10 (dez)

Sérgio Eduardo Macêdo Rezende

Método de digitalização de sólidos aplicado ao CAD/CAE

Trabalho de Formatura
Apresentado à Escola
Politécnica da Universidade de
São Paulo

Orientador:
Prof. Dr. Emílio Carlos Nelli Silva

São Paulo

2001

ÍNDICE

1. INTRODUÇÃO

Hoje em dia muitas peças complexas em Engenharia são desenhadas em programas do tipo CAD onde podem ser analisadas e modificadas. Além disso, devido ao avanço da tecnologia, com o objetivo de obter equipamentos mais precisos, resistentes e com custo menor são usados programas de CAE. Neste tipo de software o comportamento destas peças é simulado usando o método dos elementos finitos que calcula o desempenho, durabilidade, resistência e outras propriedades da peça. Entretanto há muitas dificuldades enfrentadas desde a modelagem da peça num programa de CAD até a interface deste com o CAE.

O primeiro problema enfrentado diz respeito a modelar (desenhar) esta peça complexa no programa de CAD. Caso o objetivo seja melhorar o desempenho de um determinado tipo de peça já existente é necessário inicialmente modelá-la no CAD. Além disso, muitas peças possuem tantos detalhes que desenhá-la em computador levaria meses de trabalho podendo tornar o processo muito lento ou inviável.

Outra dificuldade está na troca de arquivos entre CAD e CAE. Como o desenho feito no primeiro se encontra na forma vetorizada, há muitas funções, malhas, splines criadas pelo programa CAD que são armazenadas em um código específico daquele programa. Quando este código complexo é recebido pelo CAE muitos detalhes da peça tais como ressaltos, orifícios, pequenas curvaturas são ignoradas, simplificadas ou mesmo não compreendidas. Dependendo das simplificações os resultados podem ser bastante diferentes dos reais.

Outro problema está no fato de que, para uma futura otimização da peça alterando sua forma, é preciso modificá-la no CAD (o que seria também muito trabalhoso) e passar novamente pelas dificuldades já citadas para enviar ao CAE.

A proposta desta pesquisa para solucionar o problema é obter o modelo de CAE sólido mediante o processo de digitalização e discretização do sólido inicial. Digitalizar significa converter uma imagem bitmap em uma matriz de números onde cada pixel desta imagem será substituído por um número. Discretizar é o processo de, utilizando funções matemáticas, melhorar a resolução de uma imagem aumentando o número de pixels ou criar seções intermediárias de um sólido em estudo. A aplicação do método é exemplificado na figura a seguir.

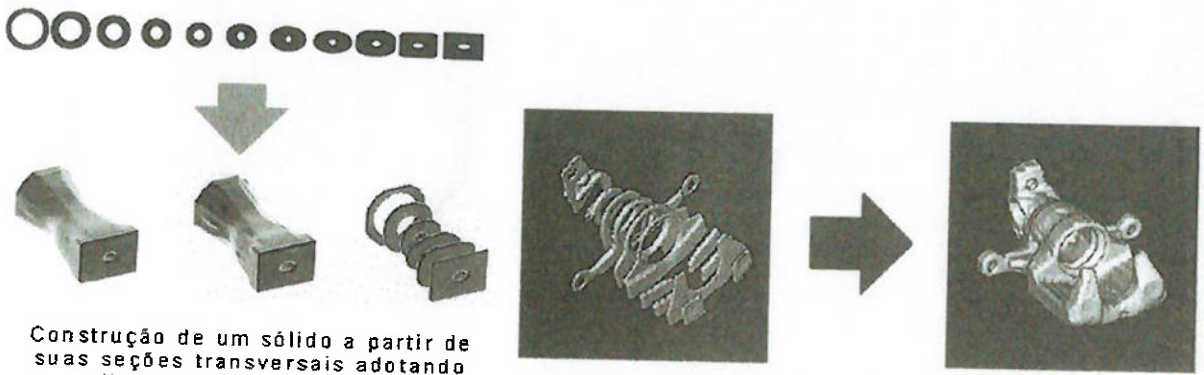


Figura 1 – Reconstrução da peça a partir de suas seções transversais.

A utilização de uma malha digitalizada vai gerar uma imprecisão na análise por elementos finitos (tensões e deformações) em relação a uma malha tradicional obtida num domínio com o contorno exato da peça. Entretanto, essa diferença está em torno de 4% à 5%. No entanto, certamente esse erro é muito menor do que o erro que ocorreria se fosse utilizado um modelo de CAE obtido com o método convencional (malhamento de um modelo de CAD), porque para obtenção desse modelo de CAE, considerando uma peça de geometria complexa (fundida ou forjada, por exemplo), seriam necessárias tantas simplificações no modelo CAD que o modelo de CAE obtido seria uma simplificação (às vezes muito grosseira) da peça real não sendo útil para representar o seu comportamento mecânico (concentração de tensões em pequenos detalhes, por exemplo) resultando em imprecisões maiores do que 4% ou 5% devido ao uso de uma malha digitalizada. Ou seja,

a construção de um modelo de CAE detalhista e "perfeito" que represente com fidelidade os contornos e detalhes de uma peça complexa (fundida ou forjada), usando o método convencional na construção de modelos, não pode ser obtido na prática. Isso porque além de levar meses para a sua construção, sendo muito custoso a mão-de-obra de CAD e CAE, existem todos os problemas de limitação dos geradores tridimensionais de malha que falham em gerar malhas em modelos tridimensionais de forma complexa com detalhes, e que acabam exigindo simplificações, muitas vezes grosseiras no modelo CAD para a obtenção do modelo CAE final, como já comentado no relatório. Assim, o custo, tempo e dificuldade desestimulam as indústrias (automotiva e máquinas ferramentas principalmente) a fazer simulações de CAE de suas peças mais complexas em geometria (por exemplo, bloco de motor fundido, caixa de transmissão fundida). A figura (2) ilustra a complexidade de uma caixa de transmissão fundida, cuja construção de um modelo CAE perfeito usando método convencional é extremamente custoso senão impossível.

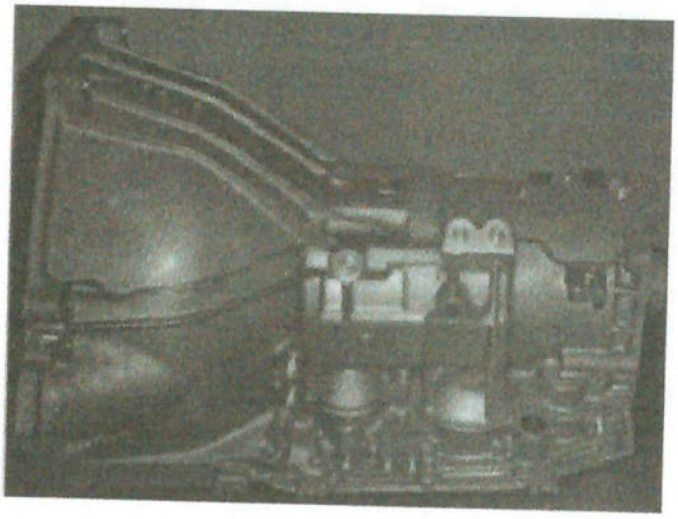
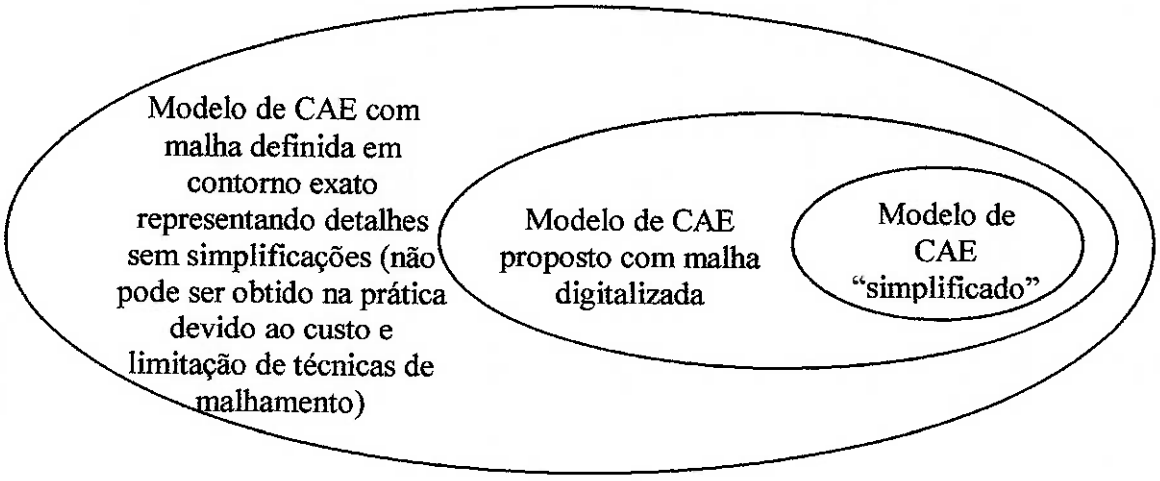


Figura 2 – Caixa de transmissão fundida.

Dessa forma, a indústria opta por não fazer ou fazer um modelo de CAE que embora consista num modelo com uma malha "regular" definida num contorno exato (não necessariamente igual ao contorno da peça, devido às simplificações comentadas acima), é um modelo muito simplificado em geometria devido à dificuldade de sua construção,

custo de obtenção (tempo e mão de obra), já comentados, e que induzirá a grandes imprecisões nos resultados de simulação da peça. Essa imprecisão é certamente maior do que a obtida com malha digitalizada, que ainda não é a solução ideal, mas já traz resultados bem melhores do que um modelo CAE muito simplificado. É importante ressaltar que esse problema ocorre apenas para peças de geometria complexa (fundidas ou forjadas, por exemplo). Para peças de geometria mais simples (usinadas por exemplo) o método convencional não apresenta problemas na geração de um modelo CAE, não se justificando nesse caso a utilização do método de digitalização proposto. O esquema da figura abaixo ilustra o conceito discutido em termos de abrangência da representatividade da peça real.



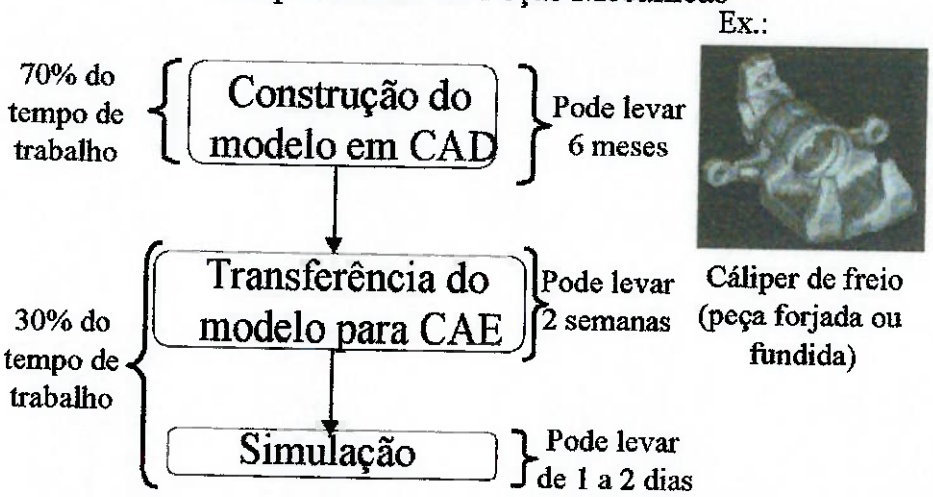


Figura 3 - Procedimento de modelagem computacional.

Já no método proposto, embora a malha seja digitalizada (ou "irregular" como mencionada) ela é obtida rapidamente (em menos de um dia, com o auxílio de um tomógrafo, como descrito), mantendo todos os detalhes da peça, sendo necessário para isso apenas controlar a resolução com que o tomógrafo obtém as imagens. Nesse sentido, o erro de 4% a 5% mencionado acima pode até ser reduzido, refinando-se a discretização da imagem bitmap das seções transversais e a do modelo final. Quanto maior o refinamento, mais próximo da peça real. A figura 4 ilustra as etapas na aplicação do método proposto e o tempo de duração das etapas.

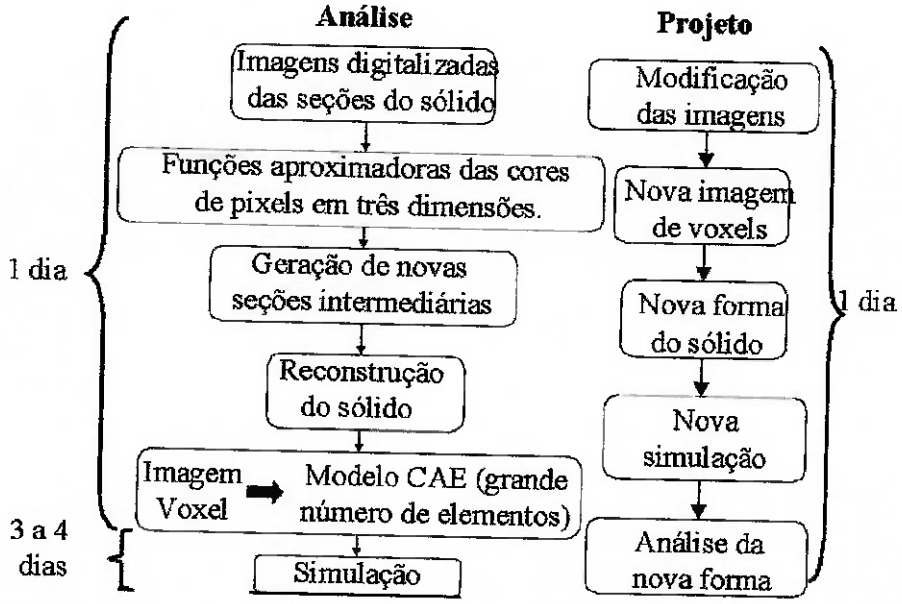


Figura 4 – Fluxograma de aplicação do método.

Além disso o custo em dinheiro também é menor. O custo de uma tomografia está em torno de R$300,00 a R$400,00 para uma peça de no máximo as dimensões de um corpo humano (existem tomógrafos maiores atualmente no meio industrial). Esse serviço pode ser contratado pela empresa (não é necessário ter um tomógrafo) saindo muito mais barato do que pagar salário para mão de obra de CAD/CAE durante 6 meses para construir uma única peça complexa, ou mesmo se essa mão de obra já está disponível, direcioná-la para um único trabalho por um longo tempo.

Portanto dedicou-se ao estudo inicial da reconstrução da imagem 3D de um sólido mediante as imagens de suas seções. Os resultados apresentados neste relatório mostram todos os processos de discretização, desde a digitalização das seções transversais do sólido (transformá-las em matrizes de zeros e uns) até a implementação da discretização com funções matemáticas para melhorar a resolução das seções e aumentar o número de seções transversais. Todos os programas foram feitos em linguagem C compatível com Windows e LINUX para verificar a eficácia do método de discretização para sólidos.

Para a discretização apresentar bons resultados foi necessário escolher a função matemática adequada cuja formulação será apresentada, bem como o motivo de sua escolha. Para isso várias funções de discretização foram comparadas considerando o domínio unidimensional. O método foi expandido para o domínio bidimensional onde foram discretizadas imagens simples, além de estudar a influência de parâmetros utilizados nas funções matemáticas.

As funções de discretização foram aplicadas em dois modelos, utilizando-se imagens de seções transversais, com o objetivo de se criar imagens de seções intermediárias. A saída consiste em um arquivo com elementos e nós para ser aberto em um programa de Elementos Finitos. Para a visualização do sólido discretizado foi utilizado o programa VTK implementado por outro aluno.

Os programas feitos em C foram aplicados em imagens de modelos reais de peças. Além disto realizou-se simulações destas peças em CAE para verificar os resultados relativos ao método de discretização.

2. FUNDAMENTOS TEÓRICOS

O objetivo da discretização é, em primeiro lugar, construir funções que passem próximas dos pontos dados e esta proximidade possa ser controlada de acordo com o caso estudado. No problema em estudo, os pontos correspondem a cores de pixels (em uma imagem) ou voxels (num sólido).

O principal critério na escolha do método de discretização é que os valores das funções não podem oscilar nas proximidades dos pontos, pois isto altera a forma inicial do sólido. A escolha do Método de Shephard [4, 9] será explicada posteriormente através da comparação com outros métodos de aproximação. A base teórica deste método é descrita a seguir.

Na figura (5) são apresentados pontos genéricos pelos quais a função aproximadora passará.

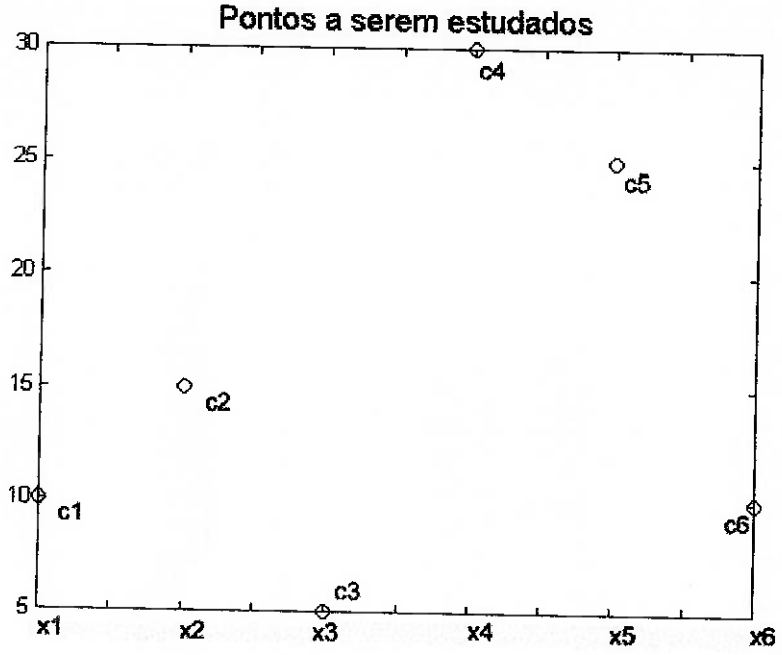


Figura 5 – Pontos genéricos a serem aproximados.

As funções aproximadoras usadas nesse trabalho são obtidas através do Método de Shephard [4, 9]. Este método é baseado na combinação dos valores que a função assume para cada ponto ($c_1, c_2, \ldots, c_n$) e um peso associado a cada ponto que faz com que a função aproximadora passe mais próximo dos valores desejados. O método será descrito em detalhe na Implementação numérica do método de digitalização. O formato da função é dado pela seguinte equação:

$$f(x)=c_1\phi_1(x)+c_2\phi_2(x)+\ldots+c_n\phi_n(x) \qquad (1)$$

Onde $c_1, c_2, \ldots, c_n$ correspondem aos valores da função $f(x)$ nos pontos $x_1, x_2, \ldots, x_n$ isto é, $f(x_1) = c_1$, $f(x_2) = c_2$, …, $f(x_n) = c_n$. Os termos $\phi_1(x)$, $\phi_2(x)$, …, $\phi_n(x)$ são chamados de funções de forma. Estas consistem na combinação de uma função polinomial multiplicada por um peso associado. A função de forma é dada por:

$$\phi_i(x)=[a_0(x)+x_i a_1(x)+x_i^2 a_2(x)+\ldots+x_i^n a_n(x)]w_i(x) \quad (i=1,2,\ldots,n) \qquad (2)$$

O termo $w_i(x)$ corresponde ao peso dado ao ponto i e será explicado posteriormente. Note que no método dos mínimos quadrados (MMQ) $w_i(x)=1$. O valor de n determina o grau do polinômio do método. A influência do grau deste polinômio será mostrada na análise dos resultados.

Por exemplo, para n=1 as funções de forma passam a ser lineares, com o seguinte formato:

$$\phi_i(x)=[a_0(x)+x_i a_1(x)]w_i(x) \quad (i=1,2,\ldots,n) \qquad (3)$$

Os valores de $a_0(x)$ e $a_1(x)$ segundo Shephard [4, 9] são obtidos ao se desenvolver a equação dada por:

$$\sum_{i=1}^{n}(f_0+f_1x_i)\phi_i(x)=f_0+f_1x=f(x) \qquad (4)$$

Onde f_0 e f_1 são coeficientes da função linear e serão cancelados no desenvolvimento da expressão (4). Para isso, o somatório de (4) é dado por:

$$(f_0+f_1x_1)(a_0+x_1a_1)w_1+(f_0+f_1x_2)(a_0+x_2a_1)w_2+\ldots+(f_0+f_1x_n)(a_0+x_na_1)w_n=f_0+f_1x \qquad (5)$$

Para simplificar a notação adotou-se: $a_0(x)=a_0$, $a_1(x)=a_1$ e $w_i(x)=w_i$. Efetuando as multiplicações e isolando f_0 e f_1 chega-se a:

$$f_0[(a_0w_1+a_1w_1x_1)+...+(a_0w_n+a_1w_nx_n)]+f_1[a_0w_1x_1+a_1w_1x_1^2+...+a_0w_nx_n+a_0w_nx_n^2]=f_0+f_1x \quad (6)$$

Simplificando f_0 e f_1, isolando a_0 e a_1, e igualando termo a termo obtém-se o seguinte sistema:

$$\begin{bmatrix} w_1+w_2+...+w_n & x_1w_1+x_2w_2+...+x_nw_n \\ x_1w_1+x_2w_2+...+x_nw_n & x_1^2w_1+x_2^2w_2+...+x_n^2w_n \end{bmatrix}\begin{Bmatrix} a_0(x) \\ a_1(x) \end{Bmatrix}=\begin{Bmatrix} 1 \\ x \end{Bmatrix} \quad (7)$$

A expressão (7) permite a obtenção de $a_0(x)$ e $a_1(x)$ presentes na expressão (3). Outro termo pertencente a equação (3) é $w_i(x)$ que corresponde ao peso associado para cada ponto e é dado por:

$$w_i(x)=\exp(-\alpha(x-x_i)) \quad (8)$$

Logo, quando o valor x da função está próximo de x_i o termo $x-x_i$ tende a zero e o peso w_i deste ponto tende a um (valor máximo do peso). Por outro lado, quanto mais distante estiver x de x_i , maior será o termo $x-x_i$ fazendo com que o peso w_i seja cada vez menor, tendendo a zero.

O fator α determina a proximidade em que a função passará em relação aos pontos desejados. Quanto maior o valor de α maior será a influência de $x-x_i$ forçando a função a passar mais próxima dos pontos dados. Este fato será demonstrado adiante.

3. IMPLEMENTAÇÃO NUMÉRICA DO MÉTODO DE DIGITALIZAÇÃO

Nos próximos sub-itens serão apresentados todos os resultados referentes a implementação e estudo do método de discretização. Primeiro será mostrada a aproximação unidimensional com o objetivo de comparar o método de Shephard [4, 9] com os métodos convencionais de aproximação justificando a sua escolha para o método. Em seguida, a função será aplicada para discretizar os modelos bidimensionais de imagens (discretização de seções) e tridimensionais (discretização de sólidos). Finalmente será explicado o método de conversão dos elementos constituintes do sólido para um formato de entrada a ser lido num programa de elementos finitos, além da aplicação da discretização em sólidos mais complexos.

3.1. Comparação do método de Shephard com métodos tradicionais considerando o domínio unidimensional.

Pretende-se, a partir de vários pontos dados definir qual será a melhor função que passará próxima destes pontos "acompanhando" a tendência destes. Além disso, a função não pode apresentar oscilações entre os pontos dados, para que os pontos intermediários (necessários na discretização) não assumam valores que prejudiquem a discretização do sólido. Para isso será feito o estudo das funções disponíveis que passam próximas aos pontos dados na figura (5).

Em primeiro lugar foi feita uma implementação do método dos mínimos quadrados. Como exemplo prático foram utilizados 6 pontos pelos quais deve-se passar uma função aproximadora. Assim, para uma aproximação melhor dos pontos em questão implementou-se o método dos mínimos quadrados (MMQ) com um polinômio de grau 4 na figura (6).

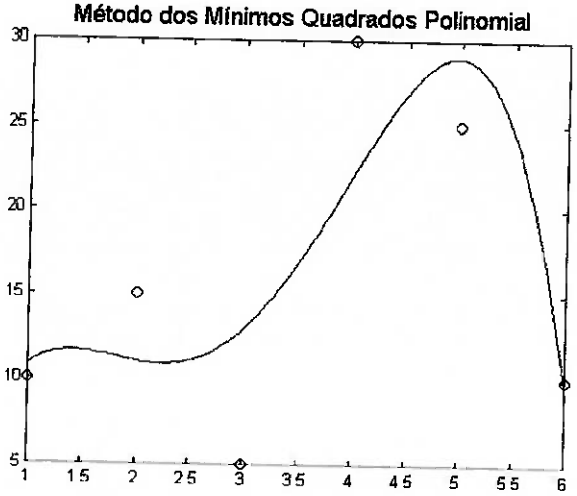


Figura 6 – Método dos mínimos quadrados com polinômio de grau 4.

Com o objetivo de melhorar os resultados, o método dos mínimos quadrados pode ser modificado de forma que se de ênfase para alguns pontos atribuindo a estes um determinado peso. Para se ter uma idéia melhor de como os pesos em pontos específicos fará com que a curva se aproxime destes a figura (7) mostra a mesma aproximação, entretanto os pontos 2 e 3 possuem um peso maior que os outros apresentando gráficos diferentes da figura (6).

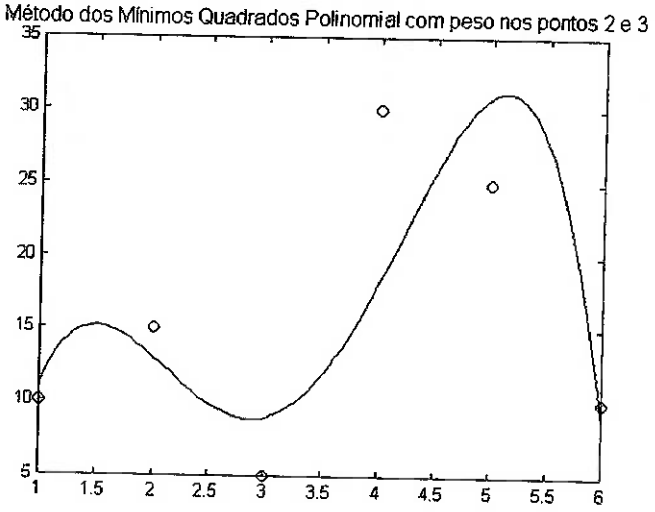


Figura 7 - MMQ com peso nos pontos 2 e 3.

Fazendo uma comparação com a figura (6) pode-se observar que a aproximação da curva aos pontos 2 e 3 é maior. Entretanto, apesar da aplicação dos pesos, a curva ainda passa longe dos pontos 4 e 5, fato que prejudica no trabalho de discretização.

Uma forma utilizada para forçar que a curva de aproximação passe pelos pontos dados é chamada de interpolação polinomial cujo resultado é mostrado na figura (8).

Pode-se observar que a curva aproximadora passa obrigatoriamente por todos os pontos dados entretanto um problema observado é a grande oscilação desta curva, principalmente entre os pontos dados, não representando a tendência da curva. Este fato é de grande prejuízo para a discretização que precisará dos pontos intermediários.

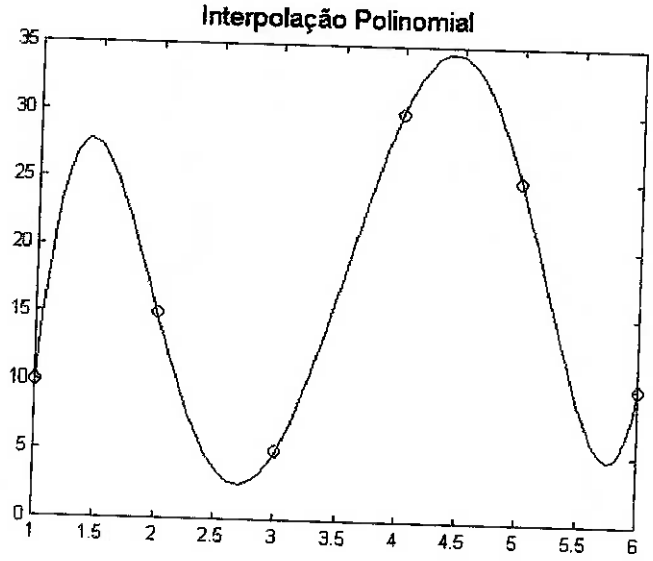


Figura 8 – Interpolação polinomial.

Outra solução que será analisada consiste no uso de curvas do tipo "spline". O resultado pode ser observado na figura (9). Verifica-se que, neste método a curva aproximadora também passa por todos os pontos dados e a transição entre estes apresenta curvas menos longas. Entretanto, este tipo de aproximação fornece curvas com transição curvilínea entre os pontos sendo que o ideal é se ter uma transição linear. Isto pode ser observado principalmente entre os pontos 1 e 2.

Para uma melhor discretização a curva que passa próximo a estes pontos deve se aproximar para uma reta e não para uma curva que oscila.

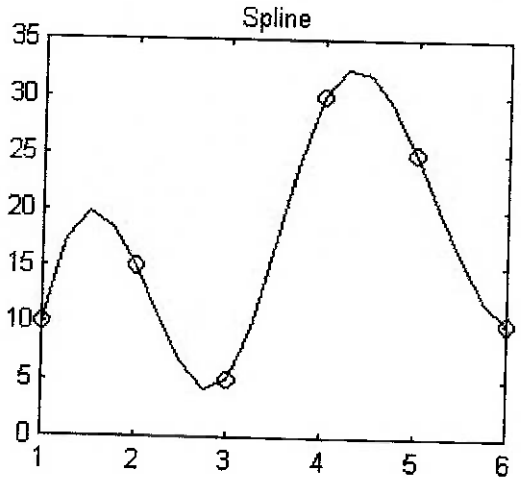


Figura 9 – Spline.

Finalmente a figura (10) mostra os resultados do Método de Shephard usando um polinômio de grau 1 (linear) e outro de grau 2.

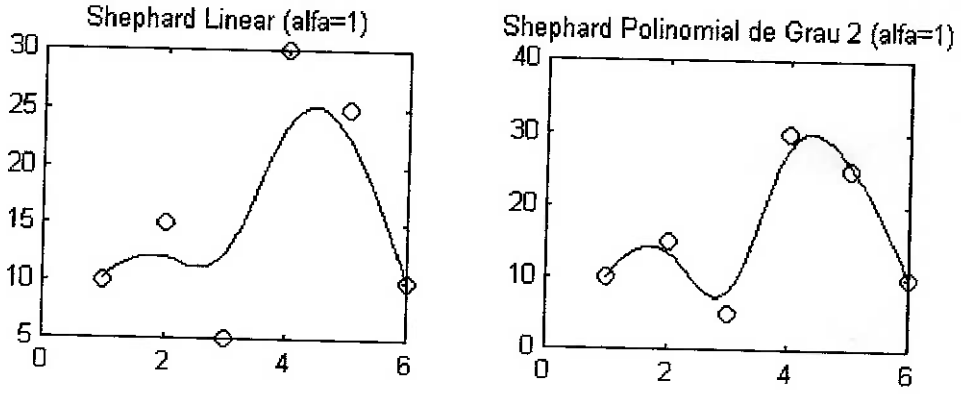


Figura 10 – Método de Shephard linear e polinomial com peso baixo (alfa = 1).

Na figura (10) o uso de um polinômio linear com baixo peso (alfa=1) como base apresenta uma curva de aproximação mais distante que o polinomial de grau 2 com o mesmo peso que demonstrou melhor aproximação dos pontos segundo o critério definido.

O próximo passo a ser dado será aumentar o peso associado a todos os pontos para "forçar" a curva a passar mais próxima aos pontos dados. O valor do peso atribuído (alfa) será igual a 2,5 a ambos os gráficos (linear e polinomial). Os resultados são mostrados na figura (11). Pode-se observar que para este caso a função aproximadora no caso linear praticamente une cada par de pontos próximos com uma reta apresentando

uma curva suave ao passar próximo a estes pontos. No caso da função polinomial as curvas não apresentam um crescimento linear entre os pontos, o que não é interessante como mostrado adiante.

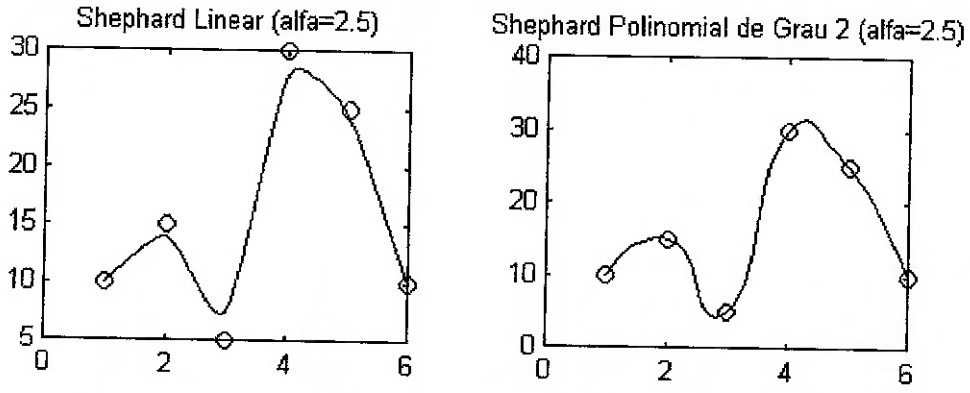


Figura 11 - Método de Shephard com peso maior (alfa = 2,5).

A figura (12) apresenta valores que mostram a grande eficácia do Método de Shephard com base num polinômio de grau um (linear) bastando para isto utilizar um peso (alfa) igual a 4,0. Os resultados do linear são muito mais representativos da aproximação desejada que a função polinomial de grau 3 mostrada na figura (12), além do número de operações matemáticas ser menor.

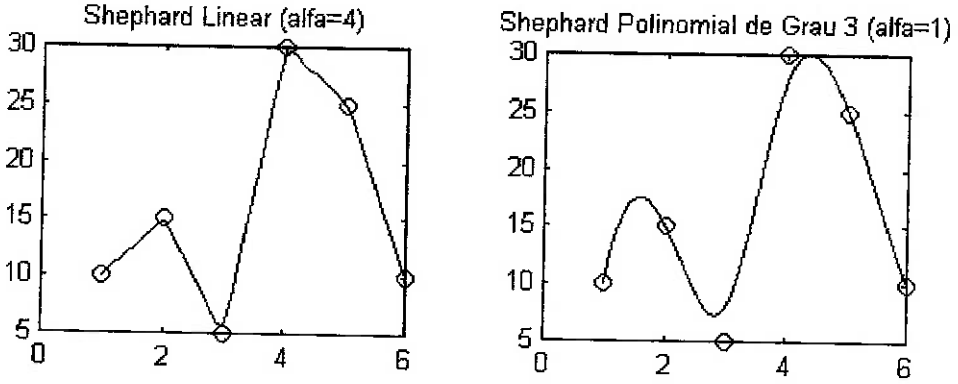


Figura 12 – Shephard linear (alfa = 4) e polinomial de grau 3 (alfa = 1).

A partir dos gráficos observados e considerando o critério definido constata-se que o melhor método a ser utilizado na discretização de sólidos será o Método de Shephard. O tipo de polinômio base do método será o de grau um (linear) por apresentar uma transição entre os pontos sem curvas intermediárias e mais direta. O peso associado ao

método será estudado futuramente na discretização de imagens e sólidos de acordo com o grau de proximidade e precisão desejados.

3.2. Aplicação da função aproximadora num modelo bidimensional

Justificado a escolha do Método de Shephard como base de discretização, a próxima etapa será a implementação destas funções aproximadoras nas direções das linhas e colunas da matriz que representa uma imagem digitalizada. Em primeiro lugar é necessário entender o conceito de imagem digital. Analisando um desenho no formato "bitmap" e aplicando uma aproximação (*zoom*) pode-se perceber que esta imagem é formada por pequenos "quadrados" que correspondem aos pixels. Quanto melhor a resolução deste desenho menor será o tamanho destes pixels e maior será a quantidade destes. Digitalizar este desenho é um processo de associar a cada um destes pixels um número. Para simplificar, podemos observar uma imagem em preto e branco. Todos os pixels com a cor preta serão substituídos pelo número um e aqueles com a cor branca terão o valor zero. A figura (13) representa bem este modelo. Será mostrado posteriormente que a imagem poderá ter cores intermediárias entre o preto e branco, isto é, escalas de cinza variando do mais claro ao mais escuro. Na digitalização estas cores de cinza terão valores intermediários entre zero e um de acordo com a tonalidade de cinza. O processo de digitalização é feito através de um editor de fotos adequado ou por um programa de digitalização de imagens.

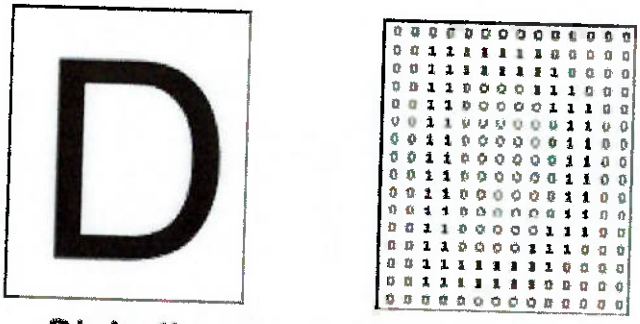


Figura 13 – Processo de digitalização de bitmap para matriz de zeros e uns.

Tendo uma imagem digitalizada o próximo passo é entender como as funções aproximadoras são implementadas. Para isso será analisado um modelo simples de uma imagem de coroa de círculo. O modelo tridimensional desta imagem na figura (14) mostra que, quando o valor do pixel é uma cor preta a função se encontra num nível alto e quando este valor é a cor branca a função está no nível zero.

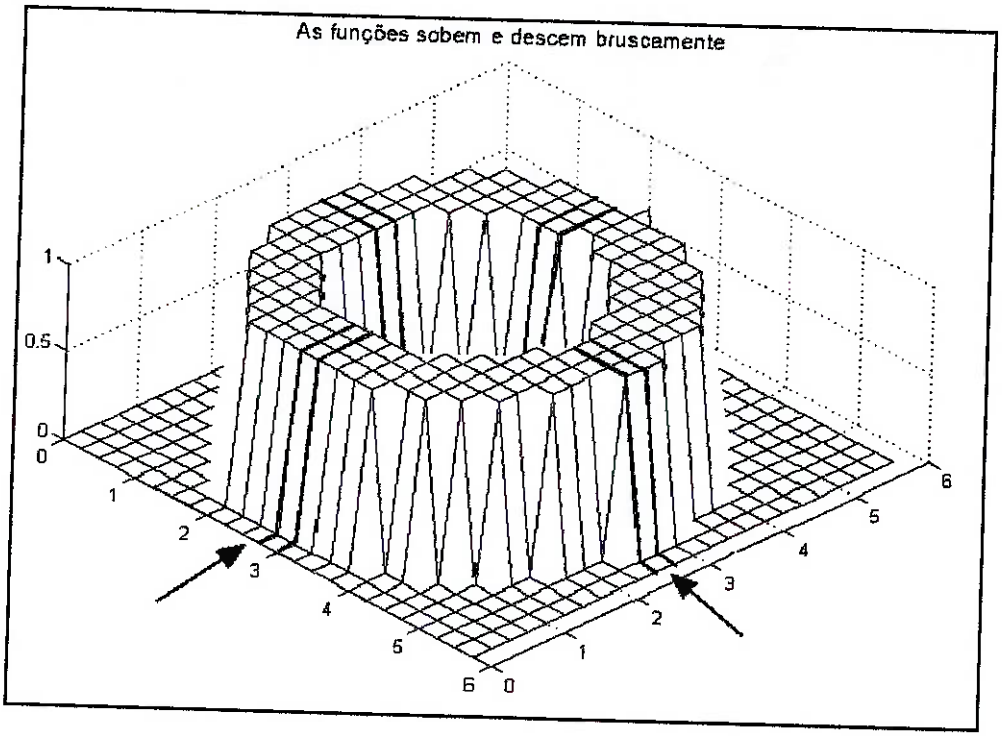


Figura 14 – Representação das funções aproximadoras.

Conclui-se que o método 2D descrito abaixo produz resultados similares ao explicado no apêndice, além de ser mais eficiente computacionalmente.

As funções aproximadoras serão aplicadas para todas as linhas e todas as colunas da imagem digital. Podemos observar, por exemplo a linha 3 da lateral esquerda da imagem tridimensional na figura (14). Quanto mais se avança nesta linha, ora ela apresenta valores altos (um), ora valores baixos. Este fato pode ser melhor observado na figura (15). A partir destes pontos a função aproximadora será implementada com um alto peso (alfa = 5). Esta função aproximadora é representada pela curva preta. A transição da curva do valor zero no ponto 2 para um no ponto 3 ocorre com uma reta inclinada. Esta inclinação é fundamental para o processo de discretização.

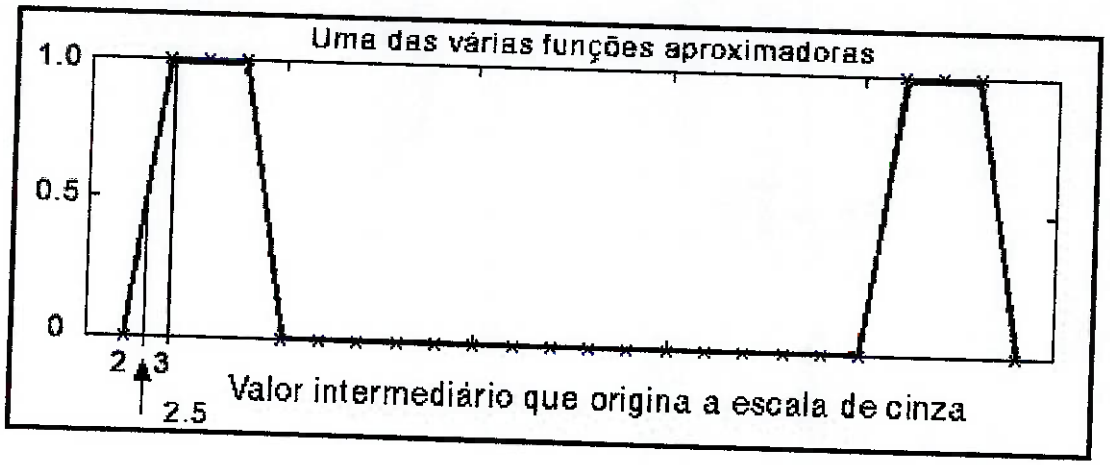


Figura 15 – Função aproximadora.

Um fato importante na discretização é que serão tomados valores intermediários para criarmos um número maior de pixels numa imagem. Assim, na figura (16) o valor entre o ponto 2 e 3 corresponde a 0,5 que não será preto nem branco e sim uma tonalidade de cinza. Este processo será aplicado para todas as linhas e colunas da figura (14) fazendo com que a imagem possua várias tonalidades de cinza nos contornos das cores pretas. Observa-se pela figura (16) que da imagem original com 25 linhas e 25 colunas, após discretizada apresenta uma imagem de escala de cinza com 49 linhas por 49 colunas pois entre uma linha e outra e entre uma coluna e outra criou-se uma nova.

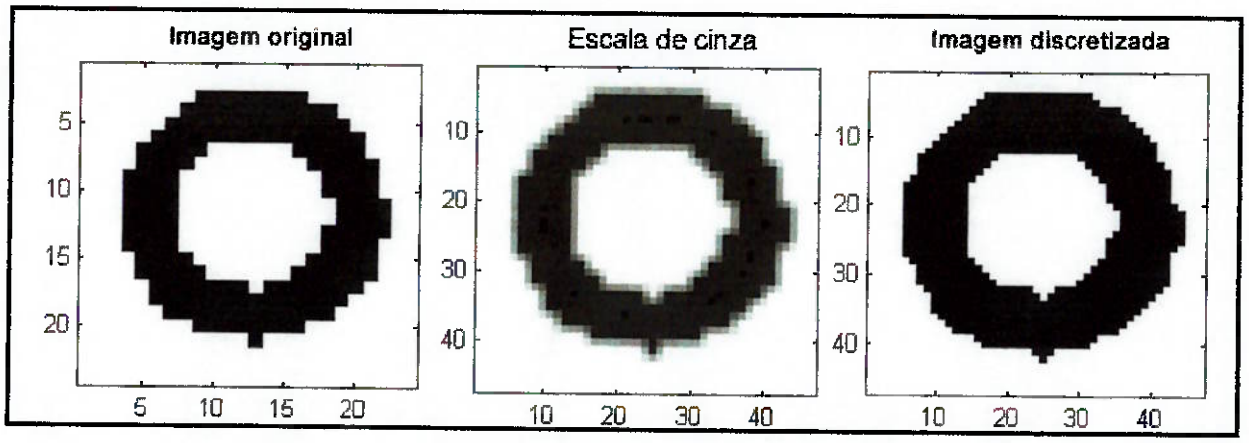


Figura 16 – Imagens do processo de discretização.

O último passo da discretização bidimensional é o uso de uma função de corte ou limiar que verificará o valor de cada pixel determinando se este terá o valor um ou zero. Desta forma os pontos com tonalidade cinza claro (valor baixo) terão um novo valor igual a zero. Os pontos com tonalidade cinza escuro (valor alto) serão substituídos pelo número um. Este processo resultará na imagem discretizada final sem escalas de cinza que é mostrada na figura (16). Dessa forma obteve-se uma representação mais refinada da imagem.

3.2.1. Análise da influência do peso na discretização de imagens

Anteriormente foram estudados valores para o coeficiente α do peso dos pontos numa função unidimensional que corresponderia a apenas uma linha ou coluna da imagem bidimensional. Neste item será analisada a influência deste coeficiente no estudo das imagens bidimensionais. O exemplo apresentado nesta análise consiste em duas colunas de cor preta, uma grande e outra menor. O objetivo será o de criar várias colunas intermediárias que correspondam a uma transição da coluna maior para a menor. Neste primeiro estudo será adotado um peso alfa igual a 1,5. O resultado pode ser observado na figura (17).

Figuras intermediárias com transição não uniforme

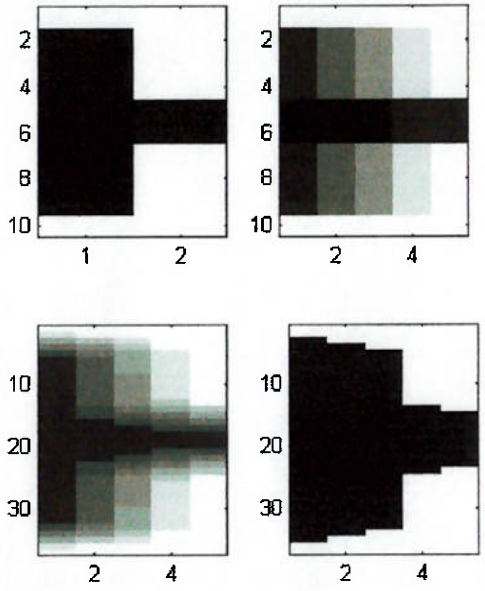


Figura 17 – Discretização de imagem com transição não uniforme.

A primeira imagem mostra duas colunas e pretende-se criar colunas intermediárias. O primeiro passo foi o de realizar a discretização apenas na direção das linhas. Logo as escalas de cinza podem ser observadas apenas na direção horizontal. Em seguida repetiu-se o processo de discretização agora na direção vertical permitindo criar uma escala de cinza também na direção vertical. Finalmente foi utilizado o processo de corte definindo que as tonalidades de cinza fossem transformadas em preto ou branco. Observe que o número de linhas passou de 10 para 35 e o número de colunas foi de 2 (no início) para 5. O uso do peso alfa = 1,5 não correspondeu às expectativas pois, observando a imagem final da figura (17) a transição da primeira coluna (maior) para a última (menor) não foi uniforme com uma descida brusca da terceira para a quarta coluna.

Sendo assim o valor do peso foi modificado para 0,5. Isto faz com que as funções aproximadoras passem mais distantes dos pontos permitindo que a transição de um ponto para outro não seja brusca. Os resultados são apresentados na figura (18).

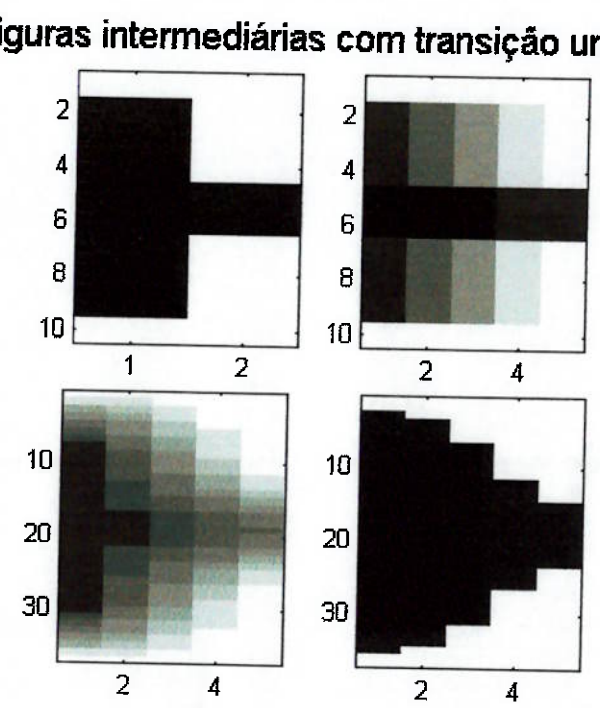


Figura 18 – Discretização com transição uniforme entre as figuras.

Na figura (18) observa-se que a escala de cinza apresenta-se mais dispersa permitindo que no processo de corte a imagem final resultasse numa transição melhor. A criação de um número maior de imagens intermediárias com o peso alfa valendo 0,5 resultará na figura (19).

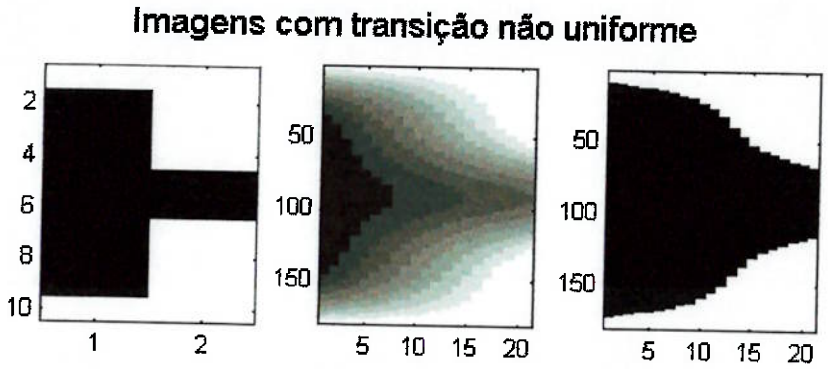


Figura 19 – Transição não uniforme entre as imagens.

Para se criar um número maior de seções intermediárias é necessário reduzir mais o valor do coeficiente alfa para que a dispersão da escala de cinza seja maior e a transição uniforme. Para se ter uma transição uniforme é necessário que a redução do coeficiente alfa seja ainda maior e, neste caso, o valor adotado foi 0,15. Os resultados são

mostrados na figura (20) que, devido a um número de imagens intermediárias maior apresenta resultados de qualidade maior que a mostrada na figura (18).

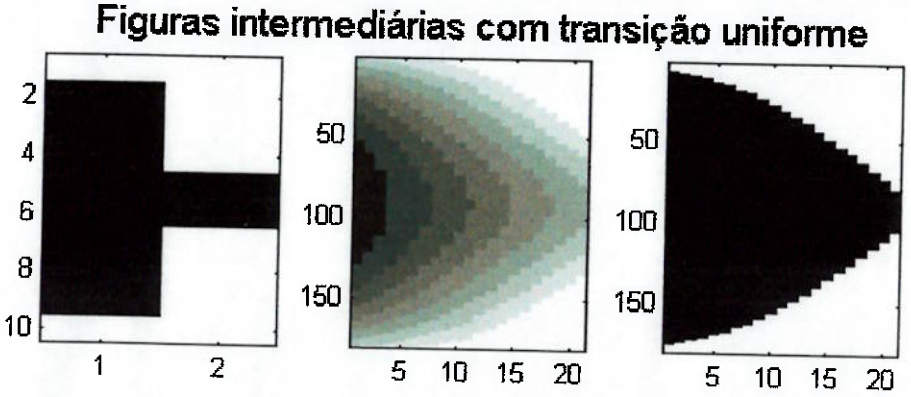


Figura 20 – Qualidade melhor na imagem discretizada uniformemente.

3.3. Uso da função aproximadora num modelo tridimensional

Como o uso do Método de Shephard [4, 9] apresentou bons resultados na discretização de imagens em duas dimensões este processo foi implementado para discretizar sólidos. No modelo unidimensional criavam-se pontos intermediários, no bidimensional procurou-se criar pixels intermediários. Neste ponto a idéia é criar seções intermediárias de uma peça tridimensional. Para verificar a eficácia deste método foram feitas num editor de fotos três imagens simples que podem ser observadas na figura (21).

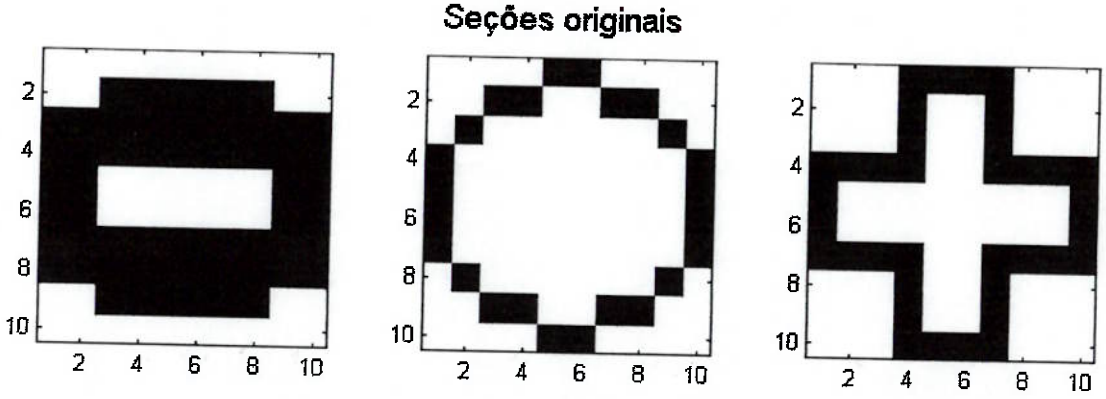


Figura 21 – Seções iniciais

Cada imagem da figura (21) representa uma seção de um sólido qualquer. O programa desenvolvido de discretização, além de executar a discretização de cada seção, criará novas seções intermediárias. Como neste caso há três imagens, logo foram formadas mais 8 intermediárias, isto é, quatro entre as imagens 1 e 2, outras quatro entre as imagens 2 e 3. O resultado final é mostrado na figura (22).

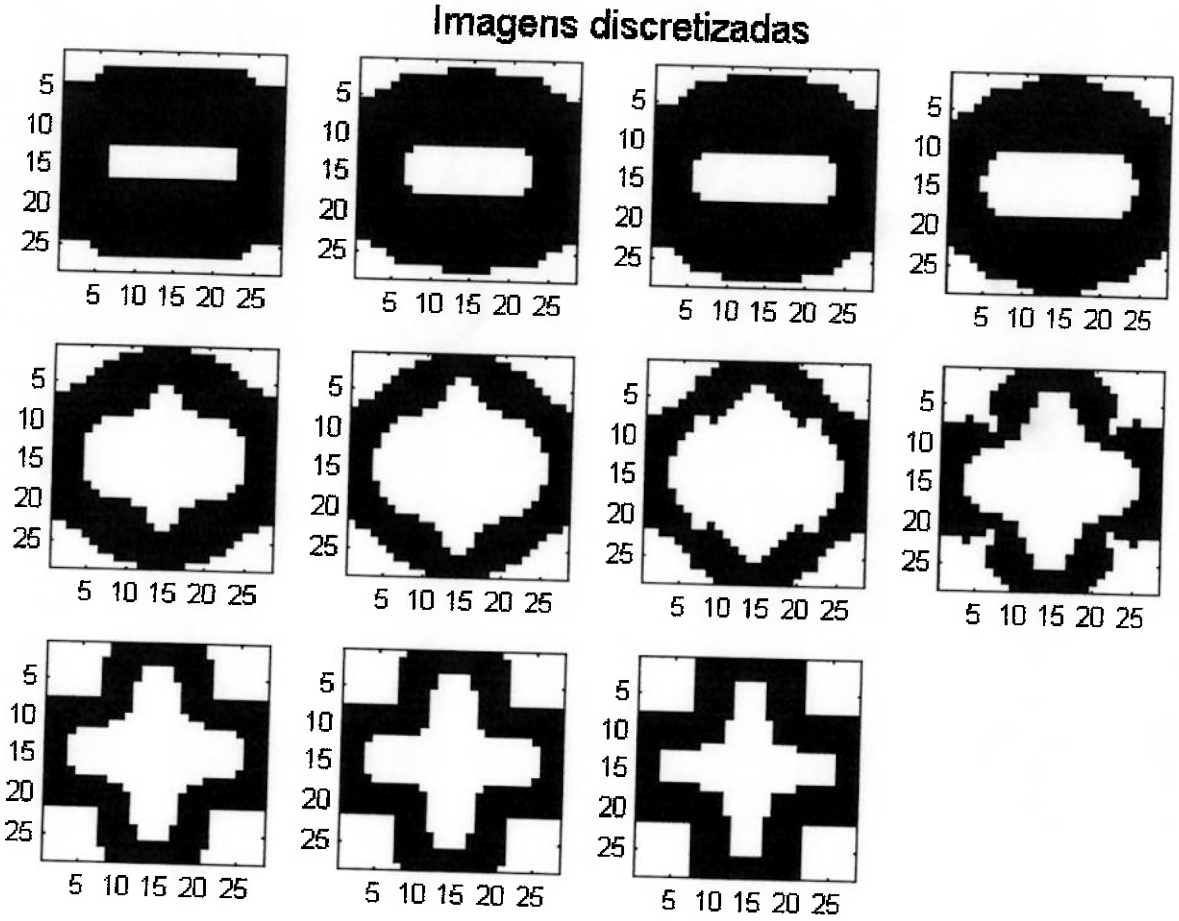


Figura 22 – Obtenção de imagens intermediárias discretizadas.

Devido a discretização cada imagem resultante passou a ser uma matriz de 29 linhas por 29 colunas. Outro fato que pode ser constatado é que as imagens resultantes representam a transição de uma para outra.

Com isto verifica-se que o método de discretização para seções de um sólido realmente cria imagens intermediárias. Sendo assim, o estudo avançará para a visualização dos resultados em três dimensões.

3.4. Conversão das matrizes para um programa de visualização

Na etapa anterior o programa de discretização foi aplicado com o objetivo de discretizar as seções existentes e criar novas. Todas essas seções estarão no formato de uma matriz tridimensional de zeros e uns onde cada nível corresponde a uma seção do sólido. O próximo passo consiste em visualizar esta matriz tridimensional. Isto é feito através de um programa que "desenhe" esta matriz. As posições identificadas pelo valor um serão visualizadas como um pequeno cubo ou um paralelepípedo (voxel) e naquelas que tiverem o valor zero nada será desenhado. Quando se discretiza uma imagem geralmente se trabalha com pixels. Na discretização de sólidos os elementos que constituem este sólido (paralelepípedos) são chamados de voxels.

3.5. Discretização de peças tridimensionais

A primeira peça complexa modelo foi uma representação da caixa de diferencial de carro. Pelo fato de no início do trabalho não se ter acesso a um tomógrafo esta peça foi modelada em um programa de CAD. Ela apresenta detalhes como orifícios laterais para a saída dos eixos das rodas e anéis circulares de pequena espessura ao longo da peça.

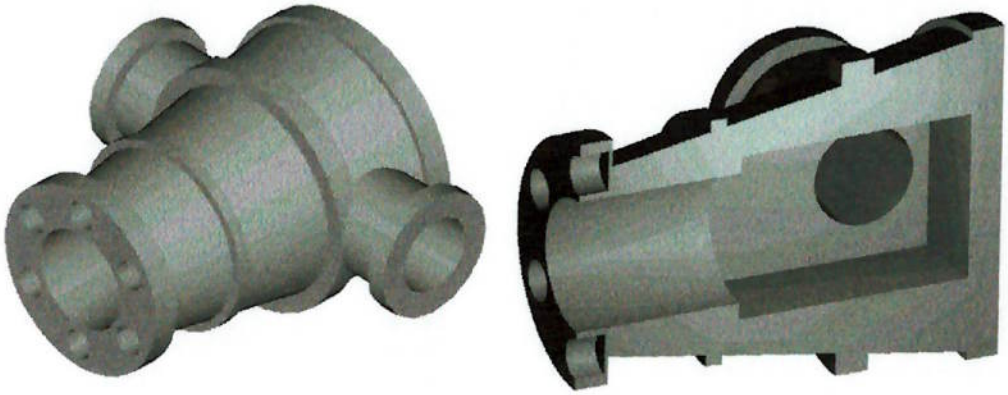


Figura 23 – Diferencial.

O passo seguinte foi a obtenção das seções transversais fazendo uso do próprio programa de CAD. Foram obtidas 44 seções transversais com resolução 120 pixels de largura por 80 pixels de altura e estas são mostradas na figura (24). Deve-se lembrar que, com o uso da tomografia computadorizada, as seções transversais são obtidas automaticamente da peça real, sem a necessidade de se modelar esta no CAD.

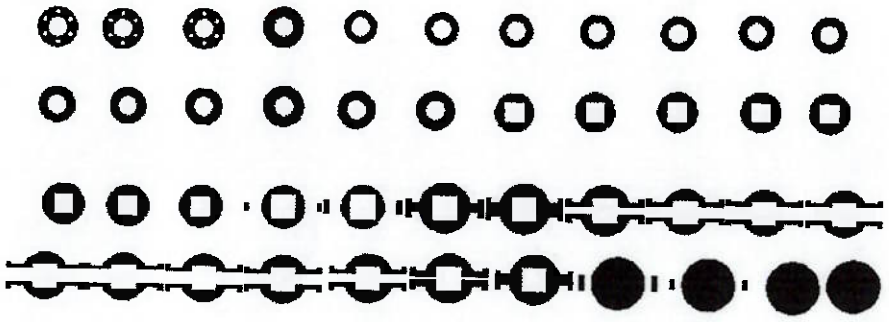


Figura 24 – Seções da caixa de diferencial.

Cada uma destas seções foi digitalizada (transformada em matriz de zeros e uns) e executou-se então o programa de discretização. Como resultado foram obtidas 43 novas seções transversais totalizando 87 e o resultado mostrado na figura (25) com o programa VTK.

Figura 25 – Diferencial discretizado.

Apesar de se ter tomado um número baixo de seções transversais a qualidade da discretização se apresentou boa. Pode-se verificar que os anéis ao longo do corpo da peça foram preservados não se perdendo informações. Também foram mantidas as características das circunferências frontais (mesmo as pequenas) e laterais.

Para finalizar os exemplos aplicados foi modelado em AutoCAD um motor elétrico apresentando um nível de detalhamento maior quanto as curvaturas. Isto fica bem evidenciado pelas aletas presentes e pelo corpo do motor.

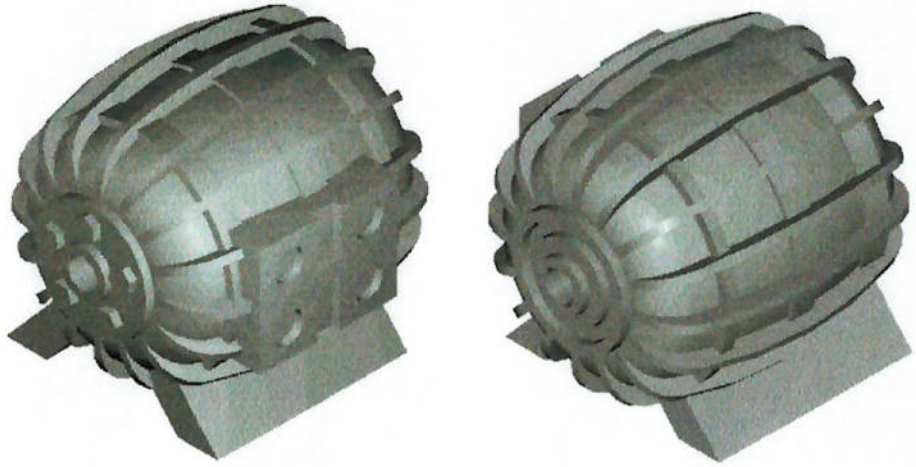


Figura 26 – Motor elétrico.

Para a discretização foi utilizado um número ainda maior de seções transversais totalizando 72.

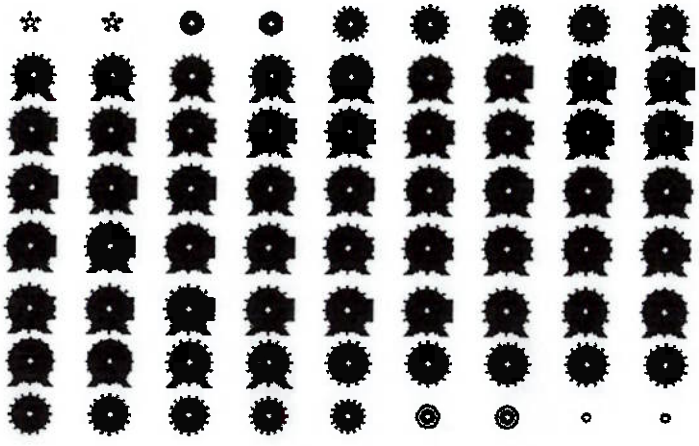


Figura 27 – Seções do motor.

Na discretização todas as características principais foram mantidas incluindo os parafusos frontais, curvatura das aletas e os detalhes da caixa de comandos na lateral.

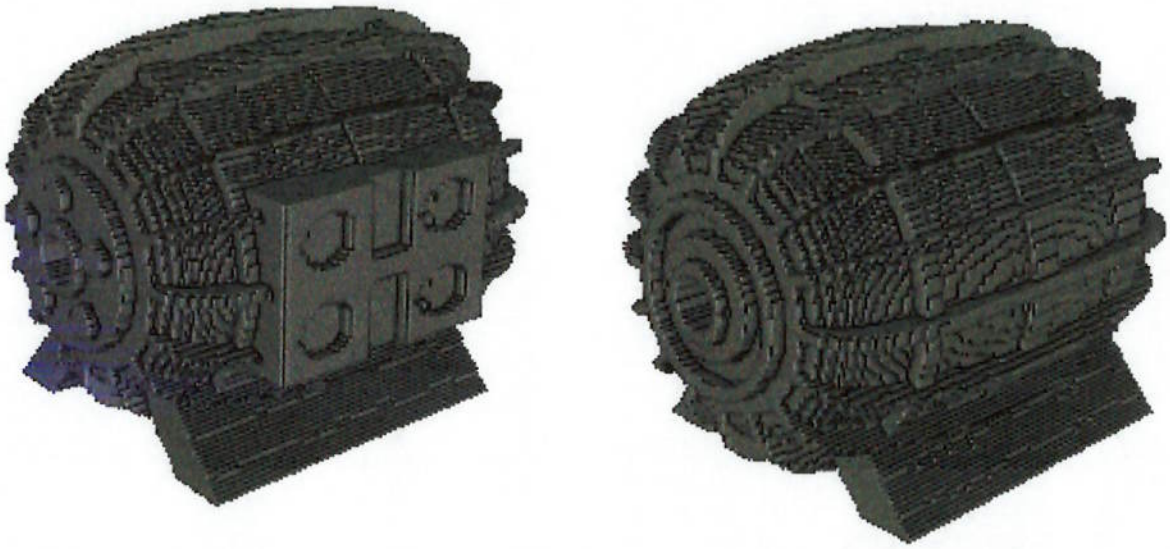


Figura 28 – Motor discretizado.

Para encerrar o estudo foram realizadas alterações no diferencial mostrado na figura (23). Para isto algumas seções transversais em formato bitmap foram modificadas com o auxílio de um programa de edição de imagens do tipo "Paintshop". Foram alteradas as seções indicadas figura (29).

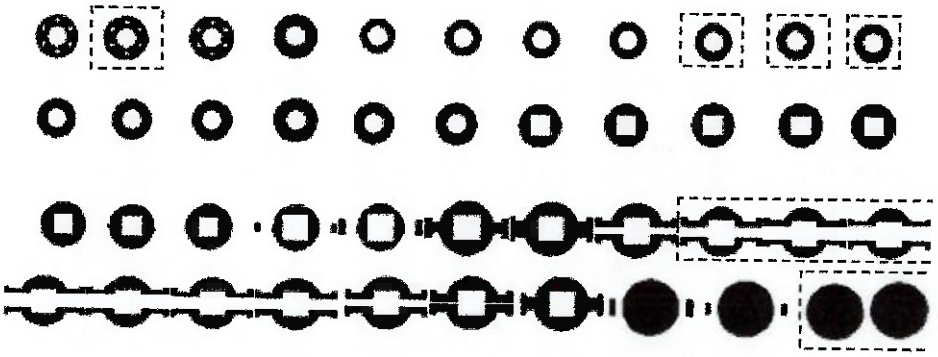


Figura 29 – Seções modificadas.

Com as modificações o passo seguinte foi semelhante ao das peças anteriores obtendo-se seções intermediárias. O resultado é apresentado na figura (30).

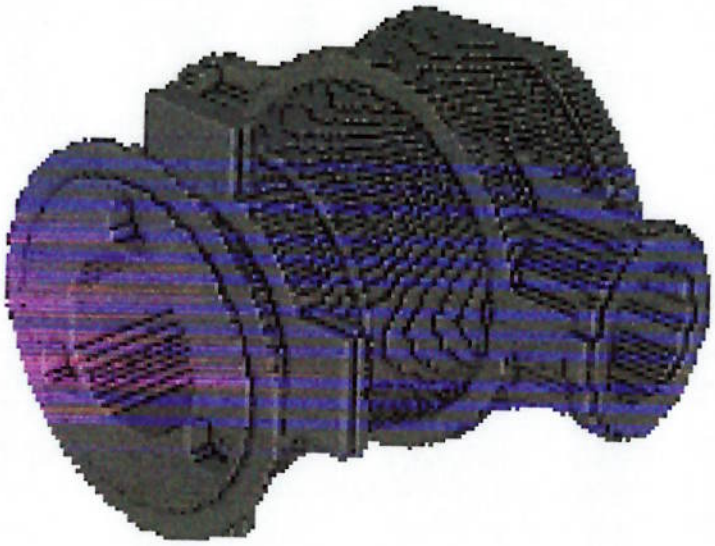


Figura 30 – Diferencial modificado.

A modificação da peça foi feita facilmente pelo programa de edição de imagens. Um conjunto de seções correspondente ao local onde se deseja modificar o formato foram alteradas bastando para isto apagar ou pintar pixels gradativamente de seção para seção.

3.6. Problemas enfrentados

A primeira dificuldade correspondeu a leitura dos arquivos. Esta leitura foi feita com sucesso no ambiente DOS/WINDOWS entretanto apresentou problemas em LINUX. Isto foi resolvido através de pesquisa mais detalhada sobre o uso de ponteiros na leitura de caracteres. Inicialmente foi utilizado o comando *fscanf(arq,"%s",b)*. O formato correto adotado após pesquisa foi *fscanf(arq,"%s",&b)* com o símbolo &.

Outro desafio que surgiu com o uso de peças mais complexas foi relativo a limitação de memória alocada pela linguagem. Vetores em linguagem C são representados como: *vetor[posição]*. Matriz bidimensional possui a seguinte

representação: *matriz[linhas][colunas].* Matriz tridimensional é escrita como: *matriz[linhas][colunas][níveis].* Entretanto, para peças complexas a matriz deve possuir cerca de 100 linhas por 100 colunas por 100 níveis. Considerando que cada elemento da matriz ocupa 4 bytes de memória é necessário possuir cerca de 3,9 Megabytes por matriz. Este tamanho não é suportado no formato *matriz[linhas][colunas][níveis].* Logo matrizes com muitas linhas, colunas e níveis não eram suportadas. Isto foi resolvido com um método denominado alocação dinâmica de memória, em que se reserva uma quantidade de memória para uma determinada variável. Assim, para se alocar memória proporcional às dimensões da matriz utilizou-se o comando: *matriz1=(int*)malloc((tlf+1)*(tcf+1)*(tsf+1)*sizeof(unsigned int)).* Quanto maior o número de linhas, colunas e níveis, maior a alocação.

Para se identificar um elemento em uma matriz basta indicar linha, coluna e nível. Como se trabalhou com vetores foram feitas contas para se identificar posição de valores no vetor que representa uma matriz. Sendo assim, por exemplo, na figura 31 a matriz possui 5 linhas por 4 colunas em linguagem C o elemento *g* da linha 2 e coluna 3 seria simplesmente representado por: *matriz[2][3].*

a b c d
e f g h
i j k l
m n o p
q r s t
Matriz 5 linhas e 4 colunas
Posição (2,3): g

Figura 31 – Representação de matriz.

Um vetor não possui colunas nem níveis, é como se fosse uma única linha e é mostrado na figura 32.

a b c d e f g h i j k l m n o p q r s t

Vetor de 20 posições

Posição(2x2+3=7): g

Figura 32 – Matriz transformada em vetor.

Logo a matriz deve ser transformada em um vetor de 20 elementos (5 vezes 4), quando uma linha termina (na letra *d*) outra deve começar em seguida. Para identificar a letra *g* deve-se multiplicar o total de colunas *4* pela primeira posição (posição da linha) *2* e somar com a segunda posição (coluna) *3* (2x2+3=7). O mesmo procedimento foi generalizado para matriz tridimensional.

Outro obstáculo foi descobrir problemas quanto a operações com números inteiros. Por exemplo, na simbologia c=a/b, se a=3 e for inteiro e b=2 e também for inteiro o resultado da conta será c=1 mesmo que c seja um número real. Para que a conta seja c=1,5 a e b precisam ser números do tipo "float".

4. INTERPRETAÇÃO DE IMAGENS GERADAS PELO MÉTODO DE OTIMIZAÇÃO TOPOLÓGICA

Otimização Topológica é um método de projeto computacional que combina algoritmos de otimização e método de elementos finitos para encontrar a topologia ótima de peças mecânicas, considerando uma função objetivo desejada e algumas restrições. Essencialmente, ela nos permite projetar estruturas com orifícios posicionados de forma ótima para maximizar (ou minimizar) uma função de custo definida para a estrutura. Este é um método mais genérico que os métodos de otimização paramétricos e de forma, onde apenas algumas dimensões ou as formas da estrutura são otimizadas, respectivamente.

A principal vantagem da otimização topológica é que ela nos permite encontrar novos "buracos" na estrutura e, conseqüentemente, a redução de peso destas peças é bem maior que a redução obtida por outros métodos. Otimização topológica tem sido vastamente utilizada nas indústrias automotivas e aeronáuticas para projetar estruturas e elementos mecânicos com alta rigidez e baixo peso. Os resultados gerados pelo M.O.T. apresentam em geral uma escala de cinza como mostrado na figura (33). Essa imagem precisa ser interpretada, o que pode ser feito utilizando-se o método implementado nesse trabalho.

No item anterior foi utilizado um processo que permitia a melhora da resolução de cada seção transversal e, com a presença das escalas de cinza era possível criar novas seções transversais.

A interpretação de imagens geradas por otimização topológica é uma particularização do que foi feito anteriormente. Ela consiste em, a partir de imagens de baixa resolução com escala de cinza, aumentar a qualidade destas criando-se novos pixels em linhas e colunas. Além disto é necessário definir um "threshold" para a escala de cinza.

No exemplo mostrado na figura (33) tem-se a imagem inicial fornecida pela otimização topológica com uma resolução de 20 linhas por 45 colunas. Note que a imagem apresenta escala de cinza.

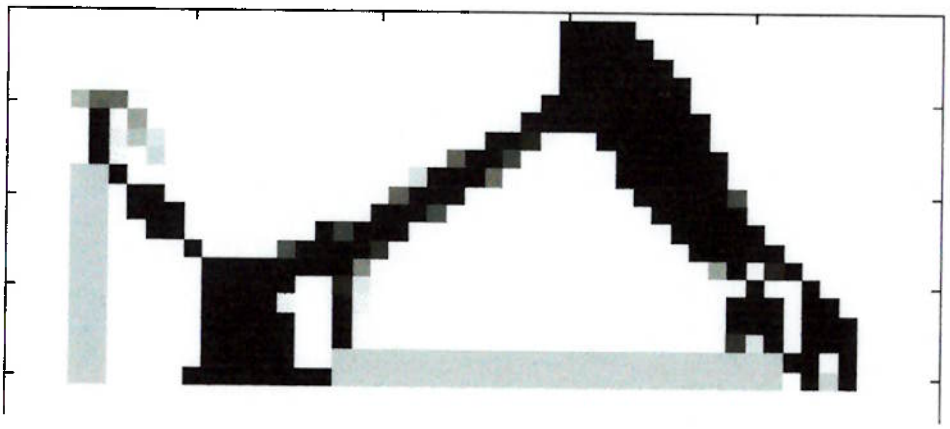


Figura 33 – Imagem inicial.

Aplicando-se o programa modificado para discretizar imagens 2-D utilizou-se o parâmetro alfa igual a 1,0 e o threshold igual a 30. Observa-se que o número de colunas passou a ser 89 e o número de linhas igual a 39 e o resultado é mostrado na figura (34).

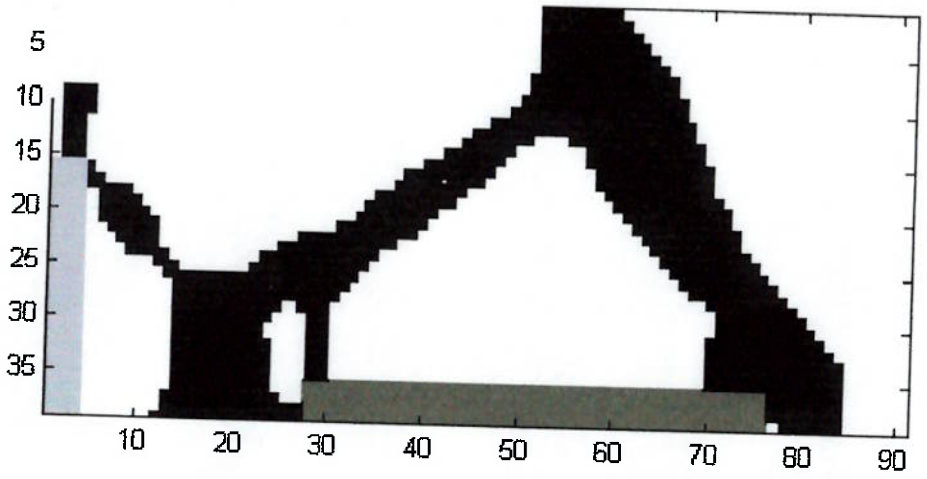


Figura 34 – Imagem discretizada.

A idéia básica para a primeira discretização foi a de manter a espessura das regiões escuras constante. Entretanto, variando-se os parâmetros a imagem discretizada pode ter sua espessura incrementada deixando um aspecto mais espesso como mostrado na figura (35). Pode-se notar entretanto que algumas regiões, onde havia pequenos espaços em brancos passaram a não existir.

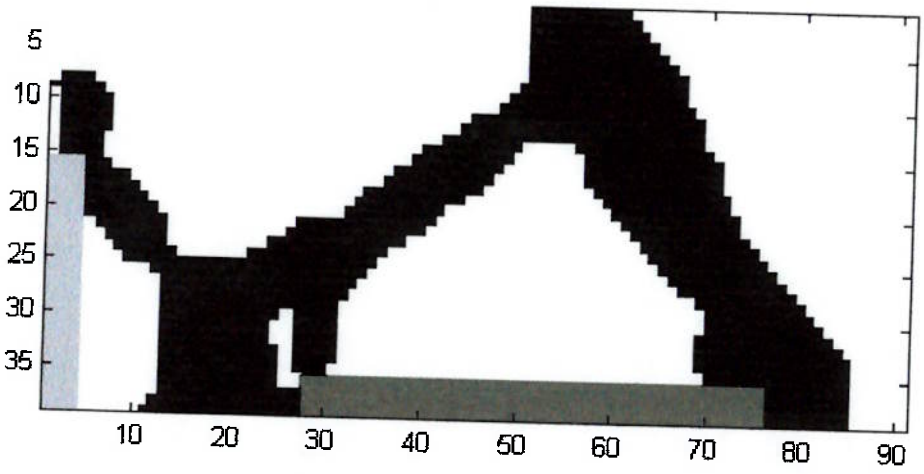


Figura 35 – Discretização grossa.

Outra alternativa foi a modificação dos parâmetros para se ter uma discretização fina. Neste caso regiões mais finas da imagem ficaram mais finas ou mesmo desapareceram. Pode-se verificar este fato na figura (36).

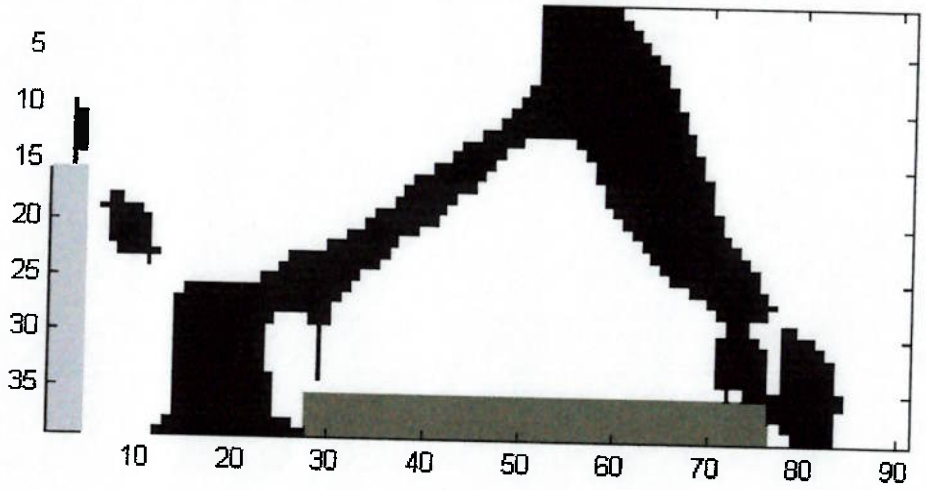


Figura 36 – Discretização fina.

Conclui-se desta forma que a discretização pode ser controlada de acordo com as necessidades do usuário em manter detalhes ou de destacar características como a espessura.

5. TOMOGRAFIA DE PEÇAS

Os resultados desta pesquisa foram apresentados para pesquisadores do Instituto Dante Pazzanese e Hospital do Coração da área biomédica e de tomografia. Através de uma reunião com esses profissionais, decidiu-se tomografar algumas peças de origem biomédica e mecânica para este trabalho. Estas tomografias foram realizadas no Hospital do Coração. Neste item serão apresentados os resultados das tomografias e os resultados da aplicação do método de discretização.

O trabalho despertou grande interesse por parte dos pesquisadores que trabalham com próteses biomédicas para o projeto de peças mais resistentes e eficientes. Os

pesquisadores tomografaram estas peças e acompanharam o processo do tratamento das imagens. As figuras abaixo mostram o tomógrafo utilizado e uma peça tomografada.

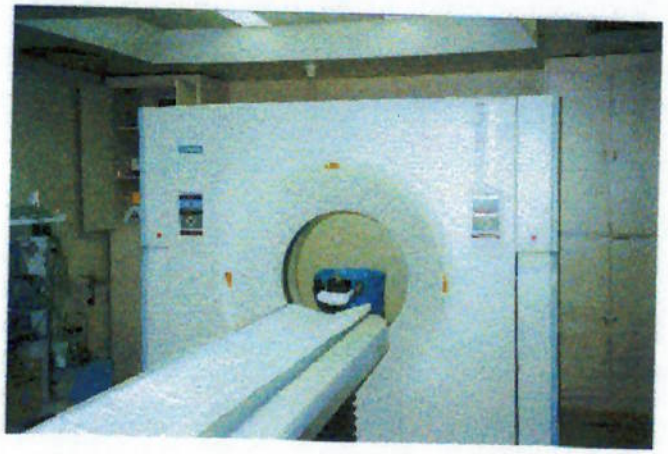


Figura 37 – Tomógrafo utilizado.

Figura 38 – Peça tomografada.

5.1. Características das peças

Para este projeto foram tomografadas duas peças biomédicas e algumas peças mecânicas. Estes modelos foram escolhidos devido a complexidade de fabricação e de modelagem em CAD. Elas apresentam diversas superfícies curvas, formas assimétricas e paredes de pequena espessura. Estes problemas exigiram um nível de discretização maior que anteriormente adotado e um dos estudos realizados foi compatibilizar o número de elementos com a capacidade de processamento dos computadores e programas de Elementos Finitos disponíveis.

Uma das peças biomédicas é uma bomba centrífuga de sangue cuja complexidade está na sua forma de cone com laterais espiraladas. Esta bomba feita de acrílico serve para o bombeamento de sangue em cirurgias (circulação extracorpórea).

Figura 39 - Bomba centrífuga.

A outra peça de aplicação médica é a cavidade de um coração artificial feita de resina. Sua forma é tal que apresenta dois orifícios, um para a entrada e outro para a saída de sangue e paredes de espessura variável ao longo da peça.

Figura 40 – Cavidade de um coração artificial.

5.2. Obtenção dos modelos discretizados

Inicialmente foram obtidas 52 imagens das seções transversais da cavidade tomografada. Cada imagem possui uma resolução inicial de 187 linhas de pixels por 365 colunas de pixels. Considerando o número de imagens esta resolução é bastante alta para se trabalhar com um programa de elementos finitos e os computadores disponíveis. Logo a resolução de cada imagem foi reduzida para 43 linhas de pixels por 84 colunas de pixels e as imagens são mostradas na figura abaixo.

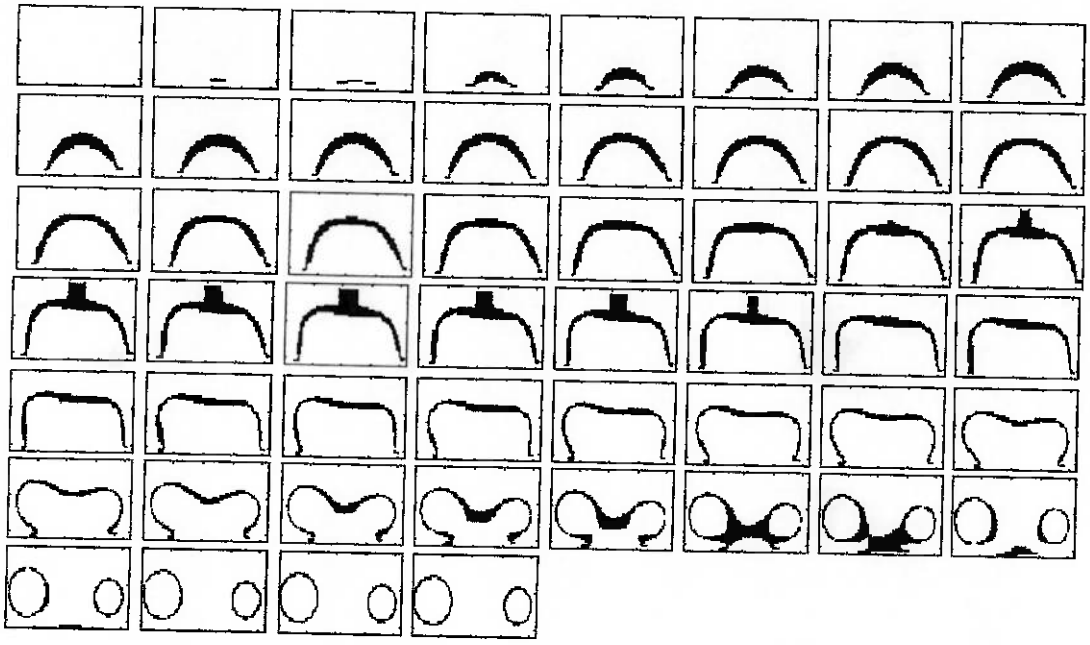


Figura 41 – Seções transversais da cavidade.

Em seguida foi aplicado o programa de discretização para o aumento da resolução da peça e criação de seções intermediárias. O resultado foi a obtenção da imagem tridimensional da cavidade com resolução de 43 linhas de pixels por 84 colunas de pixels com 101 imagens conforme mostrado a seguir.

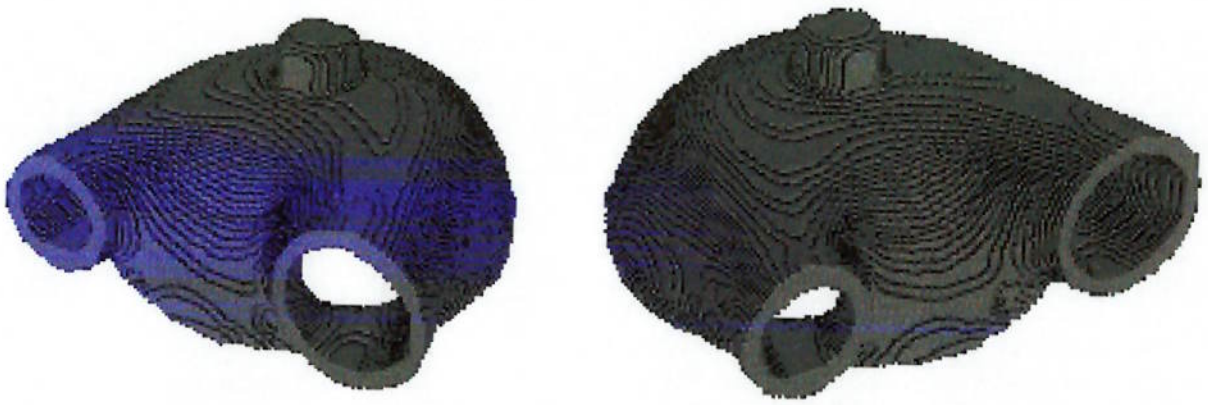


Figura 42 – Cavidade discretizada original e discretizada.

Como discutido anteriormente o método de discretização também oferece vantagens no momento em que se deseja modificar o formato da peça em questão. Para isto, algumas seções transversais da cavidade foram modificadas, em seguida utilizou-se novamente o programa de discretização e os resultados podem ser vistos na figura a seguir.

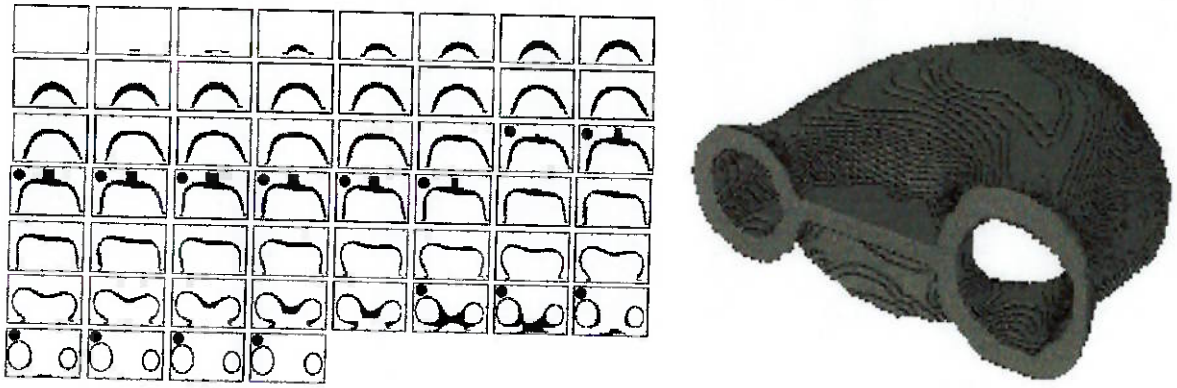


Figura 43 – Cavidade modificada.

Os procedimentos de discretização também foram aplicados à bomba centrífuga em espiral. Neste caso, devido a inclinações bruscas da peça devido ao seu formato de cone, foi necessário um número maior de seções transversais igual à 70. A resolução de cada imagem foi adaptada para 53 por 70 pixels.

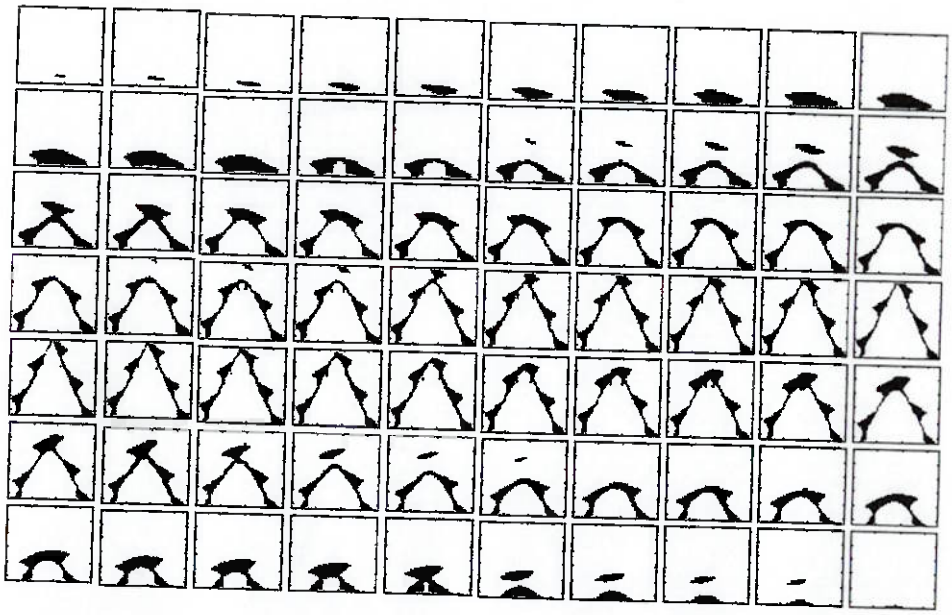


Figura 44 – Seções transversais da bomba em espiral.

As seções foram discretizadas e o resultado é apresentado a seguir. A obtenção destes modelos exigiu um estudo cuidadoso dos parâmetros de discretização de modo que o sólido não apresentasse defeitos como "buracos" devido a menor resolução das imagens.

Figura 45 – Modelo discretizado da bomba em espiral.

5.3. Conversão das peças em modelos de elementos finitos

Os modelos mostrados anteriormente representam matrizes tridimensionais compostas de voxels (pequenos "tijolos"), que não podem ser simulados em um CAE.

5.3.1. Implementação em programa de elementos finitos

As unidades básicas que formam uma imagem bidimensional são chamadas de pixels (pequenos quadrados). Na imagem tridimensional de um sólido, as unidades básicas que constituem este sólido são chamadas de voxels (pequenos paralelepípedos).

Para a implementação do modelo do sólido discretizado em um programa de CAE é preciso converter cada voxel em um elemento finito. Desta forma, o que era apenas um pequeno paralelepípedo (voxel) precisa ser transformado em um elemento que possua conectividade e nós. Cada elemento finito possui oito vértices que são denominados nós. O formato de leitura dos arquivos de entrada em um software de MEF exige que sejam especificadas as coordenadas dos nós de cada elemento finito bem como sua conectividade. A conectividade de um elemento finito representa quais nós pertencem a cada elemento (tijolo) desenhado. O modelo de transformação de pixels 2D em voxels e para elementos finitos é mostrado na figura (46).

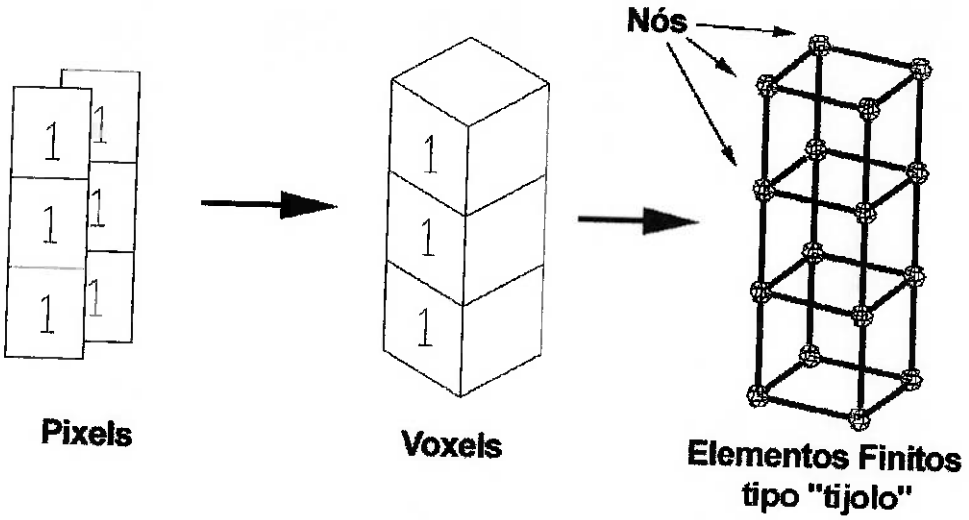


Figura 46 – Obtenção dos Elementos Finitos.

No arquivo denominado "Voxel.txt" (em anexo) é dado um exemplo de arquivo de texto gerado para ser lido pelo ANSYS. Ele possui um cabeçalho e, em seguida observa-

se a presença de várias linhas que começam com a letra "N" seguida por quatro números. O primeiro representa o número do nó. Os três números em seguida indicam suas coordenadas x, y e z respectivamente. Após a apresentação de todos os nós, o arquivo possui novos dados referentes as propriedades do sólido que está sendo carregado pelo programa de elementos finitos. Finalmente aparecem várias linhas com as letras "EN" e nove números seguidos. Neste ponto o ANSYS fará a leitura da conectividade de cada voxel desenhado no espaço. O primeiro número após "EN" representa o número do cubo a ser desenhado. Em seguida aparecem os números de todos os nós que constituem o cubo. Para gerar o arquivo de entrada do ANSYS mediante a matriz de zeros e uns do sólido, desenvolveu-se no mesmo programa "Digicon.c" a seqüência de operações necessárias. A listagem parcial deste arquivo está presente no Relatório Final de Iniciação Científica.

Um exemplo simples foi aplicado na figura (47). Entre camadas de pixel obtém-se voxels. Três imagens foram digitalizadas e o programa de conversão colocou estas numa matriz tridimensional e, através de manipulações matemáticas, foram gerados os nós e as conectividades. Este processo permitiu a transformação de voxels em elementos finitos. A figura (47) ilustra o conceito para um sólido simples.

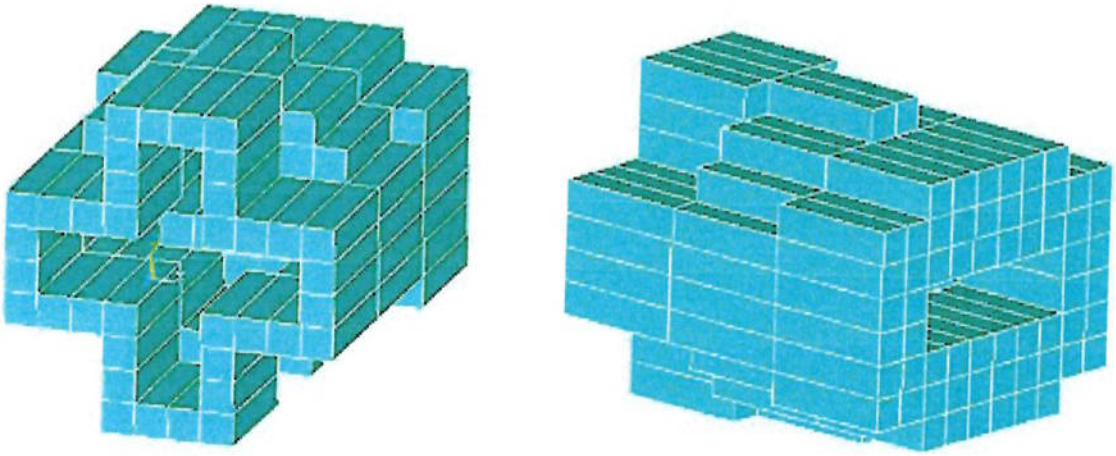


Figura 47 – Visualização do sólido.

cubo que compõe o sólido representa um voxel. Utilizando-se o programa de conversão cada voxel é transformado em um elemento finito com seus respectivos nós e conectividades. Finalmente este modelo está pronto para ser simulado no software de elementos finitos.

5.3.2. Resultados da conversão

O método discutido no item anterior foi implementado com sucesso no próprio programa de discretização para cada peça. Abaixo está o resultado para a cavidade.

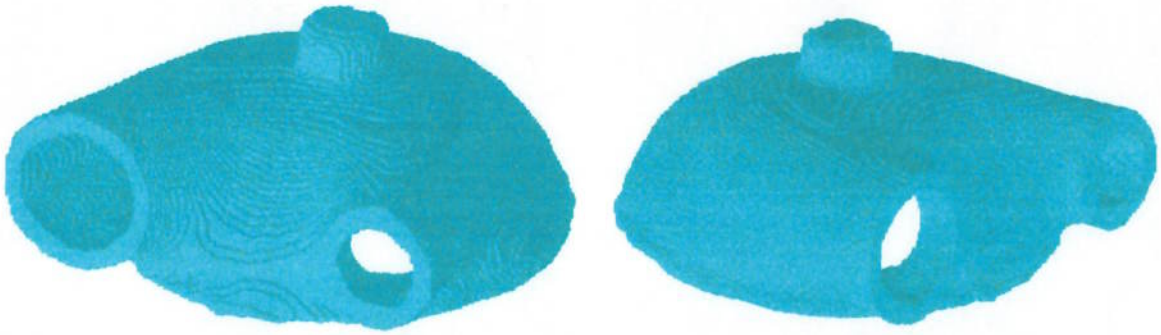


Figura 49 – Modelo CAE da cavidade.

Da mesma forma a bomba espiral também foi convertida para um modelo CAE. Este modelo, devido ao maior número de elementos, exigiu mais tempo de processamento e para ser carregado no programa de CAE. O modelo é mostrado na figura abaixo.

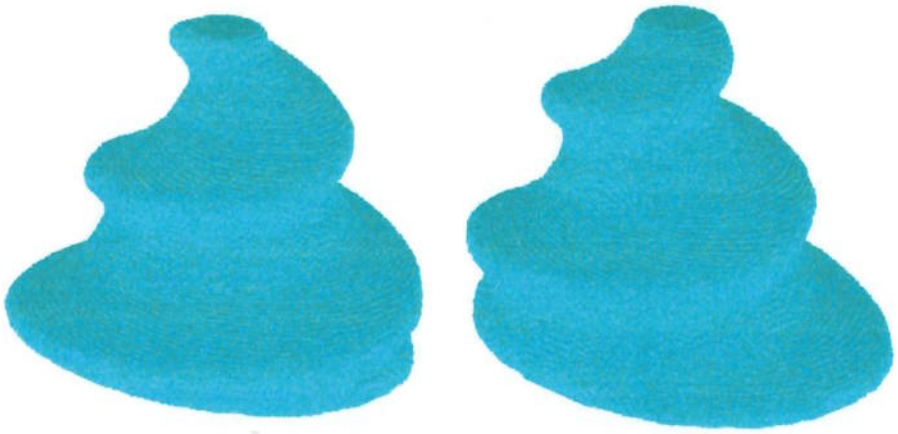
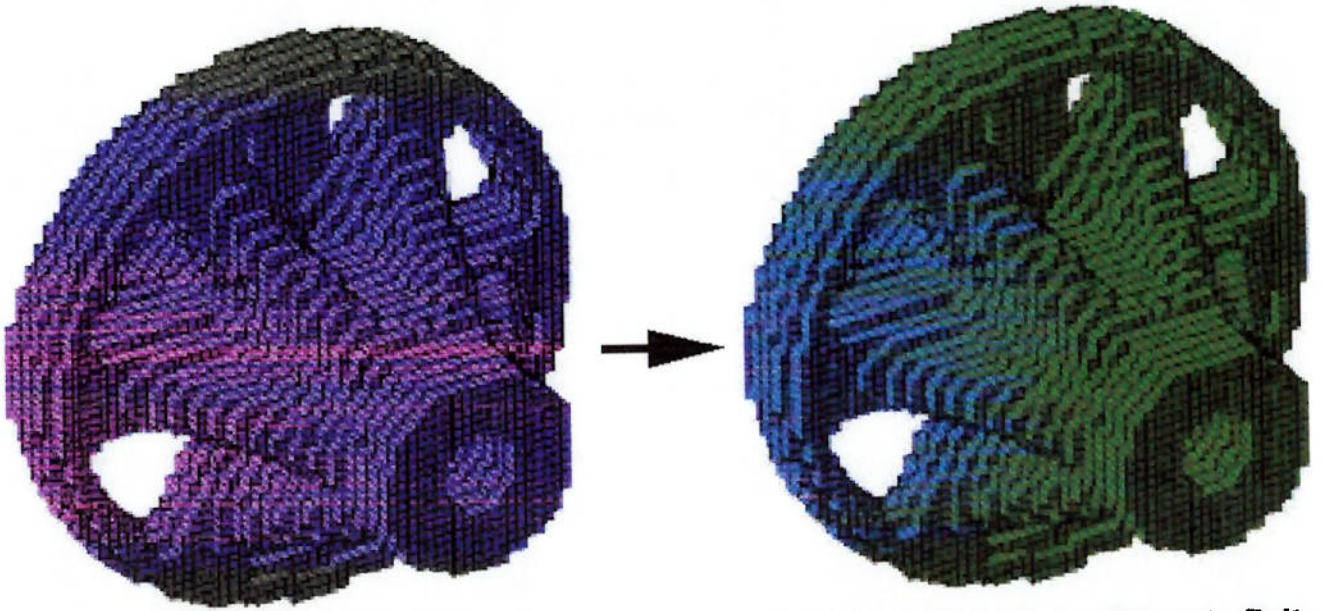


Figura 50 – Modelo CAE da bomba espiral.

Na manipulação matemática para a conversão, o primeiro passo é encontrar quais são os voxels em que há material. Isto é feito através de uma busca em cada posição da matriz do sólido discretizado, variando-se o número da linha, coluna e nível. Em seguida, sabendo a posição do voxel a ser transformado em elemento finito, o programa desenvolvido determina o número e a seqüência dos oito nós que fazem a conectividade do elemento. No passo posterior o programa encontra as coordenadas dos nós em questão. Todos estes dados são, em seguida, ordenados e armazenados em vetores. Finalmente, o último passo consiste na geração do arquivo de saída, onde são inseridos dados essenciais para o arquivo ser reconhecido pelo programa de MEF. No mesmo arquivo são impressos os nós e os elementos com suas respectivas conectividades.

Com o funcionamento deste programa de conversão para o sólido simples mostrado na figura (47), o próximo passo foi generalizar para um sólido de maior complexidade. Isto foi feito e o resultado é mostrado na figura (48).



a) Cada "cubo" é um voxel **b) Cada "cubo" é um elemento finito**

Figura 48 – Modelo de Elementos Finitos gerado pelo ANSYS.

A peça mostrada acima representa a conversão de um modelo de voxels em um modelo de Elementos Finitos, visualizado pelo programa ANSYS. Na imagem inicial cada

5.4. Reconhecimento do programa de elementos finitos (CAE)

Após carregar com sucesso os arquivos da cavidade e da espiral no programa de CAE o passo seguinte foi estudar o software de elementos finitos para executar a simulação propriamente dita. A princípio o programa escolhido havia sido o ADINA. Foi estudada a forma de importar elementos neste programa e foi feito no programa de discretização um conversor para formato NASTRAN que pode ser aberto no ADINA. Algumas peças foram carregadas e foi realizado o estudo de simulações de modelos simples.

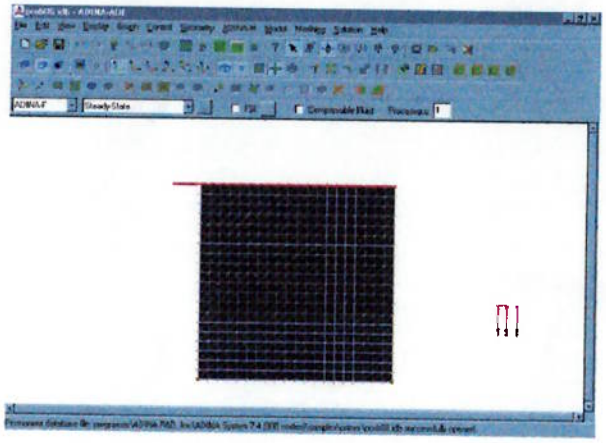


Figura 51 - ADINA.

Como a versão disponível do ADINA era limitada a segunda opção foi o ANSYS. Inicialmente carregar e simular peças complexas neste programa era um procedimento lento mas, com o uso de computadores mais velozes o ANSYS passou a ser viável. Como o programa de conversão criava um modelo arquivo para este programa logo foi realizado apenas o estudo de como realizar as simulações.

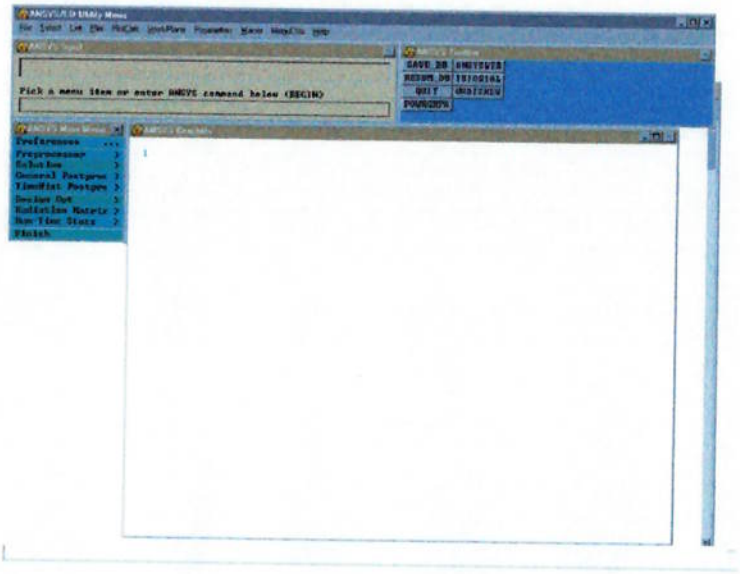


Figura 52 – ANSYS.

Inicialmente estudou-se como aplicar as restrições ao longo da peça, sabendo que esta possui apenas elementos e nós e não superfícies como em modelos criados no próprio programa. Em seguida foram aplicados esforços como forças e pressão ao longo das peças e escolheu-se as propriedades das peças tais como módulo de elasticidade, densidade e coeficiente de poisson. Finalmente foram realizadas as simulações e a análise dos resultados como deformações e distribuição de tensões.

5.5. Resultados da simulação no CAE

A primeira peça simulada foi a cavidade do coração. Para viabilizar a simulação em menor tempo sem prejuízos aos resultados, os parâmetros do programa de discretização foram escolhidos de modo que o modelo da cavidade apresentasse cerca de 20 mil elementos.

Foram aplicadas restrições ao longo da base desta e posteriormente acrescentou-se uma distribuição de pressão uniforme ao longo das regiões por onde o sangue flui.

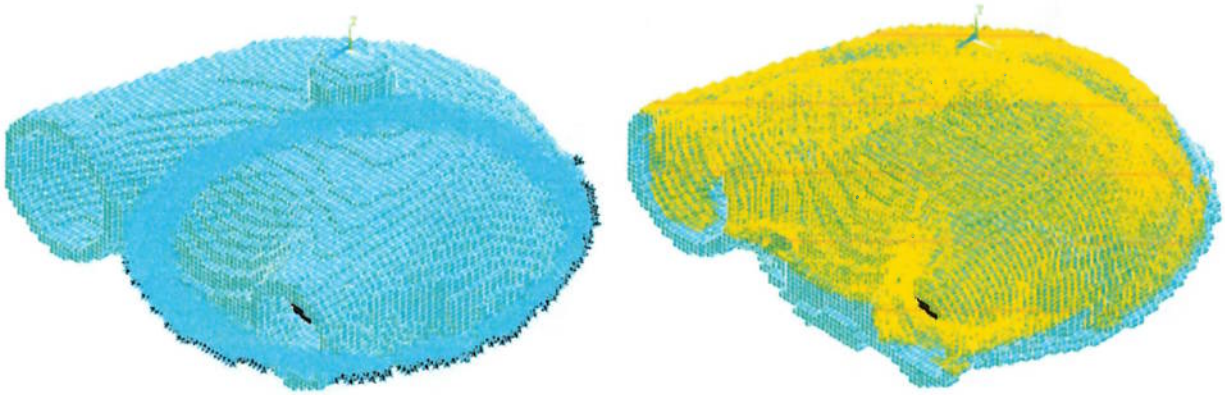


Figura 53 – Cavidade submetida às restrições e pressão.

Feitas as simulações necessárias foram plotados os resultados das distribuições de tensões de Von Mises, mostradas abaixo.

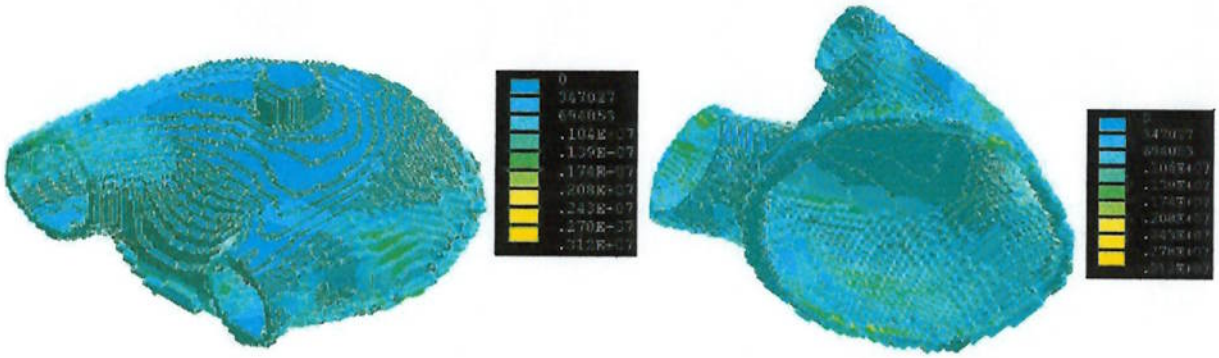


Figura 54 – Resultados da simulação da cavidade.

Nas distribuições de tensões apresentadas, as regiões em azul claro e verde a concentração é menor. Percebe-se inclusive que a região do orifício maior apresentou uma deformação alta que, se necessário, deve ser estudada com cuidado.

Para a espiral os procedimentos foram semelhantes. O modelo obtido no programa de discretização possui cerca de 30 mil elementos. Foram aplicados na base da peça as restrições para que ela não se mova devido a distribuição de pressão ao longo das espirais.

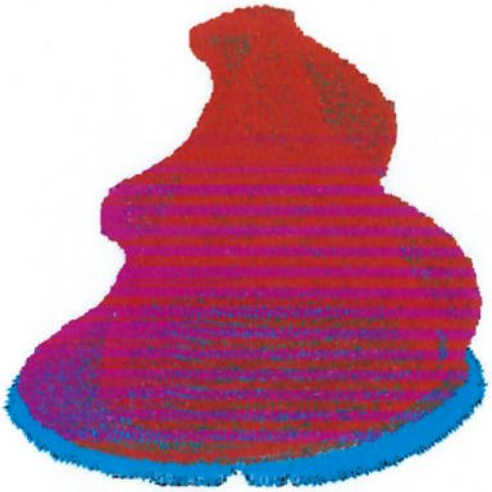


Figura 55 – Espiral submetida às restrições e pressão.

O resultado focado nesta simulação foi a deformação da estrutura mostrada pelo contorno em linha preta.

Figura 56 – Espiral deformada.

6. CONCLUSÃO

Neste trabalho foram apresentados os resultados da pesquisa de discretização de sólidos. O programa feito para a linguagem C, comentando-se com detalhamento como o programa discretiza uma peça e a converte em um modelo de elementos finitos.

Como exemplos de aplicação foram digitalizadas peças complexas modeladas em CAD demostrando a potencialidade do método. O grau de complexidade das peças em

relação foi obtida graças a grande velocidade de processamento que a linguagem C e o ambiente LINUX permite.

O programa foi particularizado para interpretação de imagens geradas pelo Método de Otimização Topológica auxiliando o trabalho do professor orientador. Discutiu-se um exemplo de imagem que foi discretizada com o estudo da influência dos parâmetros no resultado final.

Os resultados iniciais das discretização das peças foram apresentados para pesquisadores do Instituto de Cardiologia Dante Pazzanese e Hospital do Coração da área biomédica e de tomografia, respectivamente, que tomografaram peças biomédicas para este trabalho. Utilizando-se o equipamento de tomografia computadorizada do Hospital do Coração foram obtidas as seções transversais destas peças com facilidade.

O programa de discretização foi aplicado nas peças biomedicas. Foram tomografadas uma cavidade de coração artificial e uma bomba centrífuga de sangue em espiral. As imagens tomografadas de cada uma destas peças sofreram um tratamento simples afim de terem uma resolução adequada para a simulação em um programa de CAE. Foram gerados os modelos de voxels seguido pelo modelo de elementos finitos e as peças foram simuladas aplicando-se pressão e condições de contorno adequados. Os resultados atingidos estiveram dentro do planejado permitindo futuras aplicações mais complexas.

Essa pesquisa gerou artigos que foram aceitos no Congresso Nacional de Automação (CONAI) e apresentado pelo aluno, no congresso IEEE-INDUSCON realizado em Porto Alegre e no XVI Congresso Brasileiro de Engenharia Mecânica (COBEM).

Outra possibilidade de continuação do trabalho diz respeito a uma análise detalhada da simulação de uma peça real em um programa de elementos finitos, modificando seu formato para otimizar seu desempenho.

7. REFERÊNCIAS BIBLIOGRÁFICAS

1) Hanselman, D., Littlefield, B., "MATLAB : versão do estudante : guia do usuário", Makron Books

2) ANSYS, "Getting Started, for Revision 5.1"

3) Kikuchi, N., Hollister, S., and Yoo, J., "A concept of image-based integrated CAE for production engineering", *Proceedings of International Symposium on Optimization and Innovative Design*, Japan, pp.75-90, 1997.

4) Sekiguchi, M., "*Image-Based CAE and its Application to Structural Design*", Dissertation for Master of Science degree, The University of Michigan, 1998.

5) Hollister, S. J. and Kikuchi, N., "Homogenization theory and digital imaging: A basis for studying the mechanics and design principles of bone tissue", *Biotechnology and Bioengineering*, vol. 43, No.7, pp. 586-596, 1994.

6) Hollister, S. J. and B. A. Riemer, "Digital Image Based Finite Element Analysis for Bone Microstructure Using Conjugate Gradient and Gaussian Filter Techniques", *Proceedings of SPIE Mathematical Methods in Medical Imaging II*, Vol.2035, pp. 95-106, 1993.

7) Terada, K. and N. Kikuchi, "Microstructural Design of Composites by Using the Homogenization Method and Digital Images", *Mat. Sci. Res. Int.*, Vol.2, No.2, pp.73-81, 1996.

8) Bendsoe, M. P. and N. Kikuchi, "Generating Optimal Topologies in Structural Design Using a Homogenization Method", *Computer Methods in Applied Mechanics and Engineering*, No.71, 197-224, 1988.

9) Shepard, D., "A two-dimensional function for irregularly spaced data", *Proceedings of the 1968 ACM National Conference*, pp. 517-524, 1968.

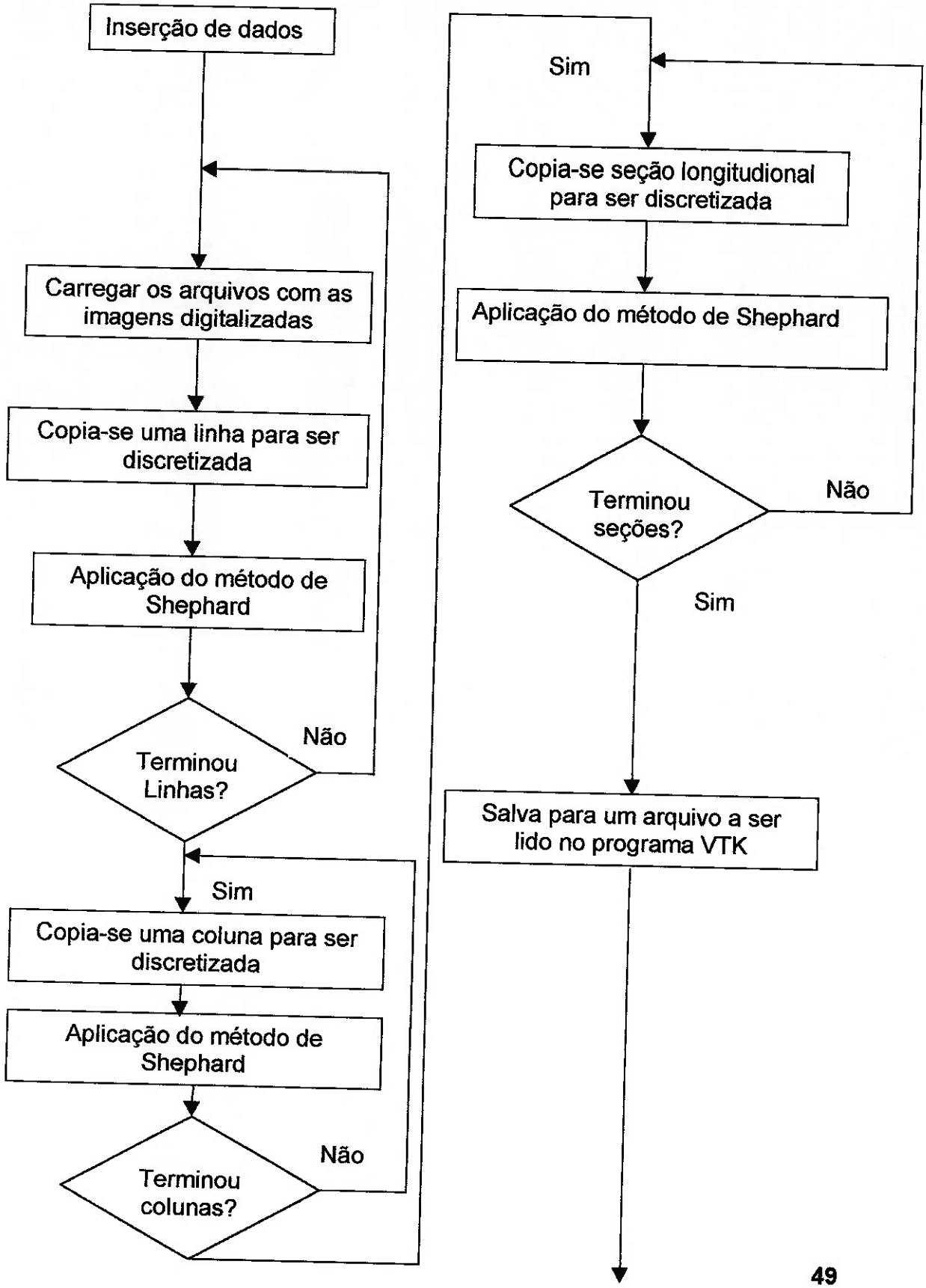
10) Carnaham, B. "Applied Numerical Methods", John Wiley&Sons, 1969.

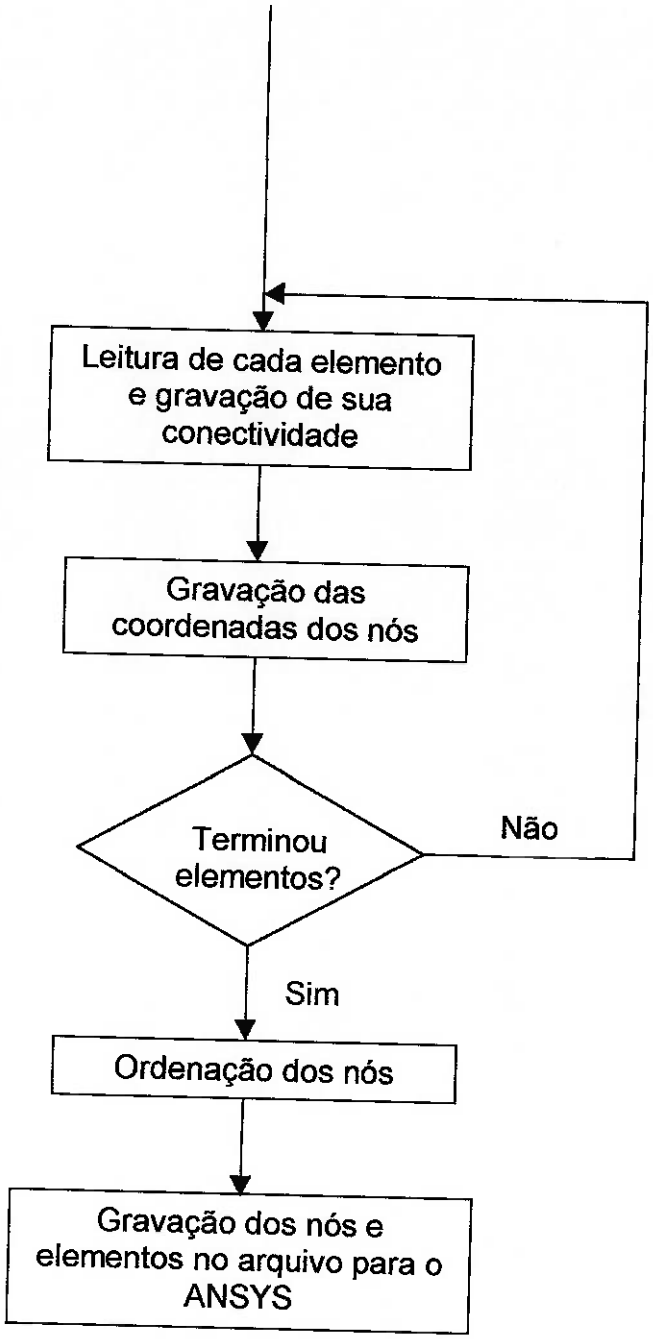
11) Nakamura, S. "Numerical Analysis and Graphic Visualization with MATLAB", Prentice Hall PTR,1996.

12) Kwon, Y.W. e Bang, H. "The Finite Element Method Using MATLAB", CRC Press, 1997

8. APÊNDICE

8.1. Fluxograma: Digicon.c




Leitura de cada elemento e gravação de sua conectividade
Gravação das coordenadas dos nós
Terminou elementos?
Não
Sim
Ordenação dos nós
Gravação dos nós e elementos no arquivo para o ANSYS

8.2. Modelos de discretização bidimensional

8.2.1. Modelo adotado

Para facilitar o entendimento do modelo de discretização de imagens foi adotado um exemplo de matriz bidimensional abaixo. Observe que ela possui 3 linhas e 3 colunas.

$$c = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \text{ ou } c_{ij} \text{ onde } i=1,2,3 \text{ e } j=1,2,3 \quad (1)$$

Primeiro será feita a discretização na horizontal. As funções que passarão pelas linhas terão o seguinte formato:

$$f_i(x)=c_{i1}\phi_{i1}(x)+c_{i2}\phi_{i2}(x)+c_{i3}\phi_{i3}(x) \text{ onde } i=1,2 \text{ e } 3 \quad (2)$$

Se i = 1 corresponde a linha 1, se i = 2 a linha 2 e se i = 3 corresponde a linha 3. Após a discretização da matriz (1) na direção horizontal chega-se a:

$$d = \begin{bmatrix} f_1(x_1) & f_1(x_2) & f_1(x_3) & f_1(x_4) & f_1(x_5) \\ f_2(x_1) & f_2(x_2) & f_2(x_3) & f_2(x_4) & f_2(x_5) \\ f_3(x_1) & f_3(x_2) & f_3(x_3) & f_3(x_4) & f_3(x_5) \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} & d_{15} \\ d_{21} & d_{22} & d_{23} & d_{24} & d_{25} \\ d_{31} & d_{32} & d_{33} & d_{34} & d_{35} \end{bmatrix} \quad (3)$$

Perceba que a dimensão da matriz c passou a ser 3 X 5 após a discretização na horizontal. A próxima discretização será feita na vertical e o formato das funções é mostrado abaixo:

$$F_j(x)=c_{1j}\psi_{1j}(x)+c_{2j}\psi_{2j}(x)+c_{3j}\psi_{3j}(x) \text{ onde } j=1,2 \text{ e } 3 \quad (4)$$

Se j = 1 F(x) corresponde a coluna 1, se j = 2 será coluna 2 e se j = 3 será a coluna 3. Pode-se estender a matriz (3) para o seguinte formato:

$$e = \begin{bmatrix} F_1(x_1) & F_1(x_2) & F_1(x_3) & F_1(x_4) & F_1(x_5) \\ F_2(x_1) & F_2(x_2) & F_2(x_3) & F_2(x_4) & F_2(x_5) \\ F_3(x_1) & F_3(x_2) & F_3(x_3) & F_3(x_4) & F_3(x_5) \\ F_4(x_1) & F_4(x_2) & F_4(x_3) & F_4(x_4) & F_4(x_5) \\ F_5(x_1) & F_5(x_2) & F_5(x_3) & F_5(x_4) & F_5(x_5) \end{bmatrix} = \begin{bmatrix} e_{11} & e_{12} & e_{13} & e_{14} & e_{15} \\ e_{21} & e_{22} & e_{23} & e_{24} & e_{25} \\ e_{31} & e_{32} & e_{33} & e_{34} & e_{35} \\ e_{41} & e_{42} & e_{43} & e_{44} & e_{45} \\ e_{51} & e_{52} & e_{53} & e_{54} & e_{55} \end{bmatrix} \quad (5)$$

A matriz acima corresponde a matriz discretizada. Entre 2 pontos foi criado um novo. Logo, para 3 colunas foram criadas outras duas e para 3 linhas foram criadas mais 2. Poderia-se criar mais pontos entre dois fazendo com que as dimensões da matriz (5) fosse ainda maior.

8.2.2. Modelo bidimensional proposto pela teoria

Na teoria, a proposta de discretização bidimensional de cada imagem segue o modelo:

$$X_{\Omega}(x,y)=\sum_{i=1}^{m}\sum_{j=1}^{n}\phi_i(x)\psi_j(y)c_{ij} \qquad (6)$$

Logo a matriz resultante será:

$$X_{\Omega}=\begin{bmatrix} X(x_1,y_1) & X(x_1,y_2) & \dots & X(x_1,y_{n1}) \\ X(x_2,y_1) & \cdot & & \vdots \\ \vdots & & \cdot & \vdots \\ X(x_m,y_1) & \dots & & X(x_m,y_{n1}) \end{bmatrix} \qquad (7)$$

Este modelo proposto, além de ser mais complexo do que o adotado, é pouco eficiente computacionalmente conforme será exemplificado a seguir na comparação:

Comparação do método teórico e o adotado

Modelo adotado: São realizadas duas seqüências de contas:

a) Serão obtidas 3 linhas e 5 colunas e, em cada posição foram realizadas 3 contas conforme a equação (2), logo: a= 3 x 5 x 3 = 45

b) Serão obtidas 5 linhas e 5 colunas e, em cada posição foram feitas 3 contas de acordo com a equação (4) logo chega-se a: b = 5 x 5 x 3 = 75

Somando-se as contas de (a) e (b) chega-se a: Total = 45 + 75 = 120

Modelo teórico: Serão preenchidas as 5 linhas e 5 colunas da matriz (7) e, para cada posição serão feitas 3 x 3 = 9 contas conforme a equação (6).

Sendo assim Total = 5 x 5 x 3 x 3 = 225

O número de contas realizadas no método teórico é superior ao do adotado. Num modelo mais complexo de imagens com mais linhas e colunas esta diferença de contas será bem maior. Por exemplo, na discretização de seções 30 x 30 para resultar em 59 x 59 obtém-se:

Modelo adotado: 30 x 59 x 30 + 59 x 59 x 30 = 157530

Modelo teórico: 59 x 59 x 30 x 30 = 3132900

O modelo teórico realiza cerca de 20 vezes mais contas que o modelo adotado. Como os resultados são similares o modelo adotado será o de discretização unidimensional nas direções das linhas e colunas.

8.3. Listagem do programa digicon.c

```
/*************************************************************************
            Programa para a discretizacao de secoes
          Versao G - Uso de matrizes do tipo int
                   Programa Otimizado
                   Matizes com alocacao dinamica
                   Discretiza secoes transversais
                   Existe opcao de nao discretizar secoes
                   Formato de saida para o VTK e ANSYS
              Autor: Sergio Eduardo Macedo Rezende
 *************************************************************************/

 # include <stdio.h>
 # include <stdlib.h>
 # include <math.h>
 # define maior 800


                        /* funcao zera */
  int zera (float matr[3][3]){

    matr[1][1]= matr[1][2]= matr[2][1]= matr[2][2]=0;

  }
                      /* funcao soma */

  int soma (float matr1[3][3], float matr2[3][3])
  {

    int i,j;

    for (i=1;i<=2;i++){
      for (j=1;j<=2;j++){

      matr1[i][j]=(matr1[i][j]+matr2[i][j]);

      }
    }

  }
                  /* funcao cria_vetor_colunas */

  int cria_vetor(float vetor_pontos[], float fator,
                    int total_pontos_finais)

  {

    int i;

    vetor_pontos[1]=1;

    for (i=2;i<=total_pontos_finais;i++){

      vetor_pontos[i]=vetor_pontos[i-1]+1/fator;

    }


}
                    /* funcao inverte */

  int inverte (float m[3][3], float a[3][3])
  {
```

```
      float d;

      d=m[1][1]*m[2][2]-m[1][2]*m[2][1];
      a[1][1]=m[2][2]/d;
      a[1][2]=-m[1][2]/d;
      a[2][1]=-m[2][1]/d;
      a[2][2]=m[1][1]/d;

   }


                     /* funcao de Shephard */



   int shephard (float f[], float vetor_pontos[], int total_pontos_iniciais,
                 int total_pontos_finais, float alfa, float F[])
   {

     int     pontos_iniciais, pontos_finais;
     float   matriz[3][3], matriz_aux[3][3], a0, a1, a[3][3],faux,aux;


     for (pontos_finais=1;pontos_finais<=total_pontos_finais;pontos_finais++){

       zera(matriz);
       zera(matriz_aux);

       for (pontos_iniciais=1;pontos_iniciais<=total_pontos_iniciais;
          pontos_iniciais++){

         aux=exp(-alfa*(vetor_pontos[pontos_finais]-
 pontos_iniciais)*(vetor_pontos[pontos_finais]-pontos_iniciais));

         matriz_aux[1][1]=aux;
         matriz_aux[1][2]=pontos_iniciais*aux;
         matriz_aux[2][1]=matriz_aux[1][2];
         matriz_aux[2][2]=pontos_iniciais*pontos_iniciais*aux;

         soma(matriz,matriz_aux);

       }

       inverte(matriz,a);    /* inverte matriz que sera a */

       a0=a[1][1]+a[1][2]*vetor_pontos[pontos_finais];
       a1=a[2][1]+a[2][2]*vetor_pontos[pontos_finais];
       faux=0;

       for (pontos_iniciais=1;pontos_iniciais<=total_pontos_iniciais;
          pontos_iniciais++){

         aux=exp(-alfa*(vetor_pontos[pontos_finais]-
 pontos_iniciais)*(vetor_pontos[pontos_finais]-pontos_iniciais));
         faux=faux+f[pontos_iniciais]*(a0+a1*pontos_iniciais)*aux;

       }

       F[pontos_finais]=faux;

     }
   }
```

```
main() {

FILE * kw;
FILE *arq;
FILE *vtk;


int   linha, tli, tlf, coluna, tci, tcf, secao, tsi, tsf,saida_ansys,
    f1, f2, f3, i=1, k=1, j, n, c, l, posc, disctrans, saida_vtk,
    pos1, pos2, pos3, pos4,pos5, pos6, pos7, pos8, *no;

float  alfa1, alfa2, alfa3, f[maior], F[maior],
     vetor_linhas[maior], vetor_colunas[maior], vetor_secoes[maior],
     threshold, paux, escx, escy, escz,
     num1,numc,nums,x,y,z;

int   *matriz1, *matriz2, *matriz3, *con, *aux;

char  s[100],
    *b,
    estarq;



x=2800;            /* Dados do comprimento, largura e altura da peça */
y=2800;
z=736;

disctrans=1;    /* Opcao para discretizar secoes transversais */

saida_vtk=1;     /* Saida de arquivo no formato VTK */
saida_ansys=0;

f1=3;
f2=3;
f3=1;

alfa1=1.5;
alfa2=1.5;
alfa3=1.5;

threshold=60;

tli=120;
tci=120;
tsi=4;



num1=tlf=((tli-1)*f1+1);
numc=tcf=((tci-1)*f2+1);
nums=tsf=((tsi-1)*f3+1);

matriz1=(int*)malloc((tlf+1)*(tcf+1)*(tsf+1)*sizeof(unsigned int));
matriz2=(int*)malloc((tlf+1)*(tcf+1)*(tsf+1)*sizeof(unsigned int));


s[0]='\0';

strcpy (s,"s1.txt");
```

```
  for(k=1;k<=tsi;k++)
  {

    arq = fopen (s,"r");
    estarq = fscanf(arq,"%s",&b);


    for(j=1;j<=tli;j++)
    {

      for(i=1;i<=tci;i++)
      {

      matriz1[i+tci*(j-1)+tci*tli*(k-1)]=100*atof(&b);
      estarq = fscanf(arq,"%s",&b);

      }

    }


    /*for(j=1;j<=tli;j++)
    {

      for(i=1;i<=tci;i++)
      {

      printf("%d",matriz1[i+tci*(j-1)+tci*tli*(k-1)]);

      }

      printf("\n ");


    }*/



     if (k<9){
      strcpy (s,"s1.txt");
      s[1]=k+'1';
      }

    if (k==9){
      strcpy (s,"s10.txt");
      }


      }


fclose(arq);



cria_vetor(vetor_linhas,f1,tlf);
cria_vetor(vetor_colunas,f2,tcf);
cria_vetor(vetor_secoes,f3,tsf);


             /* Inicio da discretizacao das secoes transversais */

if (disctrans == 1){

  for (secao=1; secao<=tsi; secao++){
    for (linha=1; linha<=tli; linha++){
```

```
        for (coluna=1; coluna<=tci; coluna++){
        f[coluna]=matriz1[coluna+tci*(linha-1)+tci*tli*(secao-1)];
        }
        shephard(f,vetor_colunas,tci,tcf,alfa1,F);
        for(coluna=1;coluna<=tcf;coluna++){
        matriz2[coluna+tcf*(linha-1)+tcf*tli*(secao-1)]=F[coluna];
        }
      }
      for(coluna=1;coluna<=tcf;coluna++){
        for(linha=1;linha<=tli;linha++){
        f[linha]=matriz2[coluna+tcf*(linha-1)+tcf*tli*(secao-1)];
        }
        for(linha=1;linha<=tlf;linha++){
        matriz3[coluna+tcf*(linha-1)+tcf*tlf*(secao-1)]=F[linha];
        }
      }
    }
  }

/* Inicio da discretizacao das secoes longitudionais */
/* Com isto surgirao secoes intermediarias           */

if (disctrans == 1) {
  for(linha=1;linha<=tlf;linha++){
    for (coluna=1; coluna<=tcf; coluna++){
      for (secao=1; secao<=tsi; secao++){
      f[secao]=matriz3[coluna+tcf*(linha-1)+tcf*tlf*(secao-1)];
      }
      shephard (f,vetor_secoes,tsi,tsf,alfa3,F);
      for (secao=1; secao<=tsf; secao++){
      matriz2[coluna+tcf*(linha-1)+tcf*tlf*(secao-1)]=F[secao];
      }
    }
    for (coluna=1; coluna<=tcf; coluna++){
      for (secao=1; secao<=tsf; secao++){
```

```
          if ( matriz2[coluna+tcf*(linha-1)+tcf*tlf*(secao-1)]>threshold )
          matriz2[coluna+tcf*(linha-1)+tcf*tlf*(secao-1)]=1;
          else
          matriz2[coluna+tcf*(linha-1)+tcf*tlf*(secao-1)]=0;

          }
        }
      }

    }            /* fim do disctrans */

    if (saida_vtk == 1) {

      vtk = fopen ("voxvtk","w");

      fprintf(vtk,"# vtk DataFile Version 2.0\n");
      fprintf(vtk,"Figura estranha utilizada para teste do software\n");
      fprintf(vtk,"ASCII\n");
      fprintf(vtk,"DATASET STRUCTURED_POINTS\n");
      fprintf(vtk,"DIMENSIONS %d %d %d\n", tcf, tlf, tsf);
      fprintf(vtk,"ASPECT_RATIO %f %f %f\n", x/tlf, y/tcf, z/tsf);
      fprintf(vtk,"ORIGIN 0 0 0\n");
      fprintf(vtk,"POINT_DATA %d\n", tlf*tcf*tsf);
      fprintf(vtk,"SCALARS scalars unsigned_char 1\n");
      fprintf(vtk,"LOOKUP_TABLE default\n");



      for(k=1;k<=tsf;k++){
        for(j=1;j<=tlf;j++)
        {

          for(i=1;i<=tcf;i++)
          {

          fprintf(vtk," %d",matriz2[i+tcf*(j-1)+tcf*tlf*(k-1)]);

          }

          fprintf(vtk,"\n ");

        }
          fprintf(vtk,"\n ");
      }

   fclose(vtk);

   }

if (saida_ansys == 1){
aux=(int*)malloc((tlf+1)*(tcf+1)*(tsf+1)*sizeof(unsigned int));
   con=(int*)malloc((tlf+1)*(tcf+1)*(tsf+1)*9*sizeof(unsigned int));

   escx=numl/x;      /* Fator de escala nos planos das seções */
   escy=numc/y;
   escz=nums/z;      /* Fator de escala na direção perpendicular às seções */

   for(i=1;i<=(tlf+1)*(tcf+1)*(tsf+1);i++){
     no[3*i-2]=-1;

   }

   posc=1;           /* Posição da conectividade */

   for (n=1;n<=tsf;n++){
     for (c=1;c<=tcf;c++){
       for (l=1;l<=tlf;l++){
```

```
        if ( matriz2[c+tcf*(tlf-1)+tcf*tlf*(n-1)]==1 ){

            /* Início da marcação da conectividade (con) de cada elemento */

          paux = (n-1)*(tlf+1)*(tcf+1);
          pos1 = con[8*posc-7]=((tlf+1)*(c-1)+1+paux);
          pos3 = con[8*posc-6]=((tlf+1)*c+1+paux);
          pos4 = con[8*posc-5]=((tlf+1)*c+1+1+paux);
          pos2 = con[8*posc-4]=((tlf+1)*(c-1)+1+1+paux);

          paux = n*(tlf+1)*(tcf+1);
          pos5 = con[8*posc-3]=((tlf+1)*(c-1)+1+paux);
          pos7 = con[8*posc-2]=((tlf+1)*c+1+paux);
          pos8 = con[8*posc-1]=((tlf+1)*c+1+1+paux);
          pos6 = con[8*posc]=((tlf+1)*(c-1)+1+1+paux);
            posc=posc+1;
          no[3*pos1-2]=(c-1)/escx;
          no[3*pos1-1]=(l-1)/escy;
          no[3*pos1]=(n-1)/escz;
        }
        }
      }
    }

    kw = fopen ("voxel","w");
    fprintf(kw,"/PREP7\n");
    fprintf(kw,"!\n");
    fprintf(kw,"!\n");
    fprintf(kw,"/TITLE, ANSYS 5 job created on 08-Sep-99 at 15:50:19\n");
    fprintf(kw,"!\n");

    j=1;
        /* Início da gravação das conectividades "voxel.txt" */
    fprintf(kw,"!\n");
    fprintf(kw,"/COM Defining Element Types\n");
    fprintf(kw,"!\n");
    fprintf(kw,"ET, 1,45, 0, 0, 0, 0, 0, 0\n");
    fprintf(kw,"/COM Properties for material ""mat1""\n");
    fprintf(kw,"/COM Date: 08-Sep-99           Time: 12:47:59\n");
    fprintf(kw,"MP,EX,1,2.E+9\n");
    fprintf(kw,"MP,PRXY,1,0.4\n");
    fprintf(kw,"MP,DENS,1,2000.\n");
    fprintf(kw,"/COM Properties for material ""mat3""\n");
    fprintf(kw,"/COM Date: 08-Sep-99           Time: 12:47:59\n");
    fprintf(kw,"MP,EX,3,1.E+9\n");
    fprintf(kw,"MP,PRXY,3,0.45\n");
    fprintf(kw,"MP,DENS,3,1000.\n");
    fprintf(kw,"!\n");
    fprintf(kw,"/COM Element props and connectivity for Region:prop1\n");
    fprintf(kw,"!\n");
    fprintf(kw,"MAT,   1\n");
    fprintf(kw,"TYPE,   1\n");
    for (i=1;i<=posc-1;i++){

     fprintf(kw,"EN, %d, %d, %d, %d, %d, %d, %d, %d, %d\n",i,con[8*i-7],con[8*i-
6],con[8*i-5],con[8*i-4],con[8*i-3],con[8*i-2],con[8*i-1],con[8*i]);

    }

     fclose(kw);


  } /* Fim do saida_ansys */


  }
```