo para *todas* as outras partes possíveis.

Tendo em mente esse critério de escolha, pode-se tomar a decisão. Este trabalho vai se ater a uma parte da estrutura do hidrogerador: a *carcaça*. Essas é uma parte importante, de grandes dimensões, projetada para suportar pesados carregamentos e para durar muitos anos.

4.2. Escolha do software utilizado para a implementação das *macros*

Outra escolha muito importante para o trabalho é em que *software* de elementos finitos as macros (que são na verdade o produto final deste trabalho) serão implementadas. dedicar minhas horas de estágio a esse trabalho (que a meu ver é de grande importância para a empresa).

O software de elementos finitos usado na área de geração de energia da ABB (e, portanto, o que foi utilizado para o desenvolvimento deste trabalho) é o

ANSYS (Swanson Analysis Systems). A versão do software é a 5.3. A linguagem utilizada para a implementação das macros será a APDL, a linguagem de programação interna ao ANSYS.

O ANSYS é um software muito poderoso, permitindo cálculo de transientes, análises de sistemas não-lineares, propriedades que variam em função de outras. Além disso oferece uma multidisciplinaridade muito grande: podem ser feitos modelamentos de sistemas mecânicos, elétricos, magnéticos, térmicos, fluidos, e mistos. Entretanto, não feita realmente uma *escolha* do software, já que mesmo que haja um mais adequado a esse trabalho, o meu acesso a ele seria bem mais difícil e, como a empresa não se utilizaria do resultado final, provavelmente eu não teria a ajuda que estou tendo e essa disponibilidade de horas para me dedicar.

4.3. A Carcaça do gerador

A primeira parte do gerador a sofrer a parametrização será a carcaça. A função da carcaça é abrigar todo o enrolamento e as chapas magnéticas do estator. Nela também se encontram os trocadores de calor, responsáveis por manter a temperatura da máquina em níveis adequados. Em muitos casos, a carcaça tem ainda a missão de transmitir os esforços da cruzeta superior à base de concreto que é a fundação da máquina.

A análise da carcaça será estática, linear e as propriedades dos materiais serão admitidas constantes. Os resultados que se busca são os níveis de tensão na carcaça devido ao carregamento, e os deslocamentos que nela se verificarão.

Os carregamentos a que a carcaça está sujeita são:

- Peso próprio;
- Peso do enrolamento e do núcleo do estator da máquina;
- Peso dos trocadores de calor;

• Diferença de temperatura entre partes eletromagnéticas do estator e prateleiras da carcaça;

• Torque nominal da máquina;

• Torques em casos extremos (erros de sincronismo e disparo)

• Empuxo magnético;

• Cargas provenientes da cruzeta superior;

4.4. A carcaça padrão: ponto de partida para a parametrização

Vamos agora eleger as características que *sempre* estarão presentes nas carcaças dos hidrogeradores. Essas serão as diretrizes da parametrização. Toda característica que variar respeitando esse padrão entrará para as macros como parâmetro.

Como se pode ver na figura 1, os elementos construtivos básicos da carcaça são:

• Prateleiras: São anéis que dão toda a volta em torno da carcaça. A parte interna da prateleira é *sempre* circular (padrão), e é definida pelo raio interno **ri** (parâmetro). A externa é *sempre* poligonal (padrão), definida pelos raios dos pontos médios de cada lado, **r1** e **r2** (parâmetros);

• Colunas: São as chapas verticais que sustentam as prateleiras, e transmitem seus esforços à fundação. As colunas são inclinadas (padrão), e o ponto médio de seu perfil está na mesma posição angular que o ponto médio do segmento de prateleira em que a coluna se encontra (padrão). O número de colunas **nc**, a largura da coluna **lc** e seu ângulo de inclinação **gama** podem variar (parâmetros).

As prateleiras, com exceção da inferior, sempre estão à mesma distância uma da outra (**dpi**). A distância da prateleira inferior à prateleira logo acima dela geralmente é um pouco maior (**dii**). A espessura da prateleira inferior também é maior que a das outras, e seu raio interno também é diferente (**rii**, sempre menor).

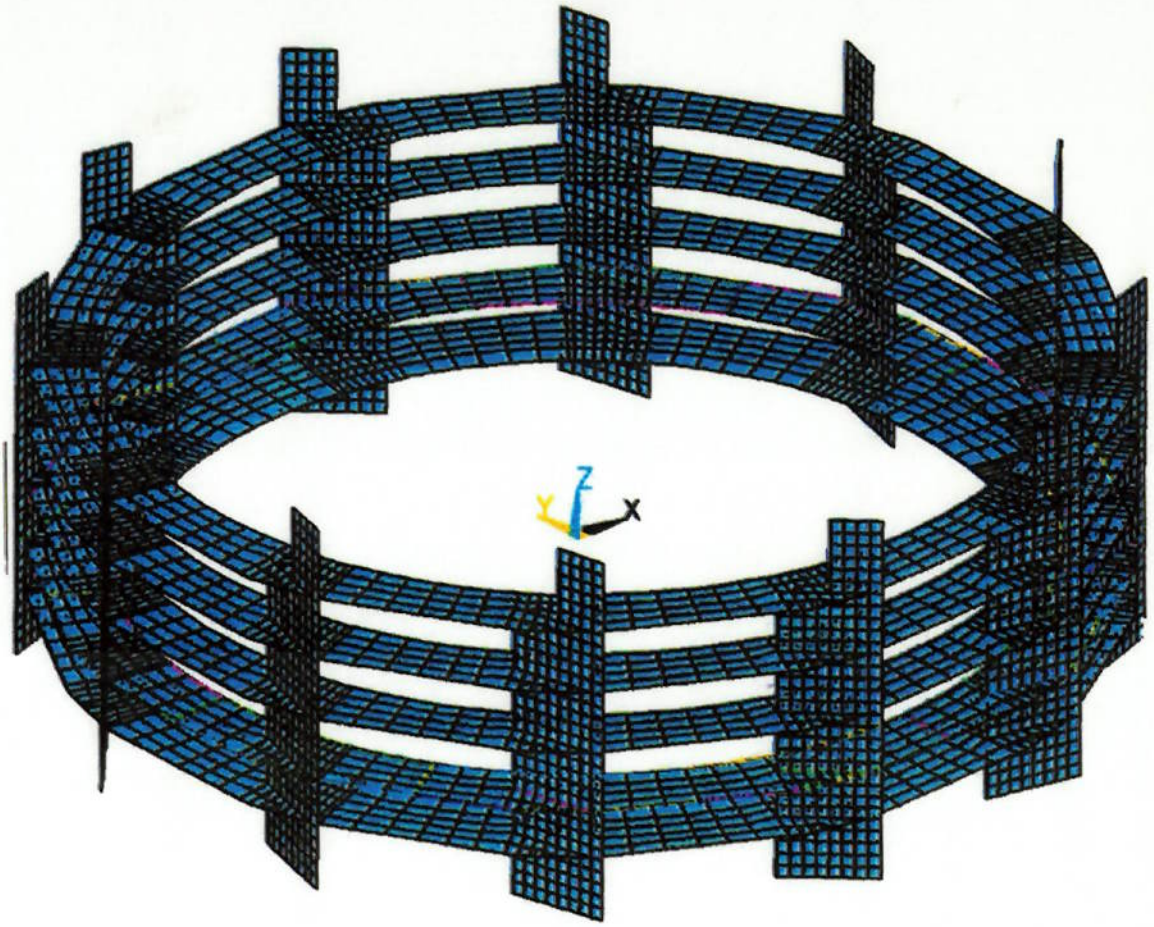


Figura 1: Modelo da carcaça do hidrogerador

A coluna, acima da prateleira superior, tem outra largura (**lcs**, menor que a largura do resto da coluna), e uma determinada altura **acs**.

Os códigos em negrito são os nomes dos parâmetros associados a cada uma dessas propriedades geométricas do modelo. Existem outros parâmetros geométricos, assim como parâmetros de carregamento, de propriedades dos materiais e dados sobre as espessuras das chapas da carcaça.

A figura abaixo mostra o aspecto de um segmento da carcaça, indicando os parâmetros geométricos que o determinam.

5. Método de Implementação

Até agora vimos apenas as definições de o que vai ser feito, e como vai ser feito. Antes de se começar realmente a escrever as macros, deve-se ater um bom tempo a estudos acerca das peças a serem parametrizadas, comparações entre projetos que já foram feitos, conversas com pessoas da empresa no sentido de verificar qual seria a geometria padrão.

Um passo seguinte é a definir a abordagem ao problema. Já sabemos o que deve ser feito para alcançar os objetivos e já sabemos os aspectos técnicos da estrutura real que são importantes na modelagem. Agora devemos nos preocupar com a maneira de implementar isso eficientemente no computador. O programa deve ter uma estrutura lógica simples e bem definida, que permita entendimento e futuras alterações por outras pessoas.

Outra preocupação é com os parâmetros: o usuário deve ter fácil acesso aos parâmetros que o interessam, e as alterações nos parâmetros devem ser feitas com agilidade.

5.1. Características da Análise

Uma coisa de extrema importância que deve ser definida antes de qualquer outra coisa é exatamente qual o tipo de análise que será feita. Em outras palavras, *o que* desejamos avaliar na nossa estrutura? Que aspectos de seu comportamento nos preocupam? O modelo da estrutura que será montado dentro do ANSYS deverá ser adequado ao tipo de análise que se deseja fazer.

Neste trabalho, a análise será estrutural, estática, linear. Queremos avaliar as tensões e deslocamentos na estrutura e admitimos o carregamento constante ao longo do tempo na análise. Além disso, consideraremos que as propriedades dos materiais são constantes (não há por exemplo, variação de temperatura que cause uma alteração sensível em qualquer propriedade dos materiais no modelo).

As macros da parametrização pressupõe que essa decisão já foi tomada. Elas funcionarão de modo a gerar o modelo o mais adequado possível para se observar as tensões e deformações na estrutura. Se se desejasse fazer um cálculo modal da carcaça, por exemplo, o modelo deveria ter outras características. Os parâmetros que alimentariam as macros poderiam até ser os mesmos (mesmo que provavelmente algumas informações podem ter utilidade apenas para uma das análises), mas eles seriam interpretados de maneira diferente conforme o tipo de análise.

5.2. Características do modelo

Definido o tipo de análise, o primeiro passo para se criar um modelo matemático propriamente dito para ser calculado por elementos finitos é definir o tipo de modelo mais adequado a essa análise. Isto é, quais tipos de elemento serão usados.

Para se tomar essa decisão deve-se ter em mente o compromisso entre qualidade dos resultados da análise e tempo de processamento / espaço em disco ocupado pelo modelo. Parece lógico que, quanto maior a precisão que se deseja na análise, mais "pesado" terá de ser o modelo. Mais espaço em disco ele ocupará e maior será a demora para o seu processamento.

Sendo assim, entre as alternativas possíveis, o tipo de elemento escolhido para a análise é o elemento de casca (SHELL) de 4 nós. Simulando a carcaça por elementos de viga (BEAM), estaríamos comprometendo a confiabilidade do modelo. Se optássemos por elementos sólidos (BRICK) ou de casca com 8 nós (nós intermediários nos lados do elemento), estaríamos ocupando espaços enormes em disco e gastando precioso tempo de processamento em nome de uma precisão não tão maior nos resultados.

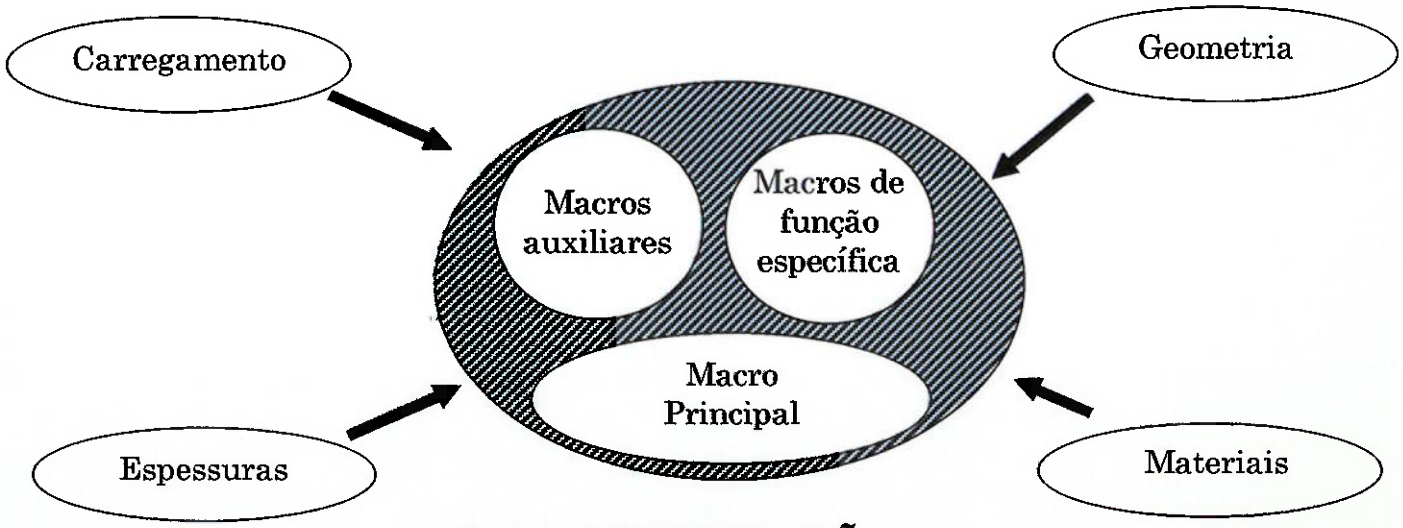
Os elementos de casca são bastante adequados para simular chapas. É necessário fornecer ao ANSYS a espessura da chapa (nesta análise estaremos usando espessuras constantes ao longo do elemento). A superfície que representa a chapa é a superfície que ocupa o "centro" da chapa (isto é, que está à meia distância

entre a superfície superior e inferior da chapa). Por exemplo, duas chapas de 50 mm de espessura, posicionadas paralelamente, com um espaço entre elas de 200 mm, deveriam ser modeladas como duas superfícies paralelas distantes de 250 mm uma da outra.

5.3. Estrutura do programa e definição dos parâmetros

Vamos agora definir o funcionamento interno das macros, e sua interação com os parâmetros.

Para facilitar o trabalho de programação e a estruturação lógica das rotinas, a parametrização da carcaça do gerador é constituída de várias macros. Uma *macro principal* (carcaca.mac), que controla o funcionamento geral do programa. *Macros de funções específicas* (malha_p.mac, malha_c.mac, carreg.mac) são responsáveis por aspectos distintos da modelagem. Há ainda as *macros auxiliares* (verifcruz.mac, delall.mac, editpar.mac), que desempenham funções menores no contexto da parametrização, ou que são acessadas separadamente ao programa principal.



A função exata de cada macro e seu funcionamento serão explicadas mais adiante.

Os parâmetros que serão usados para a geração do modelo estão agrupados em quatro *arquivos de parâmetros* (carcaca.parm, carreg.parm, mat_props.parm, esp.parm). Os quatro arquivos são lidos pela macro principal no início da execução. Estão organizados dessa maneira para facilitar o acesso do usuário aos parâmetros, e agilizar a edição e modificação de valores.

Esquematicamente, podemos então representar a estrutura lógica da parametrização da seguinte forma:

5.4. Etapas realizadas pelo programa

A construção do modelo da carcaça do gerador, feita pelo programa de parametrização, segue etapas muito bem definidas:

- Calcular as posições dos pontos chave da geometria de um segmento da prateleira da carcaça, e da posição dos pontos que definem o perfil da coluna;
- Analisar o tipo de interface que a coluna tem com a geometria externa da prateleira;
- Traçar as linhas que definem, enfim, a geometria desse segmento de prateleira;
- Distribuir os nós convenientemente sobre essas linhas - lembrando que eles devem ser mais próximos na região da coluna - conforme o tipo de interface coluna/prateleira detectado pelo próprio programa;
- Preencher toda a área desse segmento de prateleira com nós, e já definir os elementos de casca entre esses nós;
- Uma vez completada a malha do segmento da prateleira, copiar esse segmento para as outras prateleiras. Na prateleira inferior, ainda deve-se adicionar elementos na parte interna do segmento, pois seu raio interno geralmente é menor que o das outras prateleiras;

• Definir as posições dos nós da coluna e construir os seus elementos. Acima da prateleira superior, a coluna tem uma largura menor. Com a coluna pronta, tem-se toda a geometria de um segmento da carcaça;

• Se necessário, copiar essa malha para todos os segmentos da carcaça, completando assim toda a volta (isso não precisa ser feito se se pretende usar simetria);

• Aplicar em todos os nós do modelo a temperatura ambiente quando a máquina estiver em operação, e habilitar a gravidade (peso próprio);

• Selecionar todos os nós da(s) coluna(s) em contato com as fundações (concreto), e impor a eles uma condição de engastamento;

• Selecionar todos os nós da geometria interna da prateleira, aplicar neles a força devida ao torque aplicado e, além disso, a temperatura do núcleo do estator da máquina;

• Informar ao usuário que o modelo está pronto. Perguntar a ele se deve-se prosseguir com a análise.

Todas essas funções são desempenhadas pelas macros da parametrização. Existem outras duas funções executadas por macros alheias à parametrização, mas que são auxiliares e facilitam bastante a relação do usuário com o programa:

• Editar os arquivos de parâmetros;

• Apagar todo o modelo, para que se possa fazer alterações nos arquivos de parâmetros e começar de novo.

Assim, fecha-se a gama de atribuições do pacote de macros da parametrização. A estruturação lógica das ligações entre as macros e entre elas e os arquivos de parâmetros também já está traçada.

6. Funcionamento do programa

Depois de definido o problema, o caminho para a solução e a maneira de se implementar essa solução, vamos agora passar à implementação propriamente dita das macros. Cada uma das "missões" da parametrização será focada e atacada da maneira mais adequada, e o método de solução será mostrado em detalhes.

6.1. Geometria básica

O primeiro grupo de "missões" a serem cumpridas pelas macros da parametrização é a de definir os pontos e linhas que compõe a geometria básica de um segmento de prateleira. Essa tarefa, à primeira vista pode parecer bem simples, mas deve-se ter em mente que vários parâmetros podem variar influenciando o aspecto dessa geometria básica.

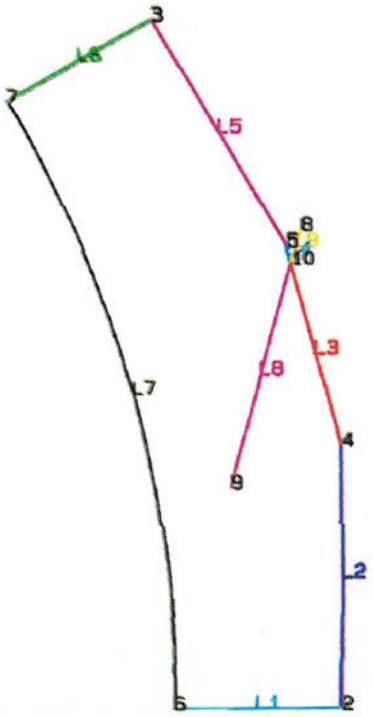


Figura 2: Geometria básica do segmento da prateleira

A figura 2 mostra a geometria básica do segmento de prateleira, com o perfil da coluna. É nessa missão que estaremos nos focando agora.

O perfil da coluna, por exemplo, pode interagir com a prateleira de diferentes maneiras, cruzando com diferentes linhas externas desta última, ou mesmo não cruzando com elas.

É também importante que a definição desses pontos e linhas seja feito de forma tal que facilite o trabalho da próxima etapa, que é distribuir os nós nessas linhas criadas.

6.1.1. Verificação dos dados de entrada:

Os parâmetros importantes na formação da geometria básica deverão ter uma certa "compatibilidade" entre si. Dependendo da combinação de valores que eles tiverem, pode ser impossível obter a geometria básica. As restrições são descritas matematicamente:

- **ri<r1**
- **ri<rc<r2**
- **rii <ri**, onde **rii** é o raio interno da prateleira inferior
- **r1***cos(**alfa**/2)<**r2**<**r1**/cos(**alfa**/2) , onde **alfa** é o ângulo de cobertura do segmento da prateleira (será explicado melhor logo adiante).

6.1.2. Posição dos pontos-chave da geometria básica:

Como se pode ver na figura 2, existem 9 pontos importantes na geometria básica. Com exceção do ponto 10 (cuja posição dependerá de como se dá o cruzamento da coluna com a prateleira), todos os pontos terão agora sua posições definidas pela *macro principal* do programa (carcaca.mac).

A primeira coisa a fazer é determinar o ângulo de cobertura de um segmento. Esse ângulo (**alfa**) é obtido simplesmente dividindo os 360° da volta completa pelo número de colunas **nc** da carcaça.

Para criar os pontos 6 e 7 que definem a linha interna do segmento, vamos usar coordenadas cilíndricas: a posição do ponto é expressa na forma (r,θ,z).

Nesse sistema de coordenadas (SC), o ponto 6 está em (**ri**,0,0), enquanto o ponto 7 está em (**ri**,**alfa**,0).

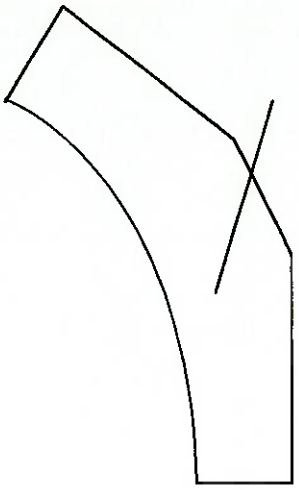
Usando o mesmo SC, podemos posicionar os pontos 2 e 3, que são os extremos da linha externa do segmento. Ao se juntar a segmentos vizinhos, os pontos 2 e 3 serão pontos médios de segmentos de reta. Portanto, sua distância ao centro é exatamente **r1**. Assim, as coordenadas do ponto 2 são (**r1**,0,0) e do ponto 3 são (**r1**,**alfa**,0).

O processo para se criar os pontos 4 e 5, que definem a outra linha externa do segmento, é um pouco mais complicado. Para o ponto 4, deve-se mudar a origem do SC para o ponto 2, e o eixo x na direção 2-6. O ponto a ser criado então tem a posição (0,-**r1/2**,0). Para o ponto 5 o procedimento é o mesmo: translada-se a origem do SC para 3 e define-se o eixo x na direção 3-7. A posição do ponto 5Y será também (0,-**r1/2**,0), mas nesse outro sistema de coordenadas.

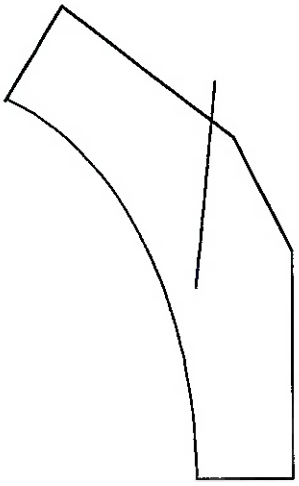
Para criar os pontos do perfil da coluna, vamos nos valer de dois pontos auxiliares, que são o ponto médio desse perfil (esse ponto é mais fácil de ser obtido), e o ponto médio da linha externa do segmento da prateleira. Em coordenadas cilíndricas, esses pontos auxiliares estão em A:(**rc**,**alfa/2**,0) e B:(**r2**,**alfa/2**,0). Muda-se então o sistemas de coordenadas para cilíndrico com origem em A e eixo x na direção A-B. Os pontos 8 e 9 do perfil da coluna, portanto, estarão nas posições (**lc/2**,**gama**,0) e (-**lc/2**,**gama**,0), respectivamente. Os pontos A e B são removidos após esta operação.

6.1.3.Análise da interface Coluna / Prateleira:

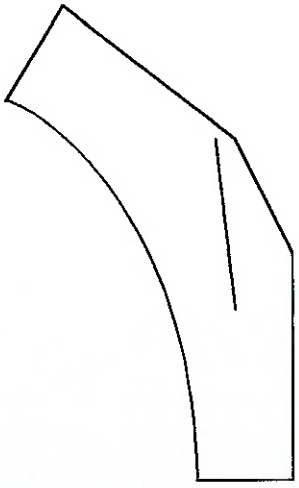
Se traçássemos agora as linhas que ligam os pontos obtidos no item anterior, descrevendo assim a linha interna do segmento, as linhas externas e a linha do perfil da coluna, iríamos nos deparar com uma das 3 situações:



Situação 1: a coluna cruza a geometria externa da prateleira na altura de sua linha intermediária



Situação 2: a coluna cruza com a geometria externa do segmento na altura de sua linha superior.



Situação 3: a coluna e a geometria externa do segmento não se cruzam.

Um dos pontos-chave da geometria básica do segmento é justamente o ponto de cruzamento do perfil da coluna com a linha externa da prateleira. As linhas da geometria externa serão cortadas neste ponto, para facilitar a divisão das linhas em nós e a posterior construção dos elementos. Sendo assim, é de grande importância determinar em que caso estamos situados. Este trabalho é realizado *pela macro principal*, com a ajuda da *macro auxiliar* verifcruz.mac.

O funcionamento da macro verifcruz.mac é baseado na geometria analítica. Ela tem como dados de entrada os números dos pontos que definem as linhas que vão participar da análise. A partir da leitura da posição dos pontos das linhas num SC global, verifica-se se há o cruzamento. Se sim, a variável auxiliar **cruza** é atualizada com o valor 1, e a macro calcula a posição exata do encontro. Senão, o valor de **cruza** é 0.

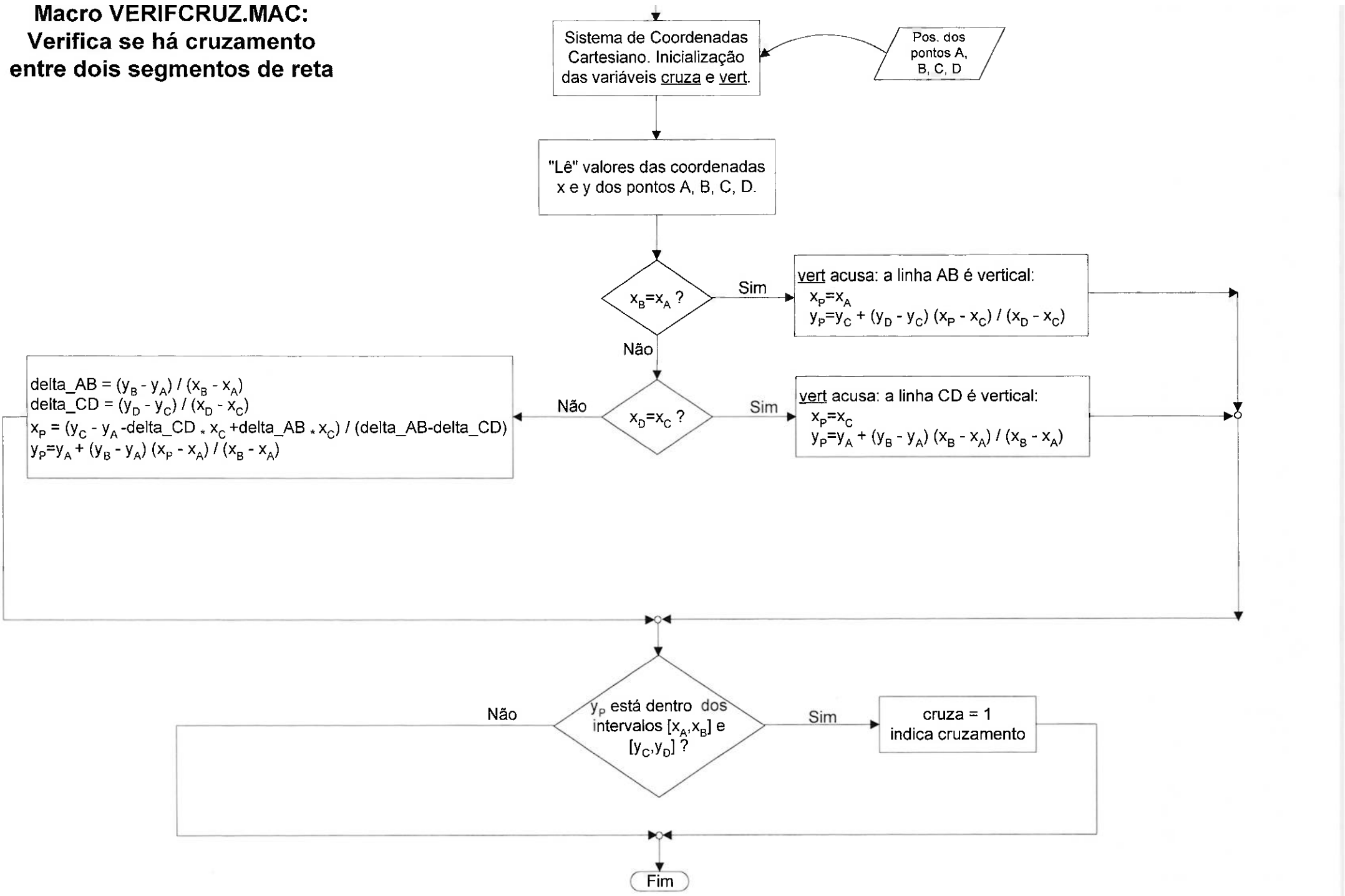
A macro principal chama a macro verifcruz.mac detectar o cruzamento do perfil da coluna com a linha intermediária externa da prateleira. Se não houver o cruzamento, ma nova verificação é feita com a linha do perfil da coluna e a linha superior externa da prateleira. Em caso de cruzamento entre essas duas linhas, o valor de **cruza** é mudado, na macro principal, para 2. Se não há cruzamento em nenhum dos dois casos, teremos **cruza**=0.

Se **cruza** for diferente de zero (isto é, o perfil da coluna "atravessa" a linha exterior da prateleira, o ponto 10 é criado na posição do cruzamento, fornecida pela macro verifcruz.mac.

O programa de parametrização desenvolvido até aqui não lida com o caso em que **cruza**=0. Assim, se não há o cruzamento da coluna com a parte externa da prateleira, a execução do programa termina por aqui.

A próxima página traz um fluxograma da macro *verifcruz.mac* para facilitar o entendimento.

Macro VERIFCRUZ.MAC:
Verifica se há cruzamento entre dois segmentos de reta



6.1.4.Traçagem das linhas que definem o segmento:

A macro principal se encarrega de traçar as linhas entre os pontos definidos nas etapas anteriores. A linha 1, ligando os pontos 6 a 2, é o limite inferior do segmento. A linha 2 é a linha inferior da geometria externa da prateleira e liga os pontos 2 e 4.

Se a variável **cruza** vale 1, isso indica o cruzamento da linha intermediária externa da prateleira com a linha do perfil da coluna. Portanto, a linha intermediária externa da prateleira será dividida em duas: a linha 3, entre os pontos 4 e 10, e a linha 4 entre os pontos 10 e 5. A linha 5 é a linha superior externa, ligando os pontos 5 e 3.

Se o valor de **cruza** for 2, o cruzamento do perfil da coluna é com a linha superior externa da prateleira. Nesse caso, a linha 3 representará toda a linha intermediária externa (pontos 4 e 5), e a linha superior estará dividida em linhas 4 (pontos 5 e 10) e linha 5 (pontos 10 e 3).

A linha 6 é o limite superior do segmento, ligando os pontos 3 e 7. Para se traçar o arco de circunferência entre os pontos 7 e 6 (linha interna), deve-se mudar o SC para cilíndrico.

A coluna será composta por duas linhas: a linha 8 do ponto 9 ao 10 (cruzamento com a geometria externa da prateleira) e linha 9 do ponto 10 ao 8.

6.2.Construção da malha de elementos

Agora que já foi definida toda a geometria básica de um segmento da prateleira, podemos passar à parte mais complicada da geração do modelo: a obtenção da malha de elementos finitos.

A versão utilizada do ANSYS possui comandos capazes de criar automaticamente uma malha de elementos sobre uma determinada área. Entretanto, dada a geometria que temos (com a linha da coluna atravessando a linha externa da prateleira em ângulos nem sempre "amigáveis"), a geração automática de malha não é muito confiável.

A solução adotada foi de mais difícil implementação, porém garante uma malha de boa qualidade para uma grande faixa de variação dos parâmetros entrados: as macros da parametrização se encarregam de construir *toda* a malha, elemento por elemento.

Em análises estruturais, o ideal é que os elementos finitos se assemelhem a quadrados (ou seja, elementos quadriláteros, com ângulos próximos de 90° e lados de tamanhos parecidos). Porém, no caso da carcaça e sua geometria peculiar, não é possível evitar o aparecimento de alguns elementos triangulares. É possível, no entanto, controlar o formato, o tamanho e o número de elementos triangulares na malha.

Devido à complexidade da tarefa de gerar a malha, três macros com essa função específica foram criadas: malha_p.mac, malha_c.mac e malhainf.mac. A primeira cuida da malha nas prateleiras, a segunda trata da malha da coluna e a terceira se encarrega de construir os elementos adicionais da malha da prateleira inferior.

6.2.1. Listas de nós

Antes de começar a descrever as etapas do processo de criação da malha, é necessário mostrar uma ferramenta essencial e largamente utilizada nessa parte do programa: o conceito de listas de nós.

As listas de nós são vetores que armazenam uma seqüência de números de nós. À medida que os nós forem sendo criados sobre as linhas da geometria básica, seus números serão armazenados nessas listas, para que o computador se "lembre" da seqüência correta na hora de seguir para a construção dos elementos. Assim, cada linha da geometria básica possui sua lista de nós: **lnint** (linha interna), **lncol** (perfil da coluna), **lncol_f** (parte do perfil da coluna fora da região da prateleira), **lnext** (linha externa), **lninf** (linha inferior).

Há ainda dois usos muito importantes para as listas de nós. Há uma lista de nós (**tri**) que armazena os números dos nós que serão vértices de elementos triangulares. O último uso, mas não menos importante, é o empregado nas listas **lnanter** e **lnatual**. Essas listas são vitais para a construção dos elementos, pois

guardam os números dos nós que serão usados para a confecção dos próximos elementos.

O conceito e a utilização das listas de nós ficará certamente mais claro à medida que a descrição dos procedimentos for avançando.

6.2.2. A criação dos nós sobre as linhas:

Existem duas variáveis internas ao programa (não são parâmetros, mas poderiam ser) que controlam o tamanho dos elementos: elas se chamam **tec** e **tep**, e representam o tamanho aproximado da borda do elemento na região da coluna e da prateleira, respectivamente. Na região da coluna, é bom que os elementos sejam menores, já que nesta região as tensões costumam ser mais altas.

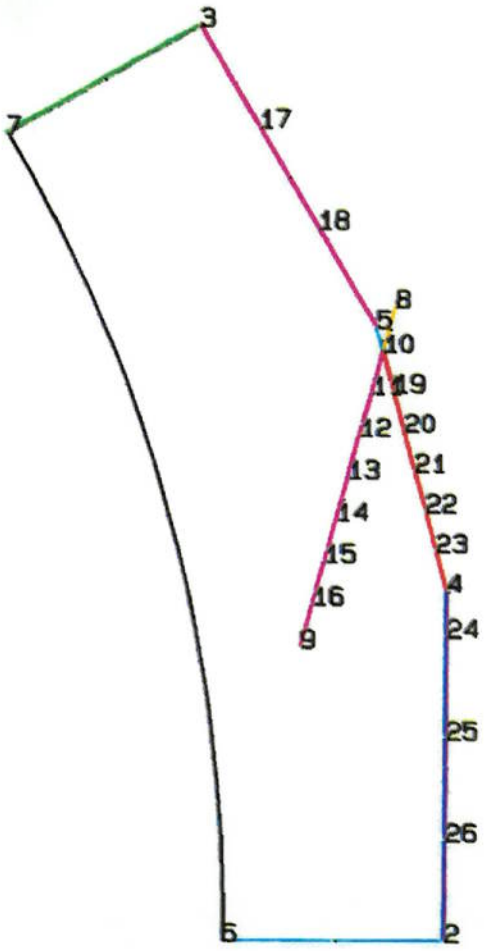


Figura 3: Distribuição dos nós pelas linhas

Para se fazer a "divisão" da linha por esses valores de tamanho da borda de elemento, é necessário conhecer os comprimentos das linhas. O vetor **cl** armazena esses valores. O número de divisões em cada linha será o número inteiro mais próximo do valor da divisão do comprimento da linha pelo tamanho do elemento. Da mesma forma, o tamanho real da borda do elemento sobre aquela linha será a divisão de seu comprimento pelo número de divisões.

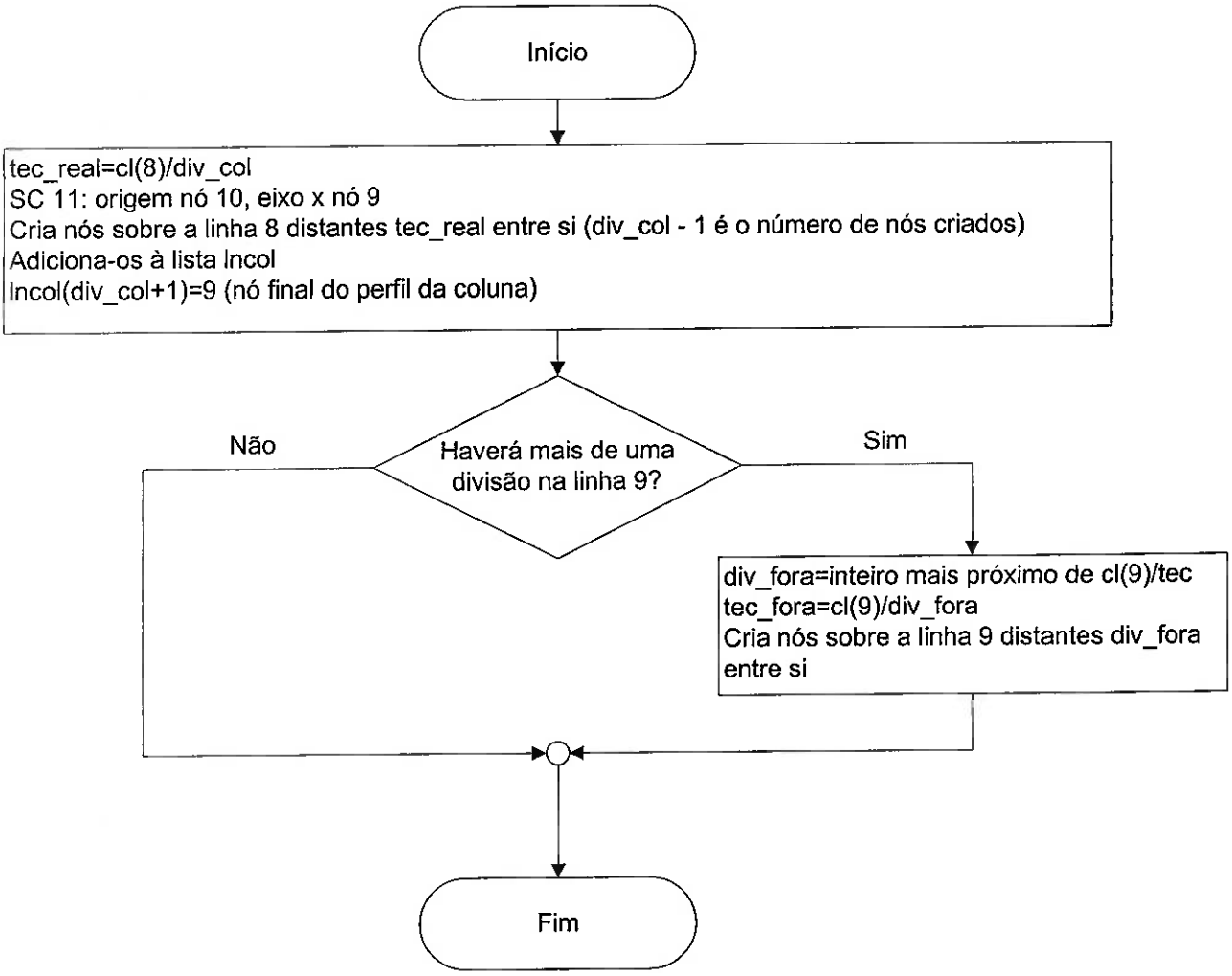
Começaremos pela parte do perfil da coluna. É criado um nó sobre o ponto 10 (o cruzamento da coluna com a linha externa da prateleira).Esse nó será o primeiro da lista **lncol**. Depois muda-se a origem do SC para esse ponto, e o eixo x é a linha do perfil da coluna. Os nós são posicionados sobre essa linha à distância igual ao tamanho da borda do elemento calculado para essa linha. Cada nó adicionado à linha tem seu número incluído na lista de nós **lncol**. Por fim, o ponto final dessa linha (ponto 9) também ganha um nó sobre ele, que ocupará a última posição de **lncol**.

A próxima linha é a parte "externa" da coluna. O procedimento é o mesmo: coloca-se um nó sobre o ponto 8, inclui-se este nó na lista **lncol_f**, verifica-se a necessidade de serem criados novos nós entre os pontos 8 e 10 (essa parte da coluna geralmente é de pequeno comprimento) e, se preciso, cria os nós necessários de maneira análoga à que foi descrita no parágrafo anterior. O nó 10 entrará na última posição de **lncol_f**.

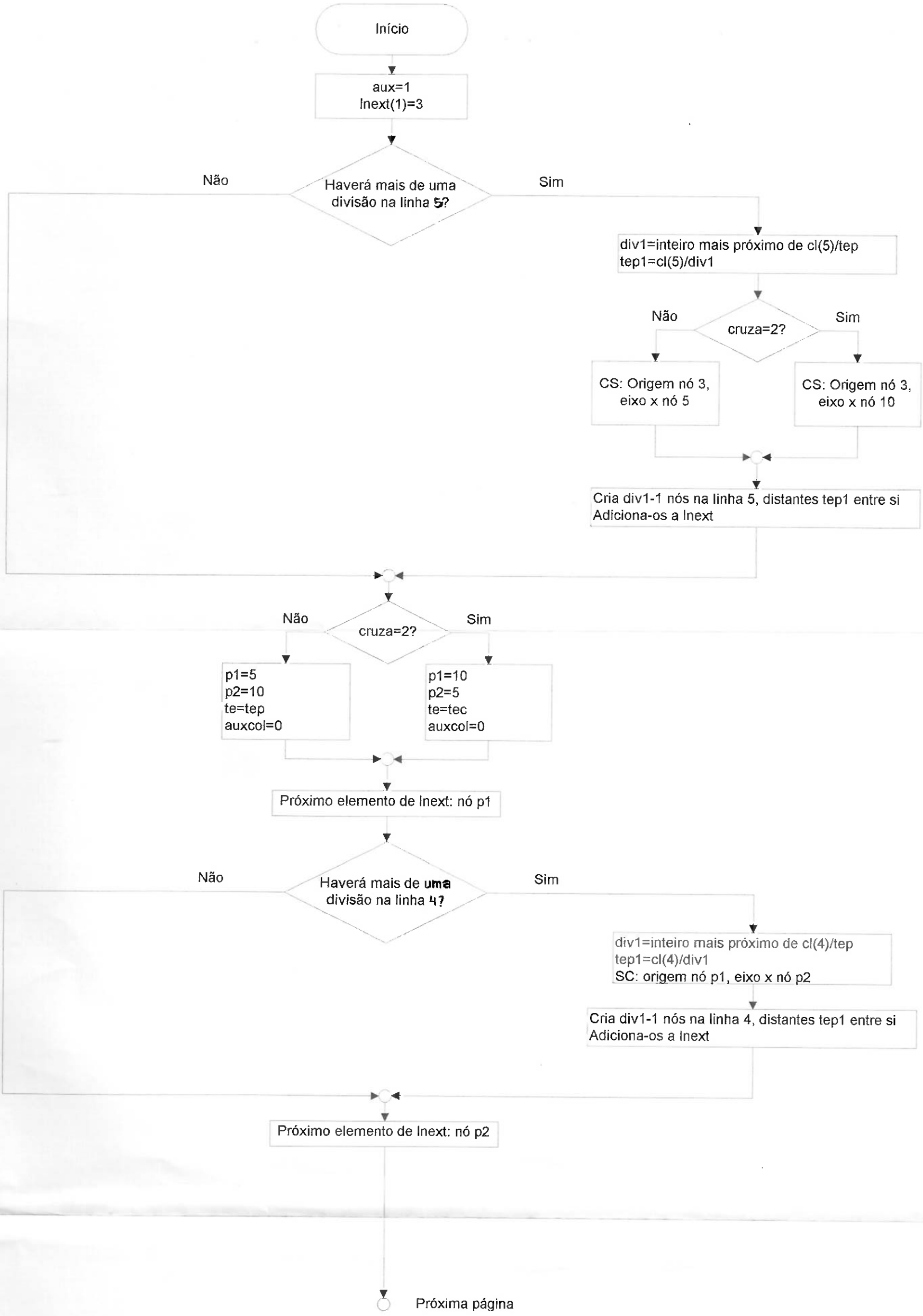
As linhas externas tem uma única lista de nós. A divisão dessas linhas deve ser feita de modo mais cuidadoso, devido à sua interface com a coluna. O primeiro nó da lista é o nó sobre o ponto 3 (extremo superior). A geometria externa possui 4 linhas, e os tamanhos de borda do elemento são diferentes ao longo delas. Para cada uma das 4 linhas a distribuição de nós é feita da mesma maneira que para as linhas do perfil da coluna. No entanto, a informação trazida pela variável **cruza** terá grande importância.

Os fluxogramas das próximas páginas explicam melhor a lógica empregada.

Macro MALHA_P.MAC: Criação dos nós sobre a linha do perfil da coluna



Macro MALHA_P.MAC: Criação dos nós sobre a linha externa do segmento



Página anterior

Haverá mais de uma divisão na linha 3, tamanho tec?

Não

Sim

A linha 3 suportará todas as divisões necessárias de tamanho tec?

Não

Sim

div1=inteiro mais próximo de cl(3)/tec
tep1=cl(3)/div1
SC: origem nó p2, eixo x nó 4

Cria div1-1 nós na linha 3, distantes tep1 entre si
Adiciona-os a lnext

Próximo elemento de lnext: nó 4

div1=o que falta para o no de divs. da coluna
SC: origem nó 4, eixo x nó 2

Cria div1 nós na linha 2, distantes tec entre si
Adiciona-os a lnext

O espaço que "sobrou" da linha 2 suporta mais de uma divisão tamanho tep?

Não

Sim

SC: origem nó 2, eixo x nó 4
div2=inteiro mais próximo de (cl(2)-div1*tec)/tep
tep1=(cl(2)-div1*tec)/div2

Cria div2-1 nós na linha 2, distantes tep1 entre si
Adiciona-os a lnext

div1=o que falta para o no de divs. da coluna
SC: origem nó p1, eixo x nó p2

Cria div1 nós na linha 3, distantes tec entre si
Adiciona-os a lnext

O espaço que "sobrou" da linha 3 suporta mais de uma divisão tamanho tep?

Não

Sim

div2=inteiro mais próximo de (cl(3)-div1*tec)/tep
tep1=(cl(3)-div1*tec)/div2

Cria div2-1 nós na linha 3, distantes tep1 entre si
Adiciona-os a lnext

Próximo elemento de lnext: nó p2

div1=inteiro mais próximo de cl(2)/tep
tep1=cl(3)/div1
SC: origem nó 4, eixo x nó 2

Cria div1-1 nós na linha 3, distantes tep1 entre si
Adiciona-os a lnext

Próximo elemento de lnext: nó p2

Fim

As divisões na linha externa se tornam menores depois que se atinge o ponto 10. O número de divisões menores na linha externa será igual ao número de divisões da linha interna da coluna. O espaço que tiver sobrado na parte inferior da linha externa será preenchido novamente com divisões maiores.

6.2.3. A construção dos elementos:

Como já foi mencionado, a geometria do segmento de prateleira com o perfil da coluna nos obriga a colocar elementos em forma de triângulos em algumas posições.

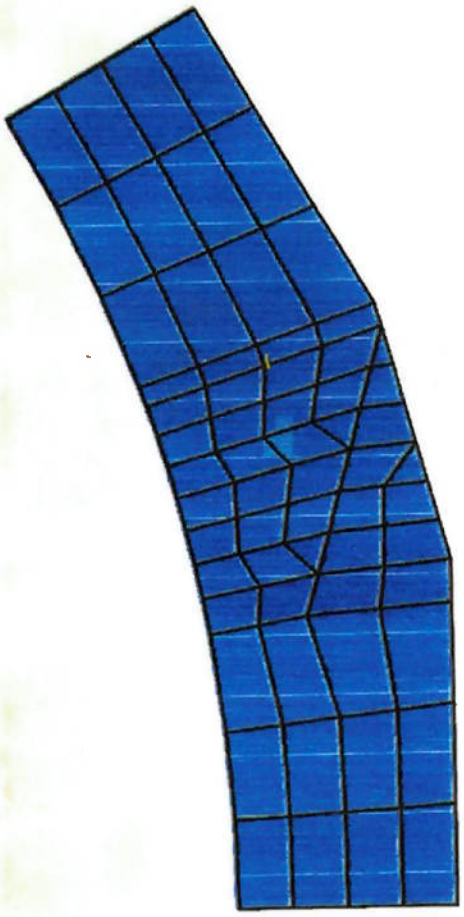


Figura 4: Malha do segmento de prateleira

Na região da coluna, cada elemento triangular criado do "lado direito" da coluna dá origem a uma nova "série" de elementos abaixo dele. Acontece que o número de divisões no limite inferior do segmento tem que ser igual ao número de

divisões no limite superior (mesmo porque quando se conectarem os segmentos da carcaça, o limite inferior de um segmento será o limite superior do próximo, e os nós na interface precisam coincidir). Por isso, para cada "série" de elementos criada do "lado direito" da coluna devido a um elemento triangular, uma outra "série" terá que ser destruída do "lado esquerdo" da coluna, pela criação de outro elemento triangular deste lado.

A lista de nós **tri** contém os nós que são vértices de elementos triangulares. A figura abaixo mostra a malha do segmento, ilustrando o que foi comentado no parágrafo anterior.

Um elemento triangular deve ser incluído do "lado direito" da coluna quando os elementos deste lado estiverem com o tamanho de lado maior do que o comprimento do lado do elemento na direção radial. Isso é verificado através da posição radial dos nós da linha do perfil da coluna (um nó é adicionado à lista tri sempre que a projeção no eixo radial de sua distância ao último nó desta lista for maior que o tamanho das divisões radiais).

A malha do segmento é feita em três etapas, que correspondem às três regiões do segmento mostradas na figura 4. A região 1 é a parte superior do segmento mais a parte "à esquerda" da coluna. A região 2 é a parte "à direita da coluna". A região 3 é toda a parte "abaixo" da coluna.

Para criar a malha na região 1, começa-se construindo nós na linha de limite superior do segmento (entre os nós 3 e 7). Um comando do ANSYS se encarrega de "preencher" a distância entre esses dois nós com um determinado número de novos nós (no caso, o número de divisões na direção radial menos 1) Esses nós, em seqüência (começando pelo 3 e terminando pelo 7), comporão a lista de nós **lnatual**.

A partir daí realiza-se um laço. Os nós que estão na lista **lnatual** são copiados para **lnanter**. Pega-se o próximo nó da linha externa (ou seja, o próximo elemento de **lnext**). A posição angular desse nó é lida e um novo nó é criado sobre a linha interna da prateleira, na mesma posição angular. A distância entre os nós é preenchida com um número conveniente de nós (variável **nnos**: diminui de 1 cada vez que um elemento triangular for criado). Esses nós são agrupados na lista

lnatual. As listas **lnanter** e **lnatual** representam, agora, a seqüência de nós de duas "linhas" vizinhas. Os elementos são criados entre esses nós, na ordem correta.

Este laço é executado até que se alcance o nó 10 na linha externa. A partir desse nó, é a linha do perfil da coluna que delimita a região 1. Assim, após atingir o nó 10, o laço prossegue, mas ao invés de se utilizar dos nós de **lnext**, irá usar os de **lncol**. Quando se chega ao nó 9 (último da parte interna do perfil da coluna), o laço termina. A malha da região 1 foi criada.

Elementos triangulares são criados nessa região. O programa os detecta através da lista **tri**. Para todo nó da linha da coluna que também faz parte de **tri**, um elemento triangular é criado ao lado dela, tendo como vértice esse nó.

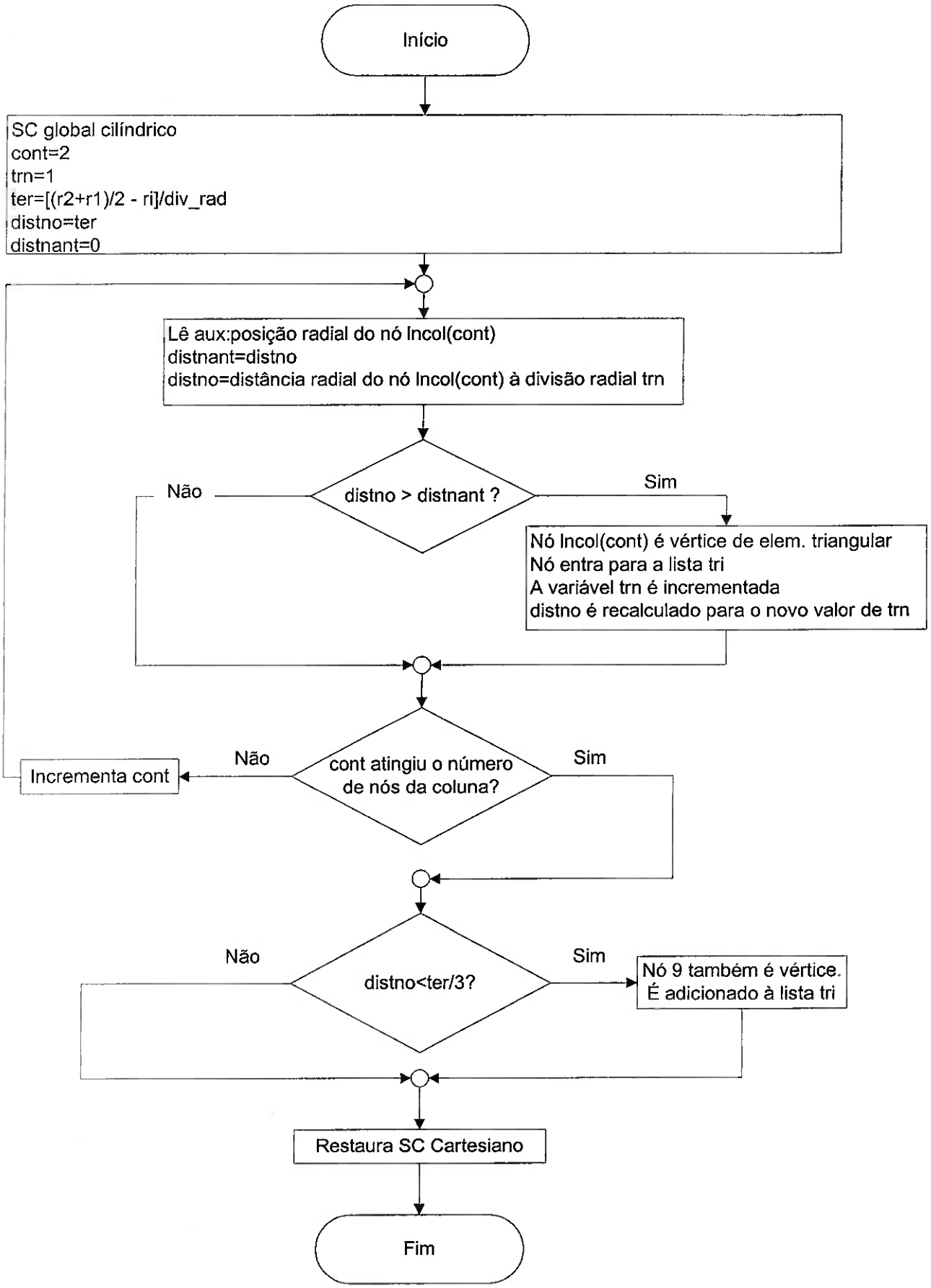
A malha, na região 2, começa com um elemento triangular, tendo como vértices o nó 10, o próximo elemento de **lncol** e o próximo elemento de **lnext**. Aqui acontece o contrário do que acontece na região 1. A variável **nnos** (número de nós entre os nós das linhas extremas da região, nesse caso a linha externa e a do perfil da coluna), começa com o valor zero. A cada elemento triangular criado, com exceção desse primeiro, o valor de **nnos** é incrementado. A maneira de gerar os nós intermediários e construir os elementos é similar à usada na região 1: listas **lnanter** e **lnatual**.

A região 3 é a que tem a criação da malha mais simples, porém necessita que um cuidado tenha sido tomado na criação da malha das duas regiões anteriores. A lista de nós **lninf** deve conter os nós que limitam a região 1 e a região 2 com a região 3. Portanto, parte dela deve ser construída no fim da laço que gera os elementos da região 1 e a outra parte, ao final da operação na região 2.

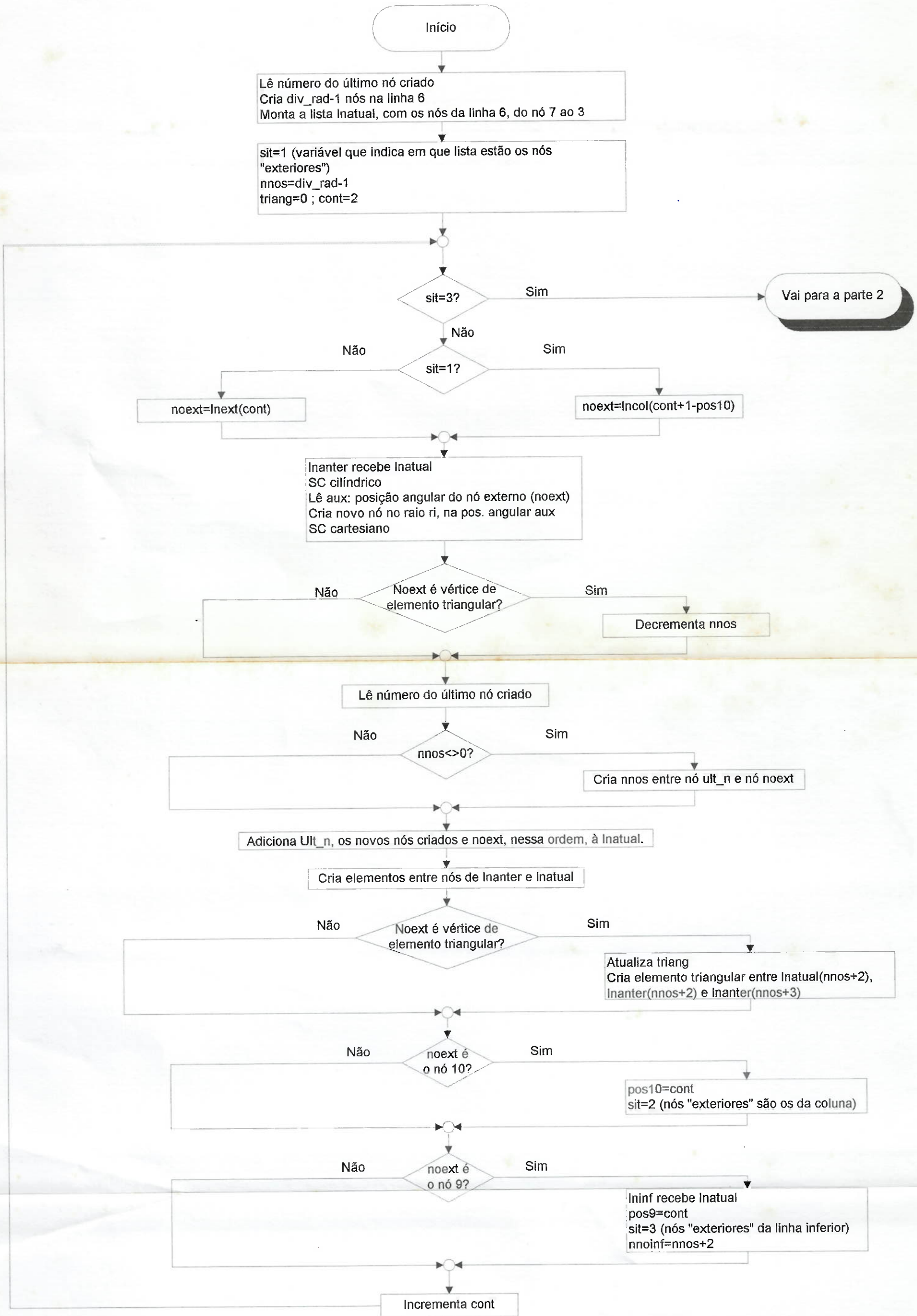
A lista **lnatual**, nessa região, será inicializada com os nós contidos em **lninf**. O laço executado agora é igual ao da região 1: atualiza-se **lnanter** com os nó s de **lnatual**, cria-se um nó na linha interna na mesma posição angular de um da linha externa, criam-se nós entre eles, atualizando **lnatual**, e usa-se os nós de **lnanter** e **lnatual** para gerar uma nova "fileira" de elementos. Esse processo termina quando se atinge o nó 2 na linha externa.

Seguem os fluxogramas.

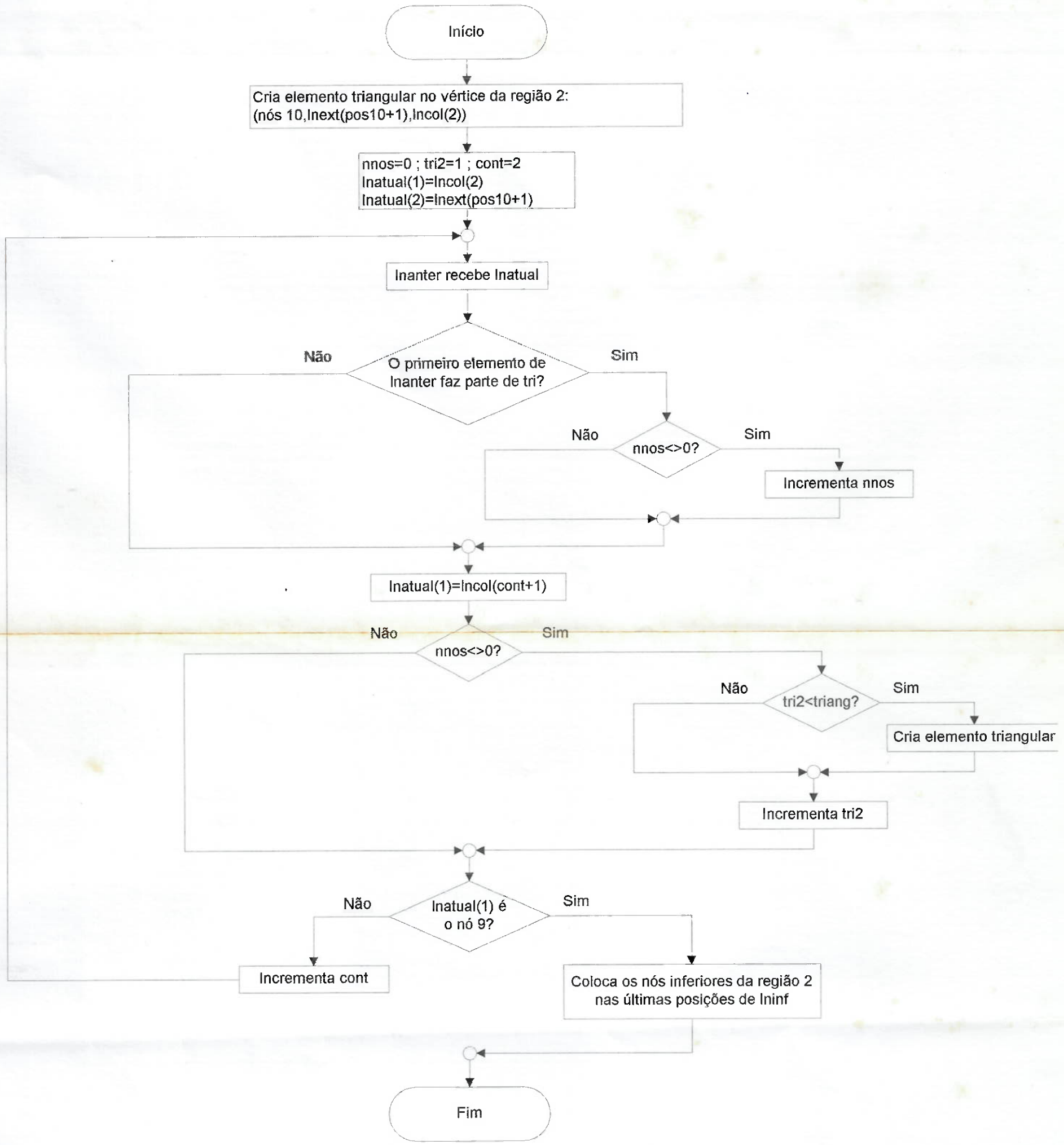
Macro MALHA_P.MAC: Determinação dos vértices de elementos triangulares



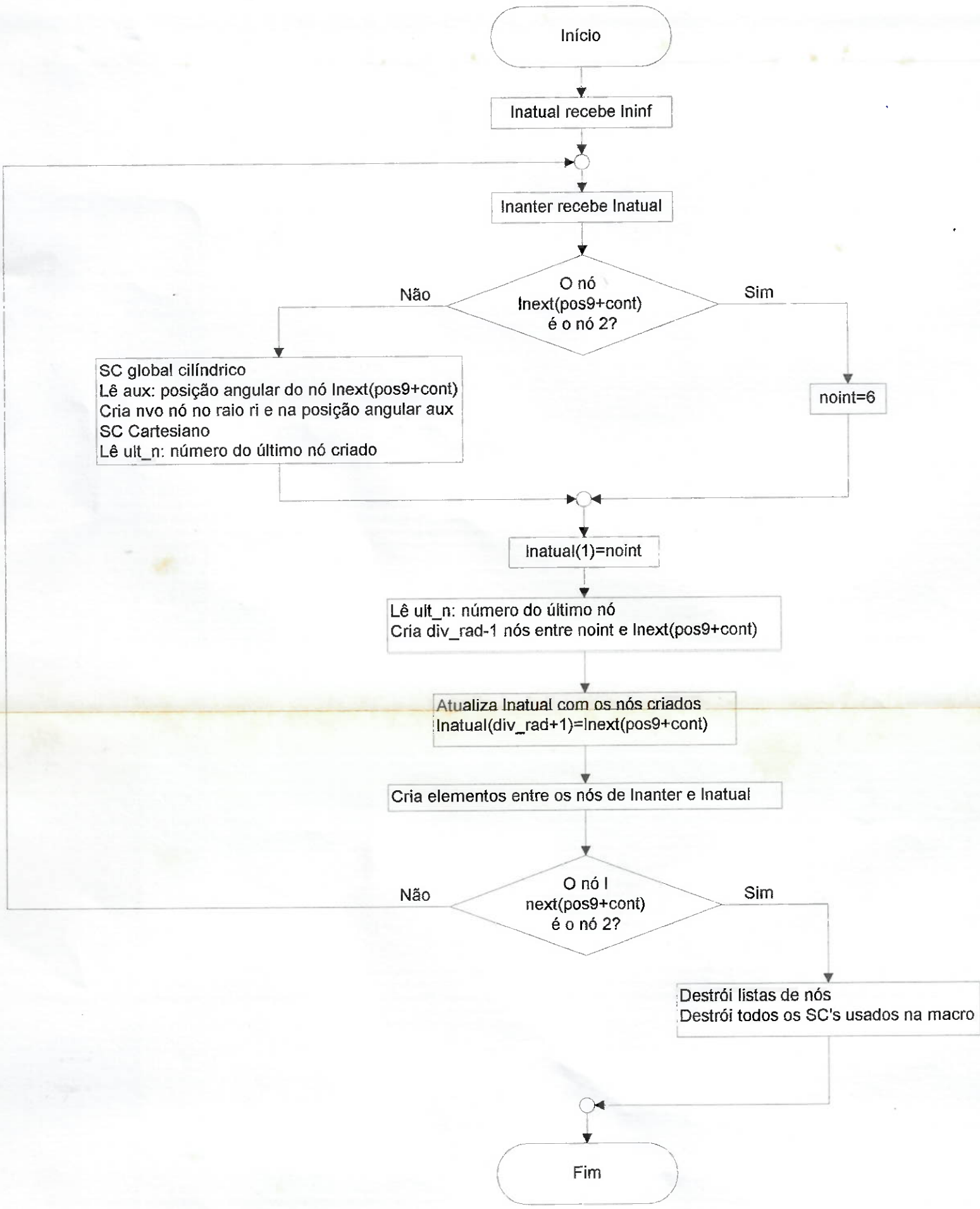
Macro MALHA_P.MAC: Construção dos elementos na Região 1



Macro MALHA_P.MAC: Construção dos elementos na Região 2



Macro MALHA_P.MAC: Construção dos elementos na Região 3



6.3. Obtenção de todo o segmento da carcaça

Todo o esforço realizado até agora nos permitiu obter *um* segmento de *uma* prateleira. A carcaça do gerador possui várias prateleiras (mais do que isso, a prateleira inferior tem espessura e raio interno diferentes das outras), e possui vários segmentos. Além disso, cada segmento da carcaça possui um coluna, cujos nós e elementos ainda precisam ser gerados.

Por outro lado, a tarefa mais complexa no que diz respeito à geometria do modelo já foi superada. Para a obtenção do segmento inteiro da carcaça, três tarefas ainda precisam ser executadas: cópia da malha da prateleira para as outras prateleiras do segmento; geração de elementos na prateleira inferior que completem sua geometria até o raio interno menor; e a confecção da malha da coluna (que já tem sua interseção com as prateleiras devidamente planejada.

6.3.1. Outras prateleiras do segmento:

A macro de função específica *malha_p.mac*, depois de criar toda a malha do segmento da prateleira, devolve o comando à macro principal, que determina a variável **inc_n**: é o número do último nó que foi criado. Essa variável é muito importante na criação dos outros segmentos de prateleira, pois os nós serão "copiados" deste para os outros segmentos, e **inc_n** é o incremento que será dado aos números de nós. Para todo nó *n* da malha do segmento original, seus correspondentes nos outros segmentos terão números iguais à soma de *n* com um múltiplo de **inc_n**.

Com o valor de **inc_n** detectado, tudo o que a macro principal faz é copiar os nós do segmento já gerado, com incremento de **inc_n** em sua numeração, para cima (**npi-1** cópias, para as prateleiras intermediárias, com um espaçamento **dpi** entre uma e outra), e para baixo (uma cópia para a prateleira inferior, à distância **dii** na vertical do segmento gerado originalmente).

Copiamos os nós, mas ainda não há elementos entre eles. Para os elementos, o procedimento é diferente: não se pode copiar um elemento dando a posição em que o novo elemento estará em relação ao primeiro. Para copiar os

elementos, é preciso informar o incremento nos números dos nós que o definem. Assim, usando novamente a variável **inc_n**, geramos os elementos das prateleiras.

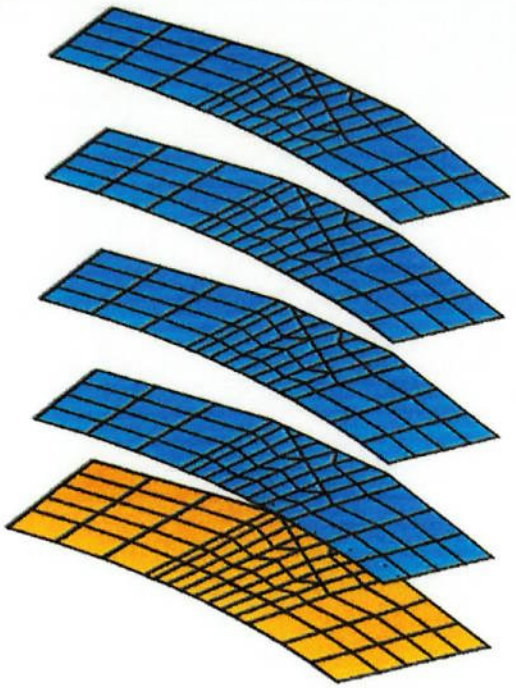


Figura 5: Todas as prateleiras do segmento

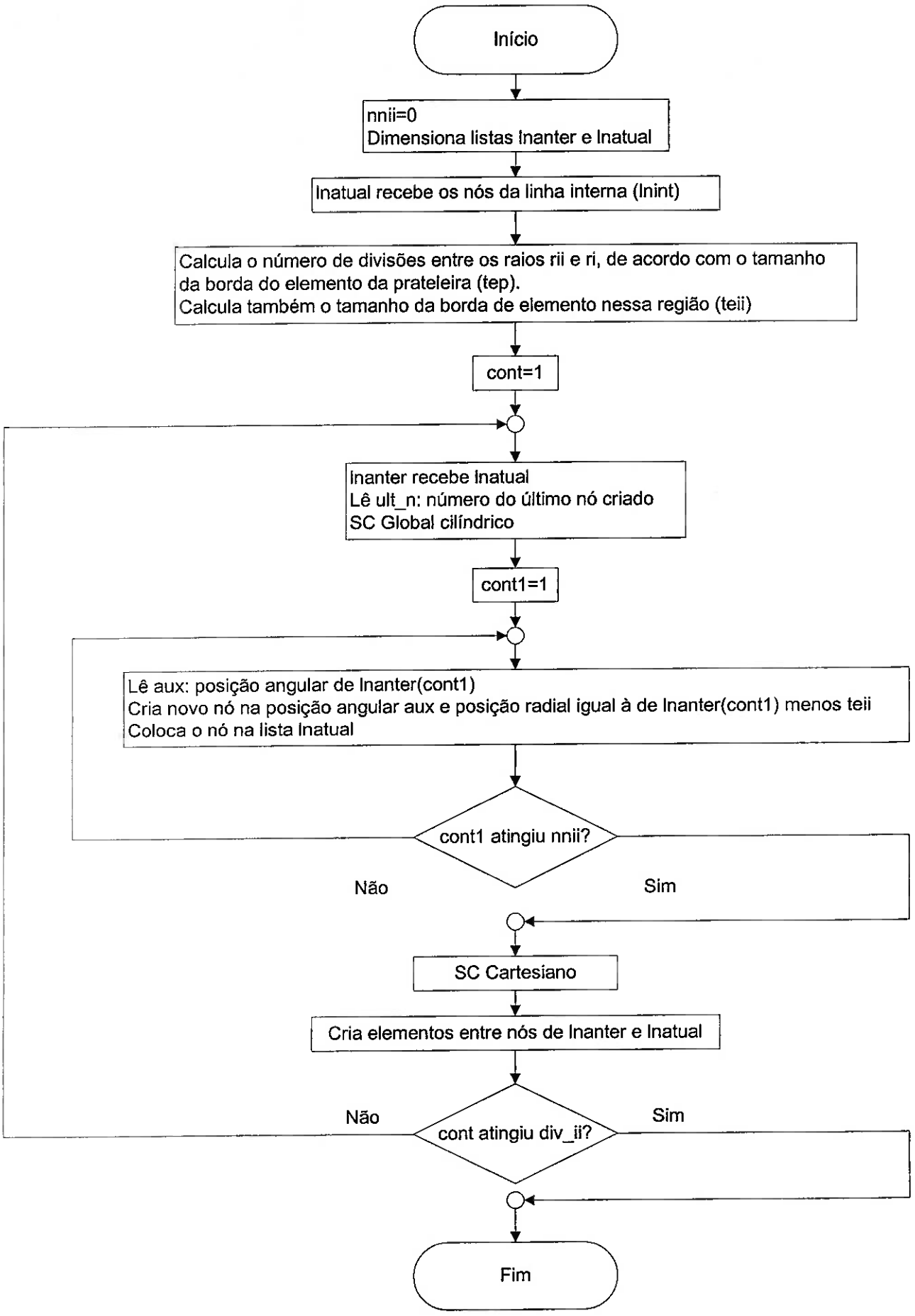
Um cuidado deve ser tomado aqui: a prateleira inferior tem características diferentes das demais prateleiras, inclusive sua espessura (o que se traduz em um conjunto de constantes reais diferente para a prateleira inferior). É preciso, antes de gerar os novos elementos, informar ao ANSYS que as constantes reais referentes àqueles elementos mudarão.

6.3.2. A prateleira inferior:

A missão da prateleira inferior, no conjunto da carcaça, é de maior responsabilidade que as demais. Grande parte do peso do núcleo e do enrolamento da máquina deve ser suportado por ela. Desse modo, sua espessura será maior, e seu raio interno, menor.

A próxima página traz um fluxograma que mostra a lógica empregada nessa parte do preograma. Na página seguinte segue a descrição.

Macro MALHAINF.MAC: Elementos da parte interna da prateleira inferior



Do ponto de vista do modelo matemático a ser gerado, esse é mais um fator complicador. O que foi feito até agora foi criar um segmento de prateleira intermediária que pudesse ser repetido para o resto da estrutura. Quando aplicamos isso para a prateleira inferior, vemos que falta uma região que deve ser preenchida com elementos. Esses elementos serão compostos de nós novos e nós antigos (que já pertencem à malha do segmento original).

Para a realização dessa tarefa, é usada a macro de função específica *malhainf.mac*.

O programa, nessa parte, deve se "lembrar" quais são os nós que fazem parte da linha do raio interno do segmento da prateleira intermediária, para que a partir deles crie os elementos que ocuparão a área compreendida entre o raios internos da prateleira inferior e das intermediárias.

Isso é conseguido, mais uma vez, graças às listas de nós. No caso, a lista **lnint**, que armazena os nós dessa linha interna do segmento. Lendo as posições angulares que esses nós ocupam, o programa cria novos nós nas mesmas posições angulares, e com um recuo radial correspondente ao tamanho de elemento que se tenha definido na prateleira. Isso garante elementos com formas próximas à quadrada também nessa região.

6.3.3. A malha da coluna do segmento

A última parte do segmento ainda por ser feita é a coluna propriamente dita. Todos os cuidados tomados até agora na obtenção do segmento da prateleira serão úteis para a construção da malha da coluna.

A macro de função específica *malha_c.mac* é a encarregada dessa missão. Ela usa também as listas de nós. A lista **lnc**, mais especificamente, é construída como uma "fusão" das listas **lncol** e **lncol_f**, definidas ainda na geração da malha da prateleira, e que representam a partes da colunas que estariam em contato ou não com a prateleira, respectivamente.

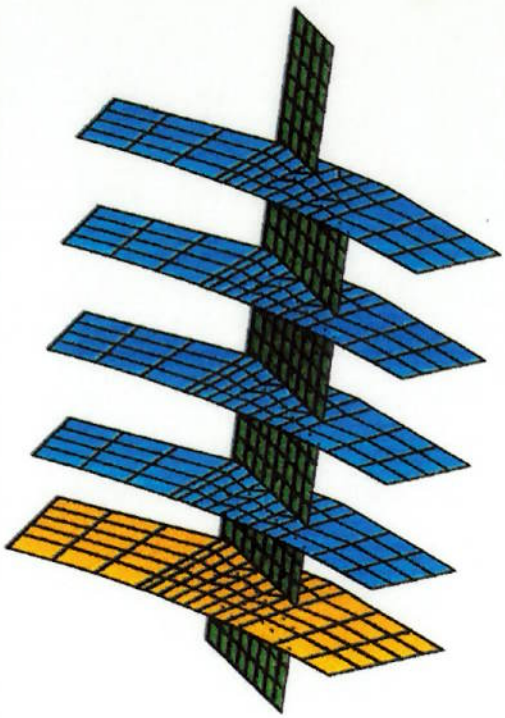
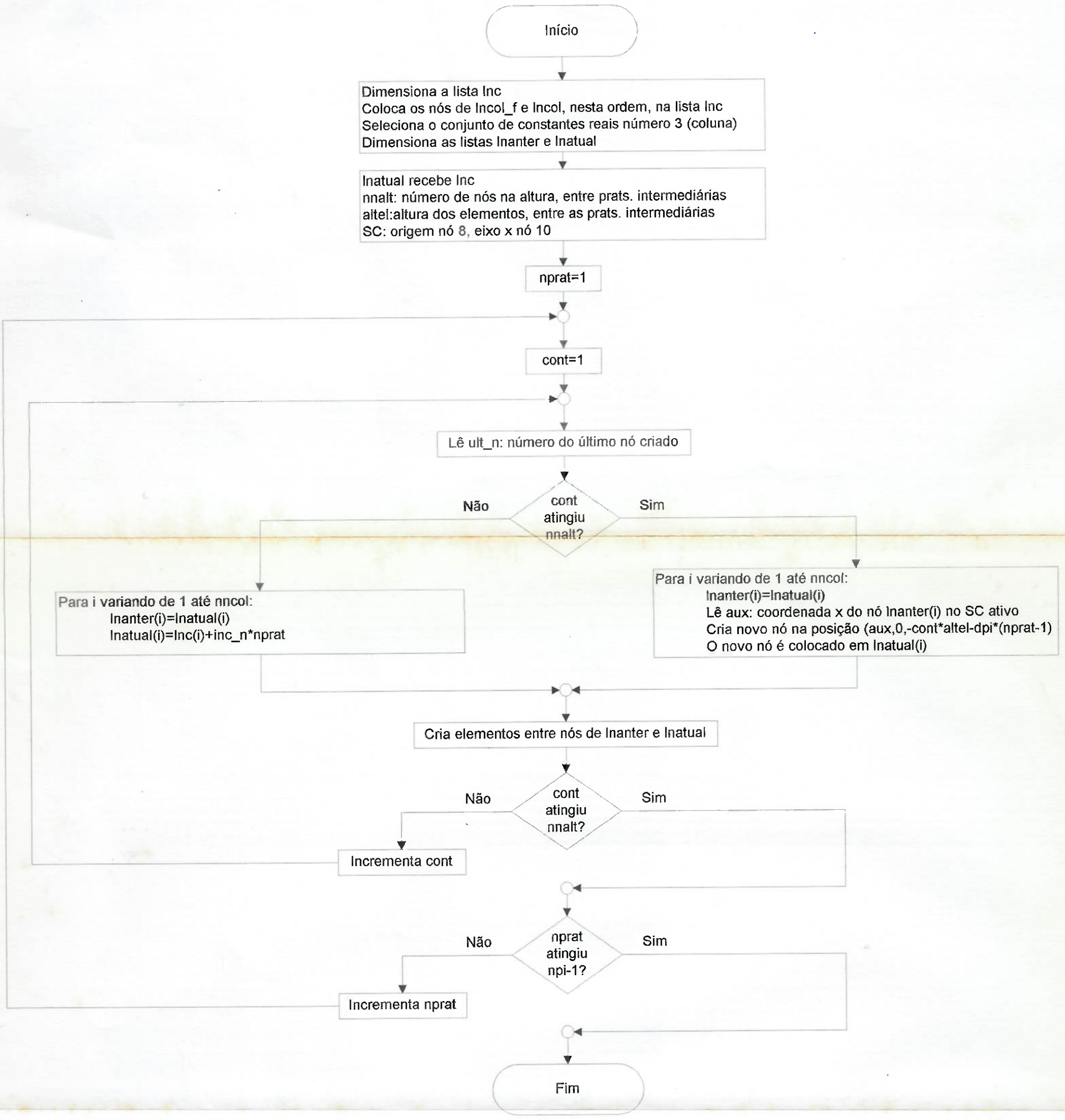


Figura 6: O segmento completo da carcaça

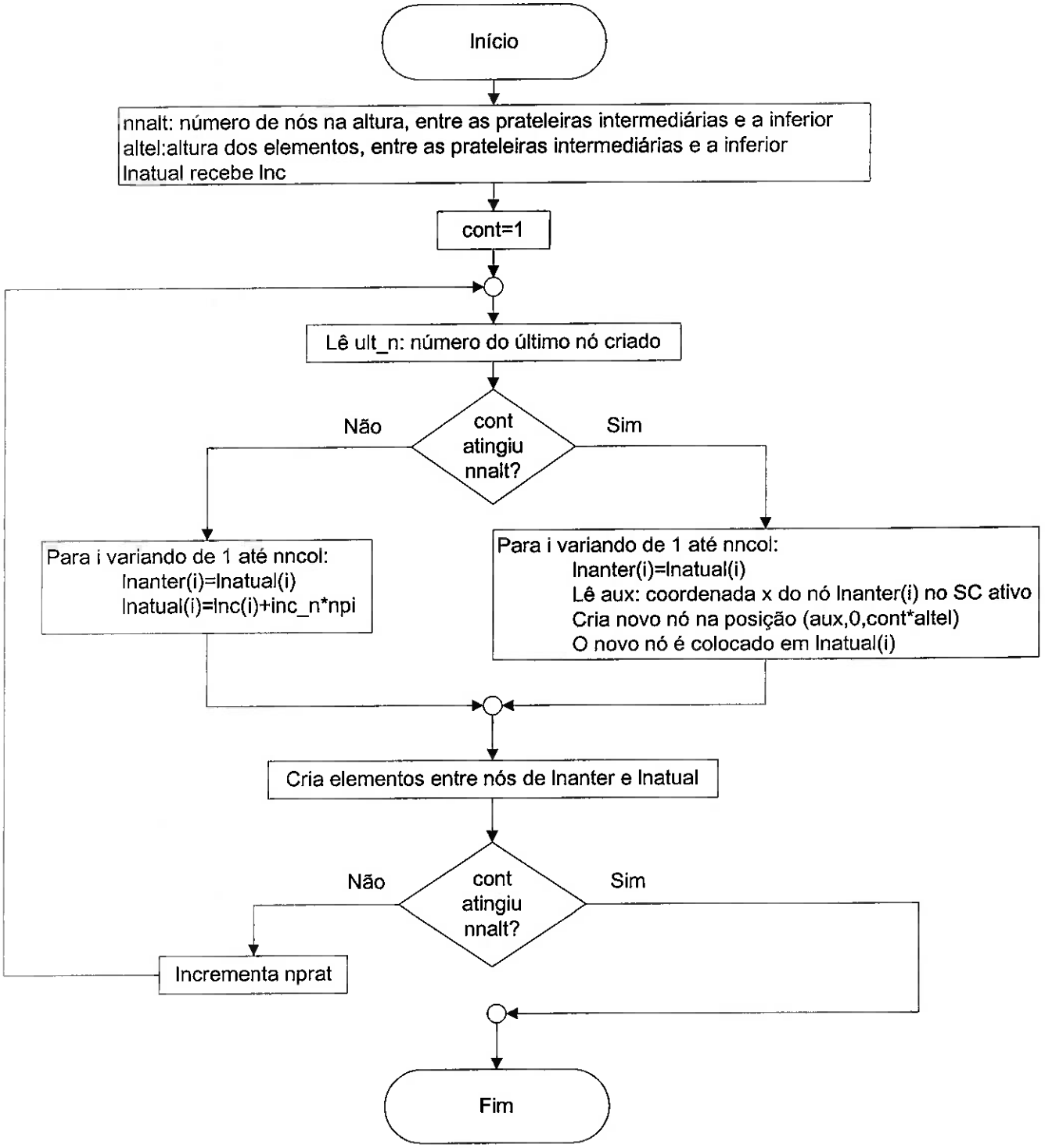
A lista **lnc** serve como ponto de partida da macro. Com o tamanho **tec** da borda do elemento da coluna, o programa cria nós alinhados aos originais na direção vertical, para definir entre eles uma "fileira" de nós. Outros nós são criados à mesma distância vertical dos anteriores, possibilitando a construção de uma nova fileira. O processo se repete até que se atinja a prateleira seguinte. Nesse caso não se criam novos nós, pois estes já existem na prateleira. Volta à cena a lista **lnc**: com ela e com a variável **inc_n**, sabemos exatamente quais são os nós dessa prateleira que acabamos de atingir.

. Os fluxogramas das próximas páginas explicam melhor a lógica empregada.

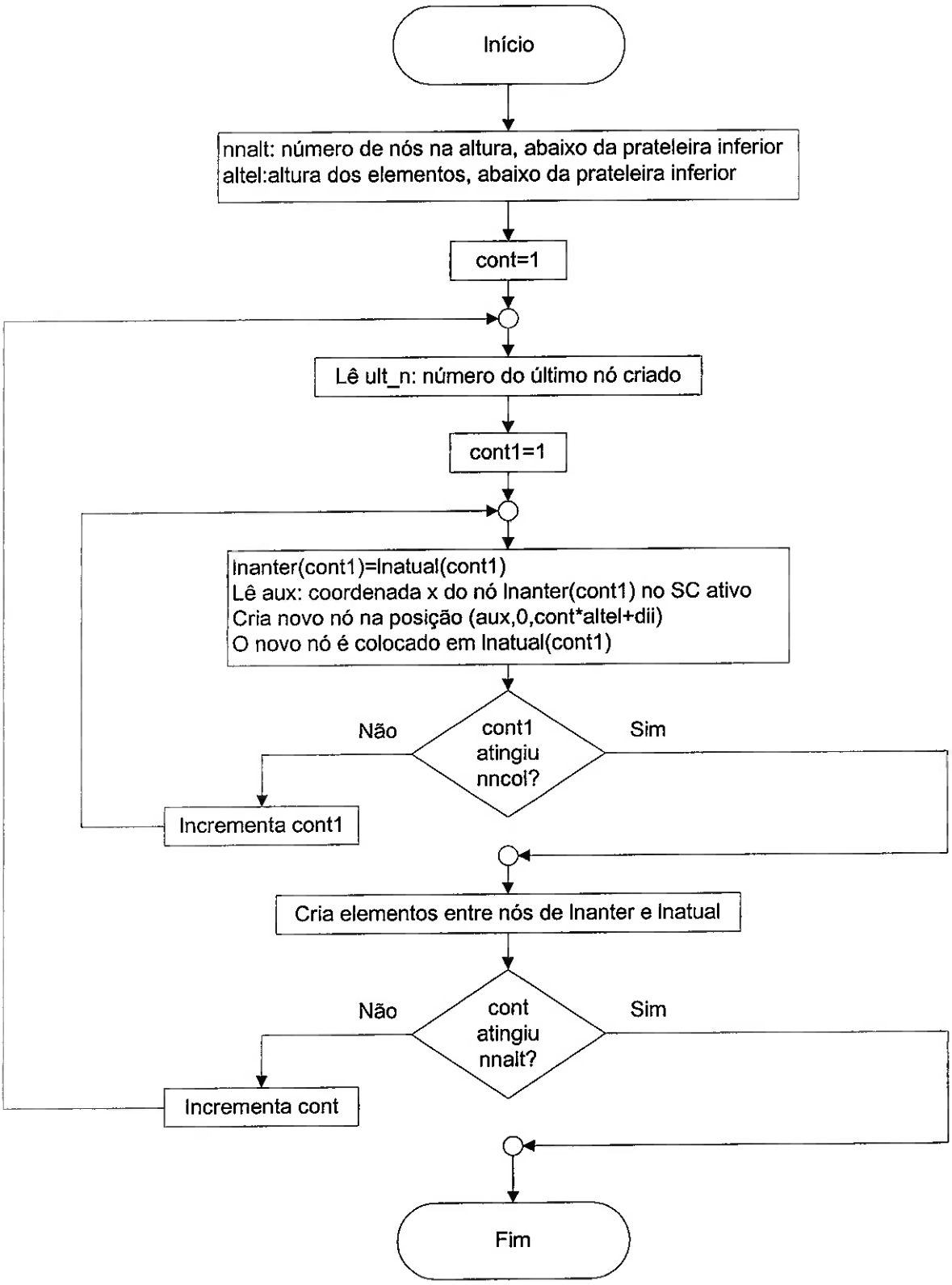
Macro MALHA_C.MAC: Malha da coluna entre as prateleiras intermediárias



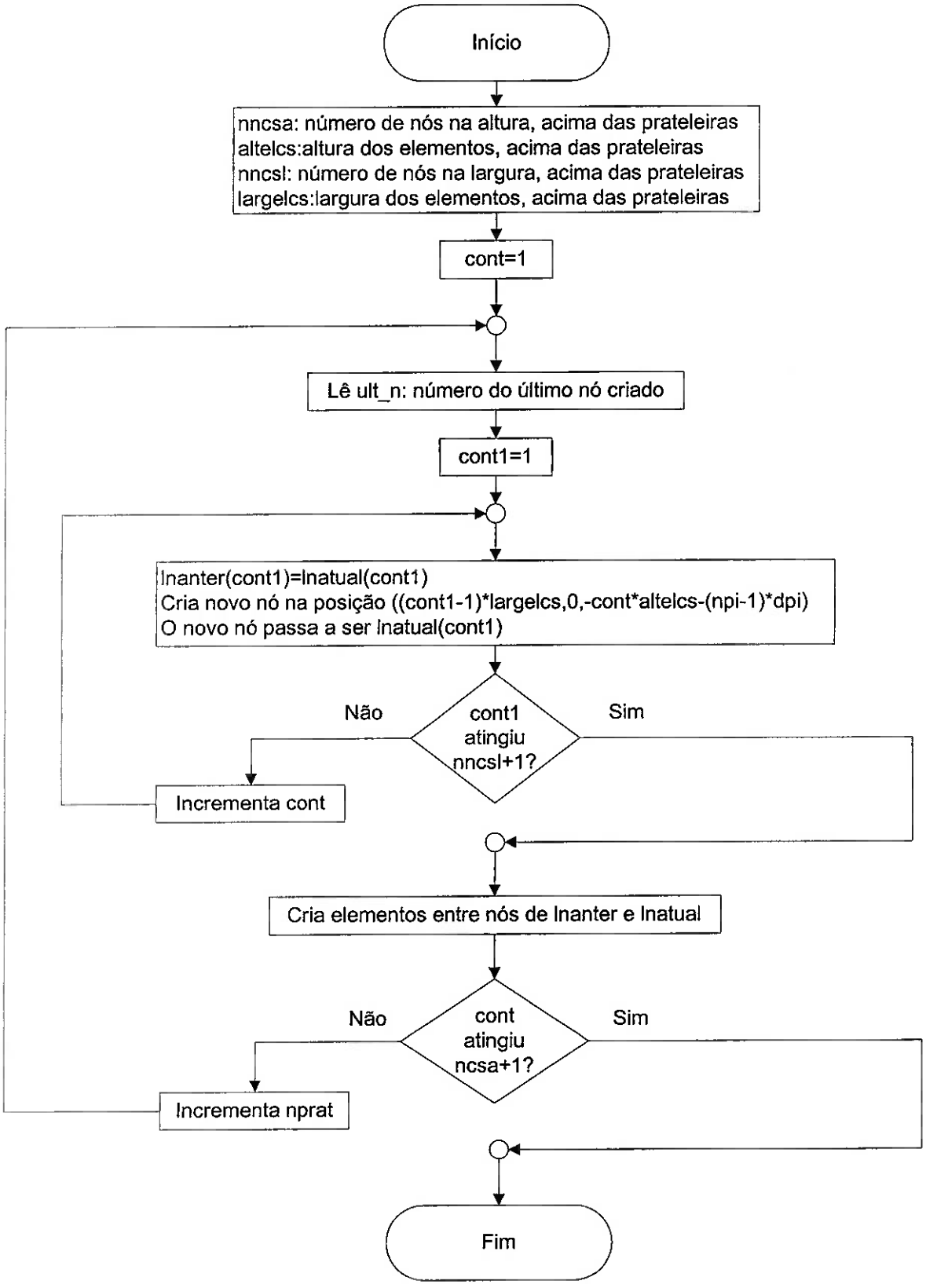
Macro MALHA_C.MAC: Malha da coluna entre as prateleiras intermediárias e a inferior



Macro MALHA_C.MAC: Malha da coluna entre a prateleira inferior e a base



Macro MALHA_C.MAC: Malha da coluna acima das prateleiras



O procedimento descrito se repete até que se atinja a prateleira mais alta da carcaça. Finaliza-se aí a primeira parte da construção dos elementos da coluna (a parte da coluna entre as prateleiras intermediárias).

Existem outras três regiões na coluna: entre a prateleira inferior e a primeira prateleira intermediária; abaixo da prateleira inferior; e acima da prateleira superior. Nessa três regiões a lógica utilizada é semelhante à da primeira, observando-se as diferenças entre as regiões. Entre as prateleiras intermediárias e a inferior o processo é quase idêntico, só mudando a distância total entre as duas prateleiras. Abaixo da prateleira inferior, a diferença é que não prateleira no outro extremo, sendo necessário criar os nós até a última linha. Acima das prateleiras o caso se assemelha ao anterior, somado ao fato de que a largura da coluna, nessa região, também é diferente.

6.4. O modelo completo

A parte geométrica do nosso modelo matemático está completa. A partir dos parâmetros contidos no arquivo *carcaça.parm*, todas as peculiaridades de cada carcaça, desde que obedecendo ao padrão estabelecido, estão devidamente representadas no nosso modelo.

A única coisa que falta ser feita é, a partir de um segmento da carcaça (que é o que *de fato* temos até agora), gerar todos os **nc** segmentos, perfazendo assim a carcaça completa.

Tecnicamente, essa é uma tarefa bastante fácil de ser executada: basta proceder como foi feito para copiar um segmento da prateleira para as outras prateleiras: verifica-se quantos nós há, copia-se todos os nós com esse incremento na numeração, e em seguida copiam-se também os elementos.

Entretanto, a utilização de um modelo completo só se justifica em casos raros. A geometria do modelo é simétrica, e em quase todos os casos, o carregamento e demais condições de contorno também o são. Dessa forma, é uma vantagem grande se trabalhar com um segmento da carcaça. O que for verificado como resultado nesse segmento valerá para todos os outros.

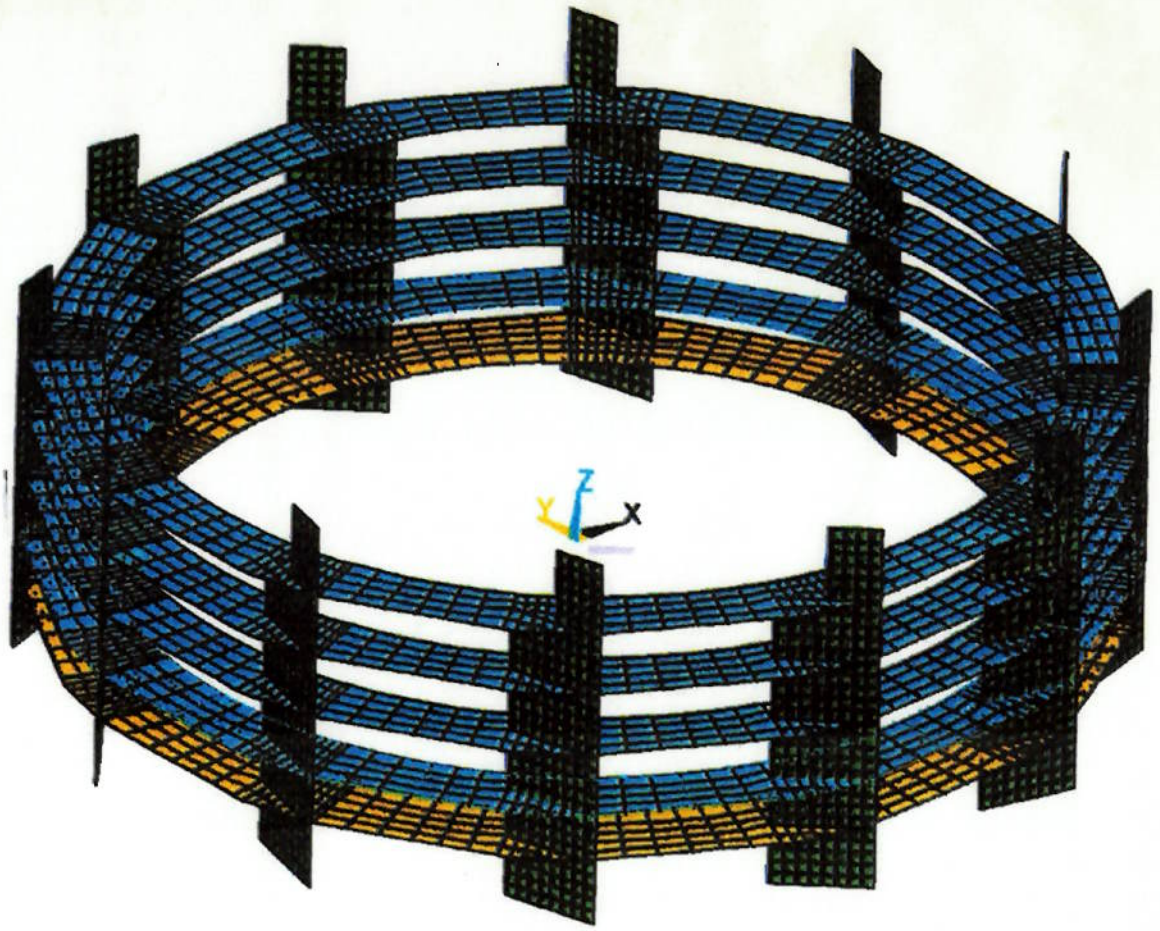


Figura 7: A geometria completa

Um modelo completo, se fosse usado no lugar de um segmento apenas, iria apenas acarretar um tempo de processamento e um espaço em disco maiores. Um modelo segmentado, devido ao fato de ser relativamente compacto, pode ter uma malha mais refinada, possibilitando a obtenção de resultados mais confiáveis (e aí sim, agregando valor à análise).

Os casos que justificam a utilização de um modelo completo estão reduzidos a dois grupos. O primeiro, é quando se tem um carregamento (ou quaisquer outras condições de contorno) assimétrico. O carregamento assimétrico mais comum existente na máquina é o chamado curto-circuito em meia roda polar: metade dos pólos do rotor entram em curto, e aparece uma atração entre pólos e enrolamento (estator). Essa atração só se dá em metade da circunferência, gerando uma resultante em uma direção radial que não tem correspondente na direção oposta. O segundo caso em que é justificado o não-uso da simetria é para fins

"didáticos". Para demonstrar as qualidades do programa a quem não tenha familiaridade com o produto hidrogerador, ou a quem não conheça o método dos elementos finitos, é aconselhável que se use o modelo completo. A visualização do componente a ser analisado fica bem mais fácil, abrindo assim um canal eficiente para que se "venda" a idéia da parametrização.

Como vimos, são poucos os casos em que se deva usar o modelo completo da carcaça. Porém, o caso desse trabalho se encaixa numa das exceções mostradas no parágrafo anterior: a da facilidade didática. Assim, este trabalho apresentará um modelo completo da carcaça, mesmo sabendo que, a rigor, ele será menos eficiente.

6.5. Carregamento e condições de contorno

A última tarefa do programa é, uma vez gerada a geometria do modelo, aplicar todas as condições de contorno que traduzam o ambiente em que a estrutura real estará operando. Feito isso, a parametrização terá cumprido sua missão e a análise poderá ser feita.

As condições de contorno aplicadas a esse modelo são: apoios, torque nominal da máquina, peso próprio e diferença de temperatura entre o enrolamento e o ambiente. Existem outros carregamentos aos quais a carcaça irá se sujeitar, porém estes estão fora do escopo desse trabalho, podendo ser incluídos em futuras versões do programa.

Uma macro de função específica cuida da aplicação dos carregamentos: carreg.mac. Por ser uma macro bastante seqüencial e de lógica simples, não há necessidade de se exibir aqui seu fluxograma.

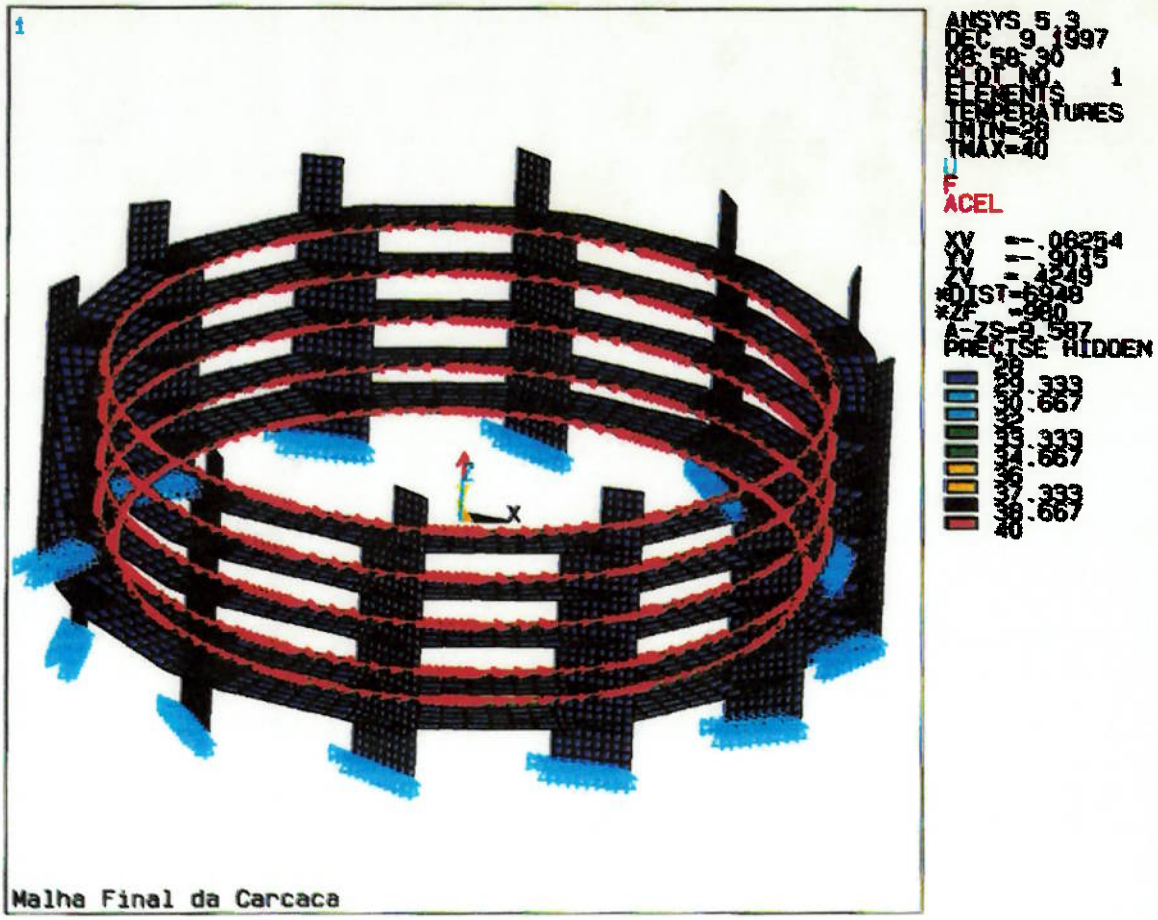


Figura 8: Modelo com carregamentos aplicados

6.5.1. Peso próprio:

O peso próprio em uma estrutura como a carcaça de um gerador é um dos carregamentos mais importantes. O peso de uma carcaça varia muito (assim como existem hidrogeradores de tamanhos bastante variados), porém é comum se encontrar carcaças com pesos na casa das dezenas de toneladas.

No ANSYS, o peso próprio é um carregamento de simples aplicação: basta especificar o valor da aceleração da gravidade. Com base nesse valor, nas densidades dos materiais e nas espessuras dos elementos (se forem elementos de chapa, área da seção transversal se forem de viga), o peso próprio pode ser avaliado

6.5.2. Torque na máquina:

Hidrogeradores de grande porte possuem rotores cujo diâmetro pode ultrapassar 10 metros. A potência gerada por esse tipo de máquina é

consideravelmente grande. Com as grandes massas e dimensões envolvidas, no entanto, essas máquinas operam a rotações relativamente baixas (na faixa dos 100 rpm). Partindo de uma potência alta e uma rotação baixa, e usando de relações físicas simples, é fácil deduzir que o torque atuante nessas máquinas será grande.

Na carcaça do gerador, o torque pode se manifestar como forças, na direção tangencial, aplicadas nos nós das linhas de raio interno das prateleiras. A macro *carreg.mac* seleciona os nós que fazem parte dessas linhas e, a partir do valor entrado como torque , da distância desses nós ao centro da carcaça e da quantidade de nós selecionados, distribui entre eles esse carregamento. A força tangencial é conseguida trocando-se o sistema de coordenadas para global cilíndrico.

6.5.3. Diferenças de temperatura:

Por ser uma estrutura que abriga circuitos elétricos e magnéticos de grandes potências, é de se esperar que, em operação, a carcaça de um hidrogerador esteja sujeita a temperaturas relativamente elevadas.

A temperatura, inclusive, é uma das maiores preocupações dos engenheiros que projetam essas máquinas. A ventilação do ar na máquina, que é proporcionada pelo próprio movimento das partes girantes, deve ser garantida. Muitas das características construtivas que os hidrogeradores possuem hoje vêm dessa necessidade de proporcionar ao ar uma fácil circulação através da máquina.

Na parametrização da carcaça, foi usada a seguinte simplificação: existe uma temperatura ambiente, que é a temperatura do ar, e os nós das linhas internas das prateleiras (que estão em contato direto com as partes eletromagnéticas), estão sujeitas a uma temperatura mais alta. Essa simplificação pode não traduzir fielmente o que acontece na estrutura real, mas está em favor da segurança quando prevê um acentuado gradiente de temperatura em uma parte da estrutura.

A implementação disso é simples: a temperatura do ar (temperatura uniforme) é entrada com um comando genérico. A temperatura das linhas internas das prateleiras é aplicada em seus nós. A seleção desses nós é feita da mesma maneira que para a colocação do torque (aliás, são exatamente os mesmos nós).

6.5.4. Apoios:

Por fim, restrições ao deslocamento de alguns nós devem ser impostas ao modelo para evitar movimentos de corpo rígido ou instabilidades.

A hipótese utilizada nesse cálculo é a seguinte: os únicos nós que possuem restrições são os nós da interface da coluna da carcaça com a base (coordenada z igual a **–dii-dib**).

Em todos esses nós, foram restringidos os movimentos nas direções x,y e z. Configura-se então um engastamento

É claro que existem outros modos de se apoiar a carcaça. Um engastamento puro e simples pressupõe uma superfície de apoio totalmente rígida. A carcaça está apoiada sobre concreto, que além de certos limites não pode ser considerado como tal. Nesse caso, o melhor seria se utilizar de uma base flexível, com elementos de mola (simulando a rigidez da base), e a esses elementos poderiam-se conectar nós imóveis.

A hipótese usada é simplificadora, uma vez que esse programa de parametrização é um processo inovador e piloto em elementos finitos. Em futuras versões do programa, quando a prática da implementação da parametrização já estiver mais consolidada, certamente muitas hipóteses assumidas aqui serão reconsideradas.

7. Conclusões

7.1. Bases conceituais do trabalho

A implementação da parametrização em cálculo por elementos finitos é uma prática que, ao longo de sua realização, vai se mostrando totalmente multidisciplinar (ao contrário do que possa parecer à primeira vista).

Do profissional (no meu caso), ou ao grupo de profissionais encarregados dessa tarefa (o que seria uma melhor maneira de abordar o problema), se exigem conhecimentos diversos. É vital que se conheça bem o método dos elementos finitos e os critérios para se obter uma análise confiável. Porém, se só se tem esse conhecimento, não se vai muito longe com a parametrização.

Um conhecimento do *produto* é também primordial, ou seja, o profissional deve estar familiarizado com o componente que ele está "parametrizando". O padrão deve ser totalmente entendido, para que por um lado não se compliquem demais as macros (com decisões que não precisariam ser tomadas e cálculos que não precisariam ser feitos) e por outro lado não se façam suposições precipitadas e incorretas (por exemplo, aquelas características que aparecem na maioria dos modelos mas *nem sempre* se verificam).

Outro elemento fundamental para o desenvolvimento da parametrização é a presença do *pensamento lógico-matemático*. As considerações que tornam o programa "esperto" (não se sujeita a fazer um modelo cujos parâmetros não são compatíveis entre si) e a estrutura das macros estão completamente comprometidas com a matemática por trás do modelo. E, como a parametrização está em forma de programas de computador, é fácil perceber a importância desse elemento nas bases desse trabalho.

Por fim, mas não menos importante que os anteriores, vem o conhecimento do *software* utilizado. As macros da parametrização estão escritas na linguagem interna do ANSYS. As tarefas que o programa realiza são nada mais nada menos que comandos do ANSYS escritos adequadamente e na sintaxe correta.

Esse foi, portanto, um vasto trabalho de estudo dos comandos do ANSYS, e consulta a seus manuais.

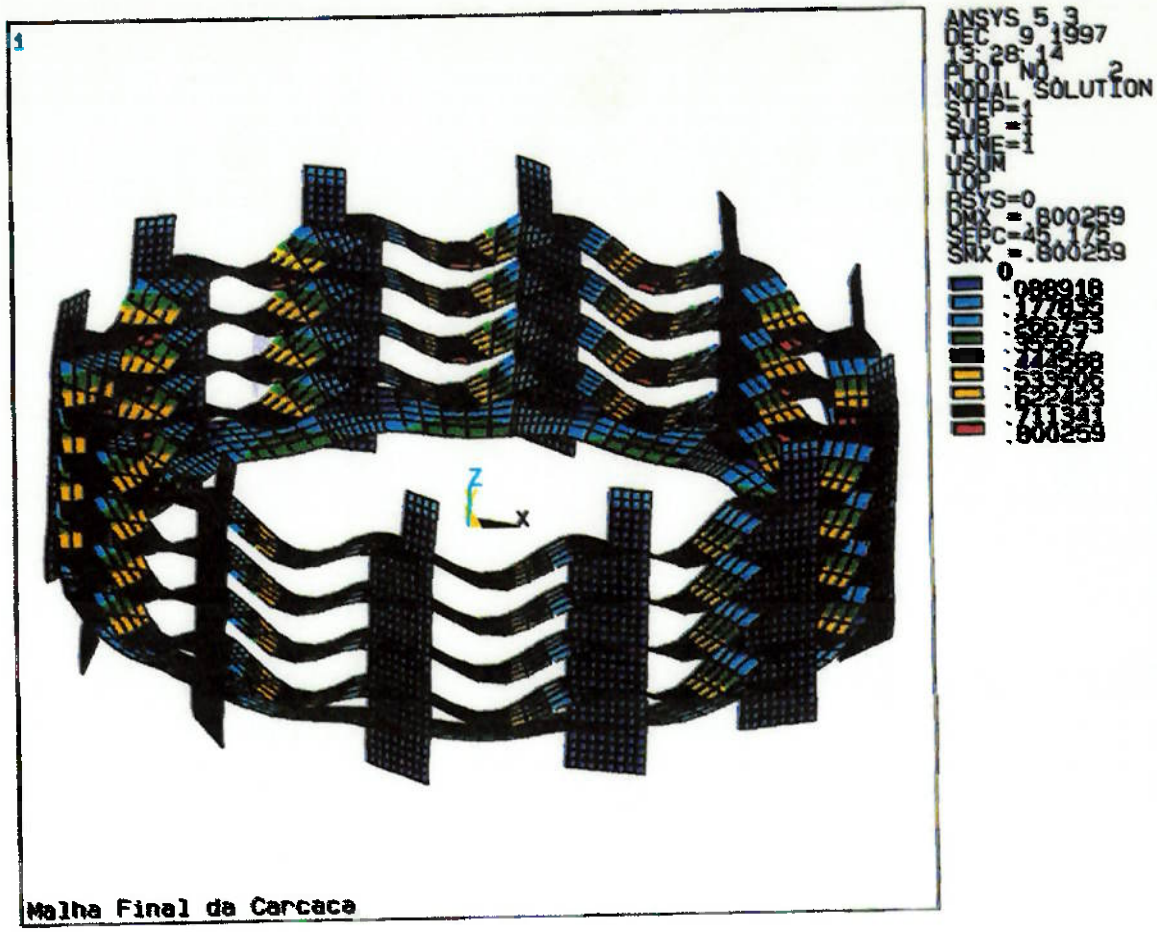


Figura 9: Exemplo de solução

7.2. As perspectivas da parametrização

Esse trabalho é apenas um marco inicial para todo um longo processo que deverá ter lugar daqui para a frente na engenharia de hidrogeradores: a automatização de projeto.

Esse processo é basicamente uma malha fechada, com o desenvolvimento de programas num ponto do ciclo, e a aferição dos resultados em outro ponto. A partir da verificação da confrontação dos resultados emitidos pelo programa com os valores reais das grandezas com a máquina em operação, é

possível fazer uma realimentação. O aperfeiçoamento do programa, com a adoção de hipóteses de cálculo mais realistas, por sua vez, trará resultados mais precisos e incentivará a parametrização dos outros componentes da máquina (já que sua viabilidade estará sendo comprovada).

Além da expansão para outros componentes da máquina, a parametrização pode, no futuro, assumir tarefas mais nobres, como a de "procurar" uma configuração ótima para a estrutura, através de combinações de resultados e estatística. Pode dar recomendações de projeto (talvez acerca de fixação de peças, transporte, içamento).

Em suma, pelas vantagens que a parametrização traz à engenharia, e à empresa em que for aplicada como um todo, é de se esperar que essa técnica tenha um futuro promissor, tendendo a penetrar com cada vez mais força na vida dos engenheiros de amanhã.

Anexo – Listagens das macros

Macro principal

```
---------------------------------------------- !
RTE 1: ENTRADA E VERIFICACAO DE VALIDADE DOS DADOS !
---------------------------------------------- !
p7
,msgpop,3
le,
es,new,carcaca,parm,/ansys/home/joao/
props
t
n,deg
=360/nc
=0
ri,gt,r1,then
    erro=1
    *endif
rii,gt,ri,then
    erro=1
    *endif
rc,lt,ri,then
    erro=1
    *endif
rc,gt,r2,then
    erro=1
    *endif
ri,gt,r2,then
    erro=1
    *endif
r2,lt,r1*cos(alfa/2),then
    erro=1
    *endif
r2,gt,r1/cos(alfa/2),then
    erro=1
    *endif
erro,eq,1,then
    *msg,error
    Os valores entrados tornam impossivel a obtencao da geometria desejada
    Verifique-os e corrija eventuais erros.
    *go,:Fim
if
,msgpop,2
---------------------------------- !
RTE 2: CONSTRUCAO DO ANEL SUPERIOR !
---------------------------------- !
sh
p7
,1
0,0,0
r1,0,0
r1,alfa,0
,11,0,2,1,3
0,(r2-r1*cos(alfa/2))/sin(alfa/2),0
,12,0,3,1,2
0,(r2-r1*cos(alfa/2))/sin(alfa/2),0
le,11,12
,1
ri,0,0
ri,alfa,0
,0
----------------------- !
RTE 3: PERFIL DA COLUNA !
----------------------- !
,1
rc,alfa/2,0
r2,alfa/2,0
```

```
,11,1,8,9,5
,lc/2,gama,0
,-lc/2,gama,0
s,0
0,11
le,11
e,8
e,9
rg,all
mp,all
t
---------------------------------------------------- !
RTE 4: VERIFICACAO DE CRUZAMENTO DA COLUNA E DO ANEL !
---------------------------------------------------- !
fcruz,5,4,8,9
cruza,eq,0,then
    verifcruz,3,5,8,9
    *if,cruza,eq,1,then
            cruza=2
            k,10,xp,yp,0
    *endif
e
    k,10,xp,yp,0
if
---------------------------------------------------- !
RTE 5: LINHAS QUE COMPOE A GEOMETRIA BASICA          !
---------------------------------------------------- !
2
4
cruza,eq,1,then
    l,4,10
    l,10,5
    l,5,3
eif,cruza,eq,2,then
    l,4,5
    l,5,10
    l,10,3
eif,cruza,eq,0,then
    l,4,5
    l,5,3
if
7
,1
6
,0
cruza,eq,0,then
    l,9,8
e
    l,9,10
    l,10,8
if
t
---------------------------------------------------- !
RTE 6: CONSTRUCAO DA MALHA DE ELEMENTOS              !
---------------------------------------------------- !
m,real,1
ber,1
cruza,ne,0,then
    *if,gama,gt,45,then
            /gtype,1,keyp,0
            /gtype,1,line,0
            real,1
            malha_p
            *get,inc_n,node,0,num,max
```

```
            !lgen,npi,all,,,,,dpi!
            ngen,npi,inc_n,all,,,,,dpi
            gplot
            egen,npi,inc_n,all
            nsel,s,node,,2,inc_n,1
            esln,s,1
            lsel,s,line,,1,9,1
            !lgen,2,all,,,,,-dii
            ngen,2,npi*inc_n,all,,,,,-dii
            egen,2,npi*inc_n,all,,,,,1
            real,2
            malhainf
            nsel,all
            lsel,all
            esel,all
            gplot
            malha_c
            gplot
            *get,inc_n,node,0,num,max
            csys,1
            lgen,nc,all,,,,alfa
            ngen,nc,inc_n,all,,,,alfa
            gplot
            egen,nc,inc_n,all
            /gtype,1,keyp,1
            /gtype,1,line,1
            nummrg,node
    *endif
if
le,Malha Final da Carcaca
t
---------------------------------------------------- !
RTE 7: APLICACAO DO CARREGAMENTO E APOIOS            !
---------------------------------------------------- !
eg
,u,,1
,acel,,1
,f,,1
t
---------------------------------------------------- !
RTE 8: SOLUCAO                                       !
---------------------------------------------------- !
,continue,O modelo esta pronto para ser analisado. Deseja prosseguir (s/n)
continue,eq,'s',then
    finish
    /solution
    solve
if
---------------------------------------------------- !
```

Verifcruz.mac

```
------------------------------------------------------------------- !
                    MACRO Verifcruz.mac                             !
------------------------------------------------------------------- !
r: Joao Claudio Cotta Cardoso PG/HCT                                !
ta da 1a. versao: 16.07.1997                                        !
ta da 0 e ultima alteracao:                                         !
------------------------------------------------------------------- !
ta e uma macro auxiliar a macro Carcaca.mac. Sua funcao e verificar !
 duas linhas se cruzam. Os argumentos da macro sao os numeros dos   !
ypoints inicial e final de cada uma das linhas.                     !
------------------------------------------------------------------- !

,0
,0
a=0
,xa,kp,arg1,loc,x
,ya,kp,arg1,loc,y
,xb,kp,arg2,loc,x
,yb,kp,arg2,loc,y
,xc,kp,arg3,loc,x
,yc,kp,arg3,loc,y
,xd,kp,arg4,loc,x
,yd,kp,arg4,loc,y

rificacao se uma das linhas e vertical, calculo das coordenadas do ponto !
 cruzamento (xp,yp)                                                     !

=0
xb,eq,xa,then
    vert=1
    xp=xa
    yp=yc+(yd-yc)*(xp-xc)/(xd-xc)
elif
xd,eq,xc,then
    vert=1
    xp=xc
    yp=ya+(yb-ya)*(xp-xa)/(xb-xa)
elif
vert,eq,0,then
    delta_ab=(yb-ya)/(xb-xa)
    delta_cd=(yd-yc)/(xd-xc)
    xp=(yc-ya-delta_cd*xc+delta_ab*xa)/(delta_ab-delta_cd)
    yp=ya+(yb-ya)*(xp-xa)/(xb-xa)
if

rificacao do cruzamento das linhas. Se elas se cruzam, yp deve estar den-
o dos intervalos de y de cada linha. Se a soma das diferencas entre yp e
 valores de y dos extremos da linha for igual a diferenca entre os valo-
s de y dos extremos, entao yp esta entre os y dos extremos.

abs(yp-ya)+abs(yp-yb),eq,abs(yb-ya),then
    *if,abs(yp-yc)+abs(yp-yd),eq,abs(yd-yc),then
            cruza=1
    *endif
if
cruza,eq,1,then
    *msg,info
    As linhas se cruzam.
e
    *msg,info
    As linhas NAO se cruzam.
if
```

Malha_p.mac

```
!-------------------------------------------------------------------
finicao dos parametros e numero de divisoes das linhas             !
                                                                   !
init,lncol,lnext:   listas de nos das linhas internas, externas e  !
                    da coluna                                      !
:   comprimento das linhas                                         !
i:  nos-chave para os elementos triangulares                       !
!-------------------------------------------------------------------
,shell63
20
50
,cl,array,9
cont,1,9
    *get,cl(cont),line,cont,leng
do
col=nint(cl(8)/tec)
ext=nint(cl(2)/tep)
rad=nint((r1-ri)/tep)
,lnint,array,div_col+2*div_ext+4
,lncol,array,div_col+1
,lnext,array,div_col+2*div_ext+4
,lninf,array,div_rad+1
,tri,array,div_rad+1
,lnatual,array,div_rad+1
,lnanter,array,div_rad+1
!-------------------------------------------------------------------
iacao dos nos sobre as linhas externas e coluna.                   !
ualizacao das listas.                                              !
!-------------------------------------------------------------------
,2,2
eat,9,1,1
t(1)=7
t(2)=6
l(1)=10
t(2)=2
            !! Coluna !!
real=cl(8)/div_col
,11,0,10,9,4
cont,1,div_col-1
    n,,cont*tec_real,0,0
    *get,ult_n,node,0,num,max
    lncol(cont+1)=ult_n
do
l(div_col+1)=9
nint(cl(9)/tec),gt,1,then
    div_fora=nint(cl(9)/tec)
    tec_fora=cl(9)/div_fora
    *do,cont,1,div_fora-1
            n,,-1*cont*tec_fora,0,0
    *enddo
if
            !! Linhas Externas !!
1
t(1)=3
nint(cl(5)/tep),gt,1,then
    div1=nint(cl(5)/tep)
    tep1=cl(5)/div1
    *if,cruza,eq,2,then
            cskp,12,0,3,10,7
    *else
            cskp,12,0,3,5,7
    *endif
    *do,cont,1,div1-1
            aux=aux+1
```

```
            n,,cont*tep1,0,0
            *get,ult_n,node,0,num,max
            lnext(aux)=ult_n
    *enddo
if
cruza,eq,2,then
    p1=10
    p2=5
    te=tec
    auxcol=aux
e
    p1=5
    p2=10
    te=tep
    auxcol=0
if
aux+1
t(aux)=p1
nint(cl(4)/te),gt,1,then
    div1=nint(cl(4)/te)
    tep1=cl(4)/div1
    cskp,13,0,p1,p2,7
    *do,cont,1,div1-1
            aux=aux+1
            n,,cont*tep1,0,0
            *get,ult_n,node,0,num,max
            lnext(aux)=ult_n
    *enddo
if
auxcol,eq,0,then
    auxcol=aux
if
aux+1
t(aux)=p2
nint(cl(3)/tec),gt,1,then
    *if,nint(cl(3)/tec),gt,div_col-aux+auxcol+1,then
            div1=div_col-aux+auxcol+1
            cskp,14,0,p2,4,7
            *do,cont,1,div1
                    aux=aux+1
                    n,,cont*tec,0,0
                    *get,ult_n,node,0,num,max
                    lnext(aux)=ult_n
            *enddo
            *if,nint((cl(3)-div1*tec)/tep),gt,1,then
                    div2=nint((cl(3)-div1*tec)/tep)
                    tep1=(cl(3)-div1*tec)/div2
                    *do,cont,1,div2-1
                            aux=aux+1
                            n,,cl(2)-div1*tec-cont*tep1,0,0
                            *get,ult_n,node,0,num,max
                            lnext(aux)=ult_n
                    *enddo
            *endif
            aux=aux+1
            lnext(aux)=4
            div1=nint(cl(2)/tep)
            tep1=cl(2)/div1
            cskp,15,0,4,2,7
            *do,cont,1,div1-1
                    aux=aux+1
                    n,,cont*tep1,0,0
                    *get,ult_n,node,0,num,max
                    lnext(aux)=ult_n
```

```
            *enddo
            aux=aux+1
            lnext(aux)=2
    *else
            div1=nint(cl(3)/tec)
            tep1=cl(3)/div1
            cskp,14,0,p2,4,7
            *do,cont,1,div1-1
                    aux=aux+1
                    n,,cont*tep1,0,0
                    *get,ult_n,node,0,num,max
                    lnext(aux)=ult_n
            *enddo
            aux=aux+1
            lnext(aux)=4
            cskp,15,0,4,2,6
            div1=div_col-aux+auxcol+1
            *do,cont,1,div1
                    aux=aux+1
                    n,,cont*tec,0,0
                    *get,ult_n,node,0,num,max
                    lnext(aux)=ult_n
            *enddo
            *if,nint((cl(2)-div1*tec)/tep),gt,1,then
                    cskp,16,0,2,4,6
                    div2=nint((cl(2)-div1*tec)/tep)
                    tep1=(cl(2)-div1*tec)/div2
                    *do,cont,1,div2-1
                            aux=aux+1
                            n,,cl(2)-div1*tec-cont*tep1,0,0
                            *get,ult_n,node,0,num,max
                            lnext(aux)=ult_n
                    *enddo
            *endif
            aux=aux+1
            lnext(aux)=2
    *endif
if
-------------------------------------------------------------- !
terminacao dos nos vertices de elementos triangulares          !
-------------------------------------------------------------- !
,1
,1
=2
1
((r2+r1)/2-ri)/div_rad
no=ter
nant=0
cont,2,div_col+1
    *get,aux,node,lncol(cont),loc,x
    distnant=distno
    distno=abs(aux-ri-(div_rad-trn)*ter)
    *if,distno,gt,distnant,then
            tri(trn)=lncol(cont-1)
            trn=trn+1
            distno=abs(aux-ri-(div_rad-trn)*ter)
    *endif
do
distno,lt,ter/3,then
    tri(trn)=9
if
,0
,0
t
```

```
---------------------------------------------------------------------- !
nstrucao dos elementos                                                 !
t: variavel que informa em que lista estao os nos externos             !
t=1 linha externa, sit=2 coluna, sit=3 linha externa inferior          !
s10: posicao do no 10 na lista de nos da linha externa                 !
os: numero de nos entre os dois extremos                               !
iang: numero de triangulos que ja apareceram, na 1a. parte             !
i2: numero de triangulos que ja apareceram, na 2a. parte               !
oinf: numero de nos a esquerda da coluna ao fim da 1a. parte           !
---------------------------------------------------------------------- !
,ult_n,node,0,num,max
,7,3,div_rad-1,ult_n+1,1
ual(1)=7
cont,1,div_rad-1
    lnatual(cont+1)=ult_n+cont
do
ual(div_rad+1)=3
--------------      Parte 1         ---------------------------!
1
=div_rad-1
ng=0
cont,2,100
    *if,sit,eq,3,exit
    *if,sit,eq,1,then
            noext=lnext(cont)
    *else
            noext=lncol(cont+1-pos10)
    *endif
    *do,cont1,1,nnos+2
            lnanter(cont1)=lnatual(cont1)
    *enddo
    csys,1
    dsys,1
    *get,aux,node,noext,loc,y
    n,,ri,aux,0
    csys,0
    dsys,0
    *if,noext,eq,tri(triang+1),then
            nnos=nnos-1
    *endif
    *get,ult_n,node,0,num,max
    *if,nnos,ne,0,then
            fill,ult_n,noext,nnos,ult_n+1,1
    *endif
    *do,cont1,1,nnos+1
            lnatual(cont1)=ult_n+cont1-1
    *enddo
    lnatual(nnos+2)=noext
    *do,cont1,1,nnos+1
            e,lnatual(cont1),lnanter(cont1),lnanter(cont1+1),lnatual(cont1
    *enddo
    *if,noext,eq,tri(triang+1),then
            triang=triang+1
            e,lnatual(nnos+2),lnanter(nnos+2),lnanter(nnos+3)
    *endif
    *if,noext,eq,10,then
            pos10=cont
            sit=2
    *endif
    *if,noext,eq,9,then
            *do,cont1,1,nnos+2
                    lninf(cont1)=lnatual(cont1)
            *enddo
            pos9=cont
```

```
            sit=3
            nnoinf=nnos+2
    *endif
do
-------------       Parte 2         ------------------------------- !
,lnext(pos10+1),lncol(2)
=0
=1
ual(1)=lncol(2)
ual(2)=lnext(pos10+1)
cont,2,100
    *do,cont1,1,nnos+2
            lnanter(cont1)=lnatual(cont1)
    *enddo
    *if,lnanter(1),eq,tri(tri2),then
            *if,tri2,lt,triang,then
                    nnos=nnos+1
            *endif
    *endif
    lnatual(1)=lncol(cont+1)
    *if,nnos,gt,0,then
            *get,ult_n,node,0,num,max
            fill,lncol(cont+1),lnext(cont+pos10),nnos,ult_n+1,1
            *do,cont1,1,nnos
                    lnatual(cont1+1)=ult_n+cont1
            *enddo
    *endif
    lnatual(nnos+2)=lnext(cont+pos10)
    *do,cont1,1,tri2
            e,lnatual(cont1),lnanter(cont1),lnanter(cont1+1),lnatual(cont1
    *enddo
    *if,lnanter(1),eq,tri(tri2),then
            *if,tri2,lt,triang,then
                    e,lnatual(tri2+1),lnanter(tri2+1),lnatual(tri2+2)
            *endif
            tri2=tri2+1
    *endif
    *if,lnatual(1),eq,9,exit
do
cont,1,nnos+1
    lninf(nnoinf+cont)=lnatual(cont+1)
do
--------------      Parte 3         ------------------------------- !
cont,1,div_rad+1
    lnatual(cont)=lninf(cont)
do
cont,1,10
    *do,cont1,1,div_rad+1
            lnanter(cont1)=lnatual(cont1)
    *enddo
    *if,lnext(pos9+cont),ne,2,then
            csys,1
            dsys,1
            *get,aux,node,lnext(pos9+cont),loc,y
            n,,ri,aux,0
            csys,0
            dsys,0
            *get,noint,node,0,num,max
    *else
            noint=6
    *endif
    lnatual(1)=noint
    *get,ult_n,node,0,num,max
    fill,noint,lnext(pos9+cont),div_rad-1,ult_n+1,1
```

```
    *do,cont1,2,div_rad
            lnatual(cont1)=ult_n+cont1-1
    *enddo
    lnatual(div_rad+1)=lnext(pos9+cont)
    *do,cont1,1,div_rad
            e,lnatual(cont1),lnanter(cont1),lnanter(cont1+1),lnatual(cont1
    *enddo
    *if,lnext(pos9+cont),eq,2,exit
do
------------------------------------------------------------------- !
m.                                                                  !
------------------------------------------------------------------- !
le,11,16
t(1)=
l(1)=
t(1)=
)=
1)=
ter(1)=
ual(1)=
f(1)=
t
```

Malha_c.mac

```
------------------------------------------------------------------- !
nstrucao da lista de nos do perfil da coluna                        !
                                                                    !
col:numero de elementos do perfil da coluna                         !
------------------------------------------------------------------- !
,lnc,array,5*div_col
l=0
cont,1,div_col+1
    cont1=cont1+1
    lnc(cont)=lncol_f(cont)
    *if,lncol_f(cont),eq,10,exit
do
l=cont1
cont,1,div_col+1
    nncol=nncol+1
    lnc(cont+cont1)=lncol(cont+1)
    *if,lncol(cont+1),eq,9,exit
do
l(1)=
l_f(1)=
------------------------------------------------------------------- !
nstrucao dos nos e elementos da coluna                              !
------------------------------------------------------------------- !
,3
,lnanter,array,2*nncol
,lnatual,array,2*nncol
----------- Entre as prateleiras intermediarias ---------------!
cont,1,nncol
    lnatual(cont)=lnc(cont)
do
t=nint(dpi/tec)
l=dpi/nnalt
,11,0,8,10,7
nprat,1,npi-1
    *do,cont,1,nnalt
            dsys,11
            *get,ult_n,node,0,num,max
            dsys,0
            *if,cont,ne,nnalt,then
                    *do,cont1,1,nncol
                            lnanter(cont1)=lnatual(cont1)
                            *get,aux,node,lnanter(cont1),loc,x
                            n,ult_n+cont1,aux,0,-cont*altel-dpi*(nprat-1)
                            lnatual(cont1)=ult_n+cont1
                    *enddo
            *else
                    *do,cont1,1,nncol
                            lnanter(cont1)=lnatual(cont1)
                            lnatual(cont1)=lnc(cont1)+inc_n*nprat
                    *enddo
            *endif
            *do,cont1,1,nncol-1
                    e,lnatual(cont1),lnanter(cont1),lnanter(cont1+1),lnatu
            *enddo
    *enddo
do
---------------------- Acima das prateleiras -------------------------!
a=nint(acs/tec)
lcs=acs/nncsa
l=nint(lcs/tec)
elcs=lcs/nncsl
cont,1,nncsa+1
    dsys,11
    *get,ult_n,node,0,num,max
```

```
    dsys,0
    *do,cont1,1,nncsl+1
            lnanter(cont1)=lnatual(cont1)
            n,ult_n+cont1,((cont1-1)*largelcs),0,-cont*altelcs-(npi-1)*dpi
            lnatual(cont1)=ult_n+cont1
    *enddo
    *do,cont1,1,nncsl
            e,lnatual(cont1),lnanter(cont1),lnanter(cont1+1),lnatual(cont1
    *enddo
do
--------------- Entre as intermediarias e a inferior -----------------!
t=nint(dii/tec)
l=dii/nnalt
cont,1,nncol
    lnatual(cont)=lnc(cont)
do
cont,1,nnalt
    *get,ult_n,node,0,num,max
    *if,cont,ne,nnalt,then
            *do,cont1,1,nncol
                    lnanter(cont1)=lnatual(cont1)
                    *get,aux,node,lnanter(cont1),loc,x
                    n,ult_n+cont1,aux,0,cont*altel
                    lnatual(cont1)=ult_n+cont1
            *enddo
    *else
            *do,cont1,1,nncol
                    lnanter(cont1)=lnatual(cont1)
                    lnatual(cont1)=lnc(cont1)+inc_n*npi
            *enddo
    *endif
    *do,cont1,1,nncol-1
            e,lnatual(cont1),lnanter(cont1),lnanter(cont1+1),lnatual(cont1
    *enddo
do
----------------- Da prat. inferior a base ------------------------ !
t=nint(dib/tec)
l=dib/nnalt
cont,1,nnalt
    *get,ult_n,node,0,num,max
    *do,cont1,1,nncol
            lnanter(cont1)=lnatual(cont1)
            *get,aux,node,lnanter(cont1),loc,x
            n,ult_n+cont1,aux,0,cont*altel+dii
            lnatual(cont1)=ult_n+cont1
    *enddo
    *do,cont1,1,nncol-1
            e,lnatual(cont1),lnanter(cont1),lnanter(cont1+1),lnatual(cont1
    *enddo
do
------------------------------------------------------------------- !
,0
,0
1)=
ter(1)=
ual(1)=
```

Malhainf.mac

```
---------------------------------------------------------- !
cro MALHAINF - 25.09.97                                    !
---------------------------------------------------------- !
ta macro constroi os elementos da parte interior da prate- !
ira inferior.                                              !
---------------------------------------------------------- !
,lnanter,array,5*div_col
,lnatual,array,5*div_col
=0
cont,1,1000
    *if,lnint(cont),eq,0,exit
    nnii=nnii+1
    lnatual(cont)=lnint(cont)+npi*inc_n
do
ii=nint(1.75*(ri-rii)/tep)
=(ri-rii)/div_ii
cont,1,div_ii
    *do,cont1,1,nnii
            lnanter(cont1)=lnatual(cont1)
    *enddo
    *get,ult_n,node,0,num,max
    csys,1
    dsys,1
    *do,cont1,1,nnii
            *get,aux,node,lnanter(cont1),loc,y
            n,ult_n+cont1,ri-cont*teii,aux,-dii
            lnatual(cont1)=ult_n+cont1
    *enddo
    csys,0
    dsys,0
    *do,cont1,1,nnii-1
            e,lnatual(cont1),lnanter(cont1),lnanter(cont1+1),lnatual(cont1
    *enddo
do
ter(1)=
ual(1)=
```

Carreg.mac

```
es,change,carreg,parm,/ansys/home/joao/
-------------------------------------!
      Apoios / Restricoes            !
-------------------------------------!
* Piso *** !
,s,loc,z,-dii-dib
l,uz,0
,all
-------------------------------------!
    Peso Proprio (gravidade)         !
-------------------------------------!
,0,0,9.81
-------------------------------------!
          Torque Nominal             !
-------------------------------------!
lecao dos nos das prats. intermediarias!
,1
,s,real,,1
,s
,r,loc,x,ri
at,all
,nosel,node,0,count
visao do torque pelos nos, e calculo da forca por no!
at=nosel/npi
_no=torqnom/nnprat*(npi+1)
o=torq_no*1000/ri
l,fy,ft_no
ecao dos nos da prateleira inferior !
,all
,all
,s,real,,2
,s
,r,loc,x,rii
at,all
licacao do torque na prat. inferior !
o=torq_no*1000/rii
l,fy,ft_no
,all
,all
```

Const.mac

```
es,change,const,parm,/ansys/home/joao/
epint
epinf
ecol
```

Material.mac

```
es,change,material,parm,/ansys/home/joao/
X,1,e
ENS,1,dens
UXY,1,pois
```

Edit_par.mac

```
,msgpop,1
,note,
ar parametros referentes a: %/(1) Geometria %/(2) Carregamento %/ &
Espessuras das chapas %/(4) Propriedades do material %/ %/ &
te o numero apropriado na caixa de dialogo.
,ednum,Que parametros deseja editar?,0
ednum,eq,1,then
    /sys,dtpad /ansys/home/joao/carcaca.parm
if
ednum,eq,2,then
    /sys,dtpad /ansys/home/joao/carreg.parm
if
ednum,eq,3,then
    /sys,dtpad /ansys/home/joao/const.parm
if
ednum,eq,4,then
    /sys,dtpad /ansys/home/joao/material.parm
if
,msgpop,2
```