4
Modelagem

No capitulo 2 foi descrito como construir e visualizar o *TetraQuad*. Dada uma malha de tetraedros, vimos como fazer para representar uma determinada superfície de nível formada pela interpolação de quádricas posicionadas nos vértices da malha. Este capítulo descreve o passo anterior, bastante conhecido na literatura como *fitting* de pontos, que consiste em, dado um conjunto de pontos, encontrar a superfície que melhor descreva o comportamento desses pontos (figura 4.1). Explicaremos nas próximas linhas o que seria uma boa caracterização para esse conjunto de pontos. O resultado desse processo são os coeficientes das quádricas associados a cada um dos vértices da malha obtidos a partir da solução de um sistema de equações.

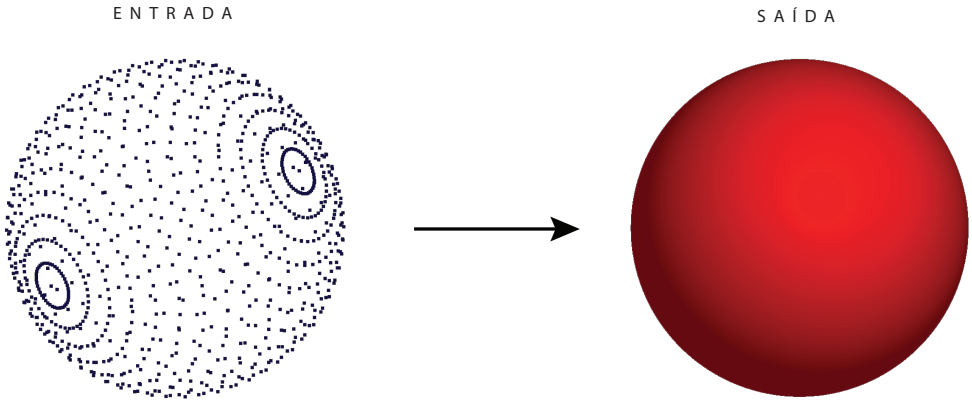


Figura 4.1: Dado um conjunto de pontos, nosso objetivo é encontrar a superfície que melhor descreva o comportamento dos pontos de entrada.

Para exemplificar de forma coerente os diferentes métodos de modelagem, utilizaremos o modelo *Cassini* (figura 4.2) com 3728 pontos como dado de entrada e uma malha de resolução $3\times3\times3$ com 103 tetraedros para construir os exemplos. Nosso objetivo é chegar ao resultado da figura 4.3 através da inclusão de restrições no sistema de equações, bem como alterações na própria malha afim de melhorar a qualidade da superfície resultante do *fitting* de pontos.

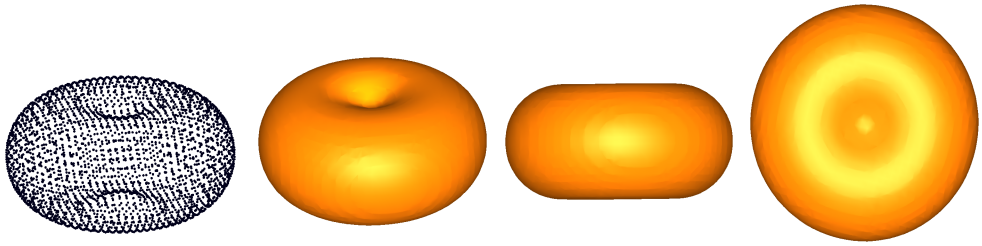


Figura 4.2: Modelo *Cassini* utilizado como entrada para gerar os exemplos deste capítulo. À esquerda a nuvem de pontos e à direita a malha de triângulos.

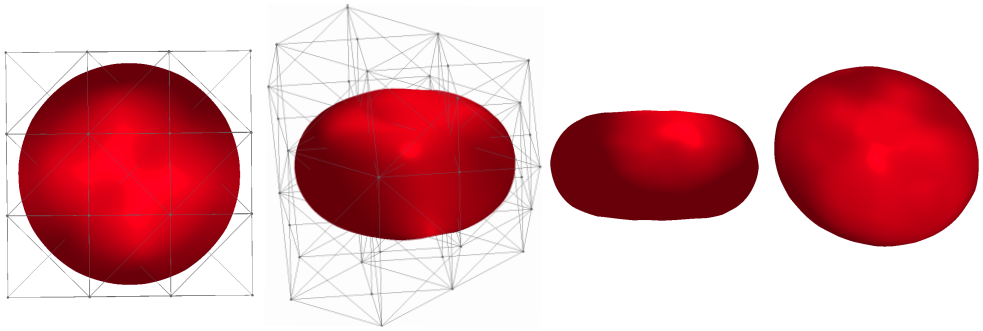


Figura 4.3: Resultado final da reconstrução do modelo *Cassini* utilizando *TetraQuads*. Escolhemos uma malha grosseira de tetraedros para evidenciar os problemas encontrados.

4.1 *Fitting* de Pontos e Mínimos Quadrados

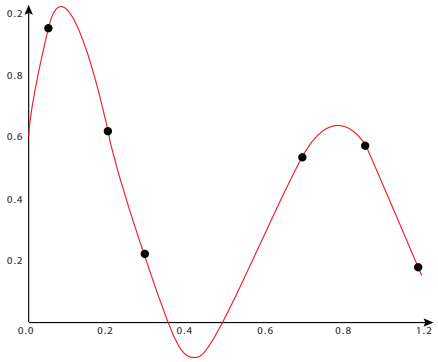
Como já mencionamos, nosso objetivo é, dado um conjunto com m pontos de entrada, encontrar a superfície que melhor representa esse conjunto de pontos. Mas o que seria uma boa representação para esse conjunto? Uma solução para esse problema seria encontrar um polinômio que interpole os pontos em questão. Esse é um problema conhecido que consiste em resolver um sistema quadrático de equações para achar um polinômio interpolante que é único e e que possui no máximo grau $m - 1$ (19).

Para facilitar o entendimento, pensemos no exemplo bidimensional da figura 4.4(a) em que são dados seis pontos de entrada $x_1, ..., x_6 \in \mathbb{R}$ com valores $y_1, ..., y_6$ associados a cada um deles. Queremos encontrar a equação

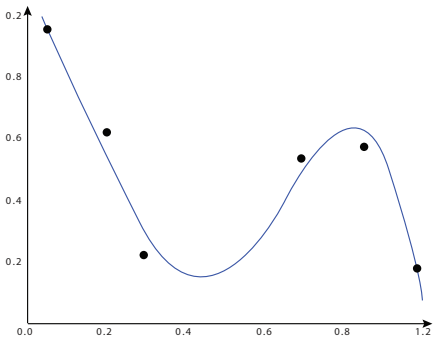
$$F(x) = c_0 + c_1 x + c_2 x^2 + ... + c_5 x^5$$

onde para cada um dos valores x_i temos $F(x_i) = y_i$ que formam um sistema de equações não singular (se todos os x_i's são distintos) fácil de ser resolvido.

Entretanto, como pode ser observado na figura, a interpolação polinomial pode atribuir características indesejadas à curva como picos que a afastam muito dos pontos de entrada.



4.4(a): Interpolação polinomial.



4.4(b): Mínimos quadrados.

Figura 4.4: Obtenção de curvas. À esquerda, curva obtida utilizando-se interpolação polinomial. À direita, curva obtida utilizando-se o método dos mínimos quadrados.

Sendo assim, preferimos aproximar os pontos do que interpolá-los. Queremos achar uma curva, não necessariamente interpolante dos pontos de entrada, que se aproxime ao máximo dos pontos mantendo suas características como na figura 4.4(b). Precisamos encontrar um polinômio de grau menor que $m-1$, no nosso exemplo de grau 4, que forma um sistema com maior número de equações do que incógnitas dito indeterminado. A resolução desse problema formulado como mínimos quadrados consiste em encontrar o polinômio

$$F(x) = c_0 + c_1x + c_2x^2 + c_3x^3 + c_4x^4$$

que minimize a soma

$$\sum_{i=1}^{4} |F(x_i) - y_i|^2.$$

No caso do *TetraQuad*, para achar os coeficientes das quádricas nos vértices da malha de tetraedros temos uma lista de equações a serem satisfeitas para cada ponto de entrada formando um sistema indeterminado (o número de pontos é maior que o de coeficientes a serem determinados). Assim, temos um sistema de equações do tipo $Ax = b$, $A \in \mathbb{R}^{m\times n}$ e $b \in \mathbb{R}^m$ com $m \geq n$ (se houver pontos suficientes nos tetraedros), e queremos minimizar o valor da norma do resíduo r dado por

$$r = b - Ax.$$

Ou seja, achar o vetor solução x que minimiza a distancia algébrica de cada ponto de entrada para o *TetraQuad*:

$$\min_{\text{quadricas } Q_v} \sum_{n} ||r||,$$

onde Q_v indica a quádrica associada a cada vértice v.

Esse é um problema bastante conhecido e como tal possui uma série de algoritmos criados para resolvê-lo. Falaremos um pouco sobre cada um dos métodos utilizados neste trabalho para resolução de mínimos quadrados. Mas antes, vejamos uma descrição melhor do processo de construção desse sistema de equações.

4.2 Construção do Sistema

Existem algumas possibilidades de escolha de equações para formar o sistema, a cada uma delas associamos um peso (um valor real positivo) que determina o quanto cada equação irá influenciar no resultado final do sistema. A equação principal, a qual damos peso 1, é aquela que define a superfície de nível 0, $F(x, y, z) = 0$, mas existem outras possibilidades, como utilizar a equações do gradiente nos pontos da superfície, reduzir a influência dos coeficientes de grau 3, etc. O desafio é encontrar as melhores restrições para o sistema e o peso dado a cada uma delas.

A dimensão do sistema é determinada através de quatro fatores. O número de pontos (n_p) junto com o número de restrições (n_r) indicam o número de linhas ou de equações que o sistema precisa atender. Já o número de vértices da malha (n_v), que forma a partição do domínio vista no início do trabalho, e o de coeficientes (n_c) que desejamos determinar por quádrica, nove para o caso em que J é fixo ou dez quando queremos usar a minimização para determinar todos os coeficientes, indicam o número de incógnitas.

O sistema é definido de forma matricial da seguinte forma:

$$M_{n_p \cdot n_r \times n_v \cdot n_c} \cdot X_{n_v \cdot n_c \times 1} = U_{n_p \cdot n_r \times 1}. \tag{4-1}$$

Três matrizes fazem parte desse sistema, a matriz M que contém os coeficientes da parte homogênea, a matriz X com as incógnitas (coeficientes das quádricas não fixados) e a matriz U com a parte não-homogênea das equações.

O preenchimento da matriz M leva em consideração que cada linha representa uma equação relativa a um tetraedro da malha. Os elementos da linha são preenchidos de acordo com os coeficientes do tetraedro que a representa,

as outras entradas são preenchidas com zero tornando a matriz cada vez mais esparsa a medida que o número de tetraedros aumenta. Veremos a importância dessa característica mais a frente ao estudarmos os algoritmos para resolução de sistemas de equações. A figura 4.5 ilustra o comportamento das primeiras linhas da matriz em um caso básico como o descrito na próxima seção. Utilizamos duzentos tetraedros e as marcações em preto indicam entradas não nulas referentes a um vértice da malha.

Figura 4.5: Primeiras linhas da matriz M em um caso básico. As marcações indicam entradas não-nulas.

Mesmo para o caso básico ($F = 0$), o sistema contempla as equações de todos os tetraedros garantindo que a superfície resultante seja continua nas faces que não são de bordo, ou seja, aquelas que pertencem a dois tetraedros distintos, mas não garante sua suavidade nesses casos. O fato da superfície possivelmente não ser suave nessas faces provoca algumas limitações, como a impossibilidade de reconstruir o campo de normais nas curvas que representam essa região como vimos no final do capítulo 2. Poderemos constatar isso nos resultados ao longo do capítulo.

4.3 Caso Básico

Começamos pelo caso básico em que aparece apenas a equação da superfície de nível zero que definimos da seguinte forma:

$$F(x, y, z) = C(p) = 0. \tag{4-2}$$

Onde p é o ponto de coordenadas $(x, y, z, 1)$ e C a equação (2-8) definida no capítulo 2. Nesse primeiro momento o coeficiente J é fixado como uma constante (ex.: $J = -1$). Nesse cenário, o sistema fornece n_p equações, uma para cada ponto de entrada, e $9n_v$ incógnitas (nove coeficientes por vértice da malha), resultando na seguinte equação:

$$M_{n_p \times 9n_v} \cdot X_{9n_v \times 1} = U_{n_p \times 1}. \tag{4-3}$$

Para facilitar a compreensão, consideremos que são dados n_p pontos de entrada e que a nossa malha possui apenas um tetraedro de vértices v_1, v_2, v_3 e v_4 como descrito na figura 4.6. Temos então o seguinte sistema:

$$M_{n_p \times 36} \cdot X_{36 \times 1} = U_{n_p \times 1}. \tag{4-4}$$

As incógnitas compõem a matriz X e o valor de J com o sinal trocado compõe a matriz U. A equação abaixo descreve melhor os elementos de cada uma das matrizes.

$$\begin{array}{c} \\ p_1 \\ \\ p_i \\ \\ p_n \end{array}\!\!\begin{array}{c} \begin{array}{ccc} v_1 & & \end{array} \\ \begin{bmatrix} m_{1\,1} & \dots & m_{1\,9} \\ \vdots & \ddots & \vdots \\ m_{i\,1} & \dots & m_{i\,9} \\ \vdots & \ddots & \vdots \\ m_{n\,1} & \dots & m_{n\,9} \end{bmatrix} \end{array}\!\!\begin{array}{c} \begin{array}{c} \\ \end{array} \\ \begin{bmatrix} \dots \\ \ddots \\ \dots \\ \ddots \\ \dots \end{bmatrix} \end{array}\!\!\begin{array}{c} \begin{array}{ccc} & & v_4 \end{array} \\ \begin{bmatrix} m_{1\,28} & \dots & m_{1\,36} \\ \vdots & \ddots & \vdots \\ m_{i\,28} & \dots & m_{i\,36} \\ \vdots & \ddots & \vdots \\ m_{n\,28} & \dots & m_{n\,36} \end{bmatrix} \end{array} \begin{bmatrix} A_1 \\ B_1 \\ \vdots \\ I_1 \\ \vdots \\ A_4 \\ B_4 \\ \vdots \\ I_4 \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

Os elementos da 1ª à 9ª colunas representam a quádrica associada ao vértice v_1, da 10ª à 18ª a quádrica do vértice v_2 e assim por diante. Abaixo podemos observar o valor do elemento da i-ésima linha e primeira coluna de M.

$$m_{i\,1} = x_i^2 z_i b_{13} + x_i^3 b_{11} + x_i^2 y_i b_{12} + x_i^2 b_{14}$$

A figura 4.7 ilustra um resultado obtido com o caso básico. Como podemos observar, essa não foi, em sua totalidade, uma reconstrução adequada devido ao surgimento de diversos artefatos indesejados que por sua vez não descrevem o comportamento dos pontos de entrada do modelo.

Essa restrição sozinha não foi suficiente para reconstruir corretamente a maioria das superfícies. Por esse motivo, foi testada a inclusão de novas equações ao sistema.

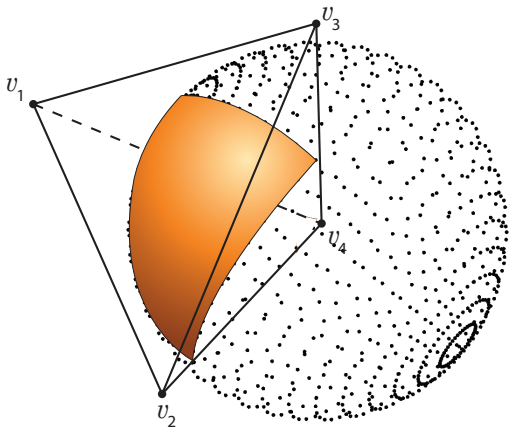


Figura 4.6: Visão geral do caso simples empregando apenas um tetraedro.

Figura 4.7: Reconstrução do modelo *Cassini* obtida através do caso básico: $F = 0$. Observe a diversidade de artefatos indesejados que não descrevem adequadamente o comportamento dos pontos de entrada.

4.4 Incluindo as Normais dos Vértices da Superfície

A segunda restrição testada foi a inclusão da relação de paralelismo entre as normais dos pontos de entrada e o gradiente do *TetraQuad* nesses pontos.

$$[{}^{\partial C}/_{\partial x}, {}^{\partial C}/_{\partial y}, {}^{\partial C}/_{\partial z}] \times [n_x, n_y, n_z] = 0. \tag{4-5}$$

Esse fato inclui uma linha no sistema para cada coordenada da normal (n_x, n_y e n_z) resultando em $4n_p$ equações (uma $C(p) = 0$ e três da normal, sendo uma redundante). Considerando ainda o exemplo do início do capítulo, um tetraedro e n_p pontos de entrada, chegamos ao seguinte sistema:

$$M_{4n_p \times 36} \cdot X_{36\times 1} = U. \tag{4-6}$$

Os elementos das matrizes referentes ao i-ésimo ponto do modelo estão representados abaixo.

$$\begin{bmatrix} \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ m_{4(i-1)+1\,1} & \cdots & m_{4(i-1)+1\,9} & \cdots & m_{4(i-1)+1\,28} & \cdots & m_{4(i-1)+1\,36} \\ m_{4(i-1)+2\,1} & \cdots & m_{4(i-1)+2\,9} & \cdots & m_{4(i-1)+2\,28} & \cdots & m_{4(i-1)+2\,36} \\ m_{4(i-1)+3\,1} & \cdots & m_{4(i-1)+3\,9} & \cdots & m_{4(i-1)+3\,28} & \cdots & m_{4(i-1)+3\,36} \\ m_{4(i-1)+4\,1} & \cdots & m_{4(i-1)+4\,9} & \cdots & m_{4(i-1)+4\,28} & \cdots & m_{4(i-1)+4\,36} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \end{bmatrix} \begin{bmatrix} A_1 \\ B_1 \\ \vdots \\ I_1 \\ \vdots \\ A_4 \\ B_4 \\ \vdots \\ I_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Essa restrição melhorou a suavidade, em particular das normais. Entretanto, aumentou consideravelmente o tamanho do sistema e funcionou bem (gerou resultado suave mantendo a topologia do objeto) apenas para alguns modelos comportados, mas não foi suficiente para modelos mais complexos (com características geométricas difíceis de representar) como o *Bunny*. Podemos observar isso na figura 4.8 onde aparecem componentes espúrias.

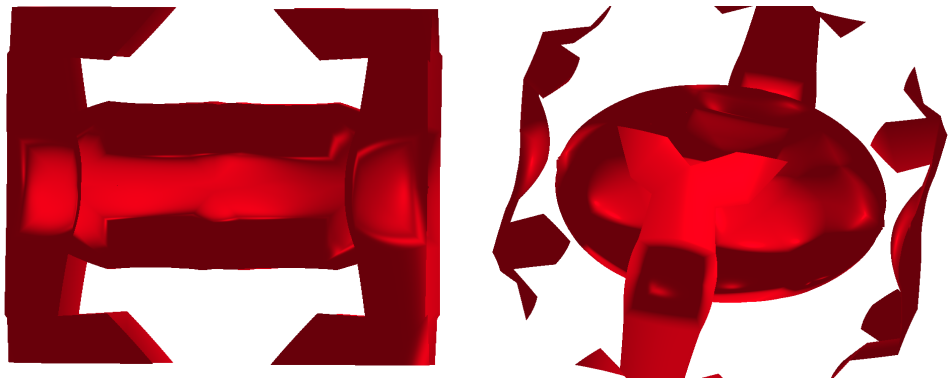


Figura 4.8: Reconstrução do modelo *Cassini* obtida através do caso básico ($F = 0$) e o ajuste das normais dos vértices e da superfície.

4.5 Redução da Influência dos Coeficientes de Grau 3

É mais difícil de descrever a superfície quando existem mais de duas raízes dentro do tetraedro. Isso quer dizer que para cada raio que atravessa

o tetraedro são encontradas três interseções com a superfície. Muitas vezes esse não é o comportamento esperado, por esse motivo, resolvemos diminuir esses casos para facilitar a reconstrução. Isso é feito igualando cada um dos elementos de grau 3 da equação (4-2) a zero forçando a expressão da superfície a estar mais próxima de um polinômio de grau 2. A equação abaixo exemplifica essa restrição para o coeficiente de x^3.

$$A_1 b_{11} + A_2 b_{21} + A_3 b_{31} + A_4 b_{41} = 0. \tag{4-7}$$

Consideramos o mesmo exemplo simples onde A_i representa o primeiro coeficiente da quádrica associada ao vértice i e b_{ij} representa o elemento da linha i e coluna j da matriz de coordenadas baricêntricas do tetraedro.

A nova restrição inclui 10 novas equações por tetraedro ao sistema (com três variáveis, existem dez coeficientes de grau 3). Pensando no nosso exemplo básico com o sistema formado apenas pela equação $F(x, y, z) = 0$ e as equações com os coeficientes de grau 3, chegamos à seguinte expressão:

$$M_{n_p+10n_s \times 36} \cdot X_{36\times 1} = U. \tag{4-8}$$

Os elementos referentes ao vértice v_1 do nosso exemplo estão representados abaixo.

$$\begin{bmatrix} \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ m_{n+1\,1} & \dots & m_{n+1\,9} & \dots & m_{n+1\,28} & \dots & m_{n+1\,36} \\ m_{n+2\,1} & \dots & m_{n+2\,9} & \dots & m_{n+2\,28} & \dots & m_{n+2\,36} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ m_{n+10\,1} & \dots & m_{n+10\,9} & \dots & m_{n+10\,28} & \dots & m_{n+10\,36} \end{bmatrix} \begin{bmatrix} A_1 \\ B_1 \\ \vdots \\ I_1 \\ \vdots \\ A_4 \\ B_4 \\ \vdots \\ I_4 \end{bmatrix} = \begin{matrix} \\ u_{n+1} \\ \\ \\ u_{n+9} \\ u_{n+10} \end{matrix} \begin{bmatrix} \vdots \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

O ideal é que essa restrição seja combinada com outras. Apenas a redução da influência dos altos coeficientes não foi suficiente para gerar resultados satisfatórios a partir do nosso modelo de exemplo como podemos observar na figura 4.9.

4.6 Diferentes Superfícies de Nível

Uma outra regularização inspirada pelas funções de base radial (RBF) do trabalho de Carr et al (5) foi empregada com o objetivo de construir

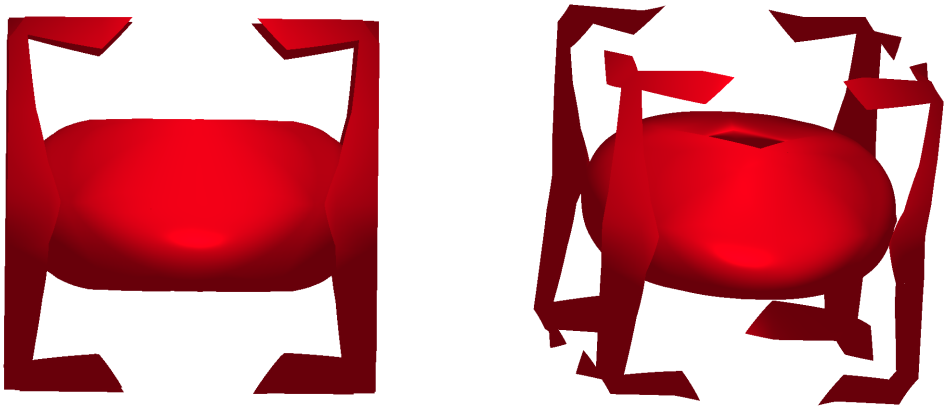


Figura 4.9: Reconstrução do modelo *Cassini* com $F = 0$ e redução da influência dos coeficientes de grau 3.

uma extrapolação suave da superfície eliminando as componentes espúrias e reduzindo a formação de buracos na malha. A figura 4.10 ilustra a ideia da abordagem. Construímos dois conjuntos de pontos novos movimentando os pontos de entrada na direção da normal de cada um deles e definimos novos valores para F.

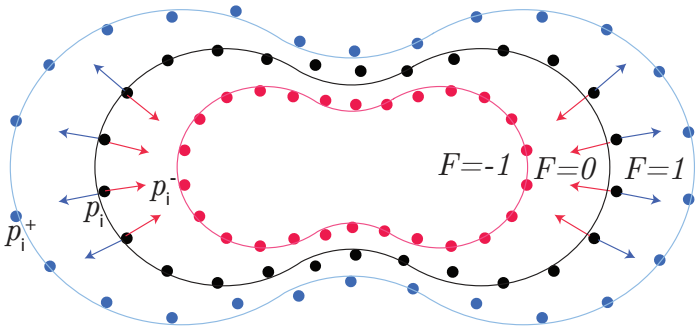


Figura 4.10: Construção dos novos conjuntos de pontos para $F = -1$ e $F = 1$.

Primeiramente obtemos os novos conjuntos de pontos projetando os pontos originais na direção da normal ($p_i^+ = p_i + s\vec{n}_{p_i}$) e depois na direção contrária ($p_i^- = p_i - s\vec{n}_{p_i}$). Para cada um dos pontos de entrada originais $F(p_i) = 0$, nos pontos movimentados na direção da normal, $F(p_i^+) = 1$, e na direção contrária, $F(p_i^-) = -1$. Em ambos os casos, s é um escalar positivo e $\vec{n}_{p_i}$ a normal no ponto p_i. Isso cria três linhas para cada ponto na matriz M. Considerando $F = 0$ com J fixo e as duas novas equações (F=1 e F=-1), temos o seguinte sistema:

$$M_{3n_p \times 36} \cdot X_{36\times 1} = U. \tag{4-9}$$

Os elementos das matrizes para esse caso estão descritos abaixo.

$$\begin{matrix} \\ p_i^+ \\ p_i \\ p_i^- \\ \\ \end{matrix}\begin{bmatrix} \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ m_{3(i-1)+1\,1} & \cdots & m_{3(i-1)+1\,9} & \cdots & m_{3(i-1)+1\,28} & \cdots & m_{3(i-1)+1\,36} \\ m_{3(i-1)+2\,1} & \cdots & m_{3(i-1)+2\,9} & \cdots & m_{3(i-1)+2\,28} & \cdots & m_{3(i-1)+2\,36} \\ m_{3(i-1)+3\,1} & \cdots & m_{3(i-1)+3\,9} & \cdots & m_{3(i-1)+3\,28} & \cdots & m_{3(i-1)+3\,36} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \end{bmatrix} \begin{bmatrix} A_1 \\ B_1 \\ \vdots \\ I_1 \\ \vdots \\ A_4 \\ B_4 \\ \vdots \\ I_4 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 0 \\ \vdots \\ 2 \\ 1 \\ 0 \end{bmatrix}$$

Essa restrição sozinha mostrou bons resultados com o modelo *Cassini* como entrada (figura 4.11). Entretanto, como poderá ser observado no capítulo de resultados, apenas as equações das superfícies de nível não foram suficientes para reconstruir satisfatoriamente modelos mais complexos.

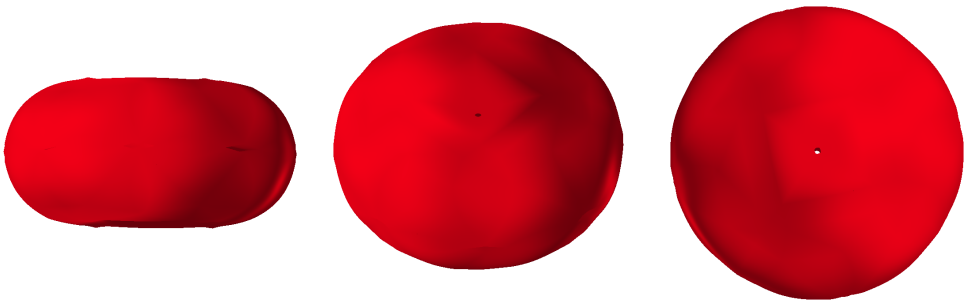


Figura 4.11: Reconstrução do modelo *Cassini* empregando diferentes superfícies de nível.

4.7 Coeficiente J

Nos exemplos anteriores estávamos fixando um valor para o coeficiente constante das quádricas que chamamos de J. Testamos incluir esse coeficiente como incógnita no sistema. Isso acrescenta uma nova coluna à matriz M e o número de linhas da matriz de incógnitas X passa a ser $10n_v$, onde n_v é o número de vértices da malha de tetraedros. A matriz U por sua vez é formada apenas pelo isovalor desejado, no nosso exemplo, esse isovalor é 0. Demos o nome de "J livre" a essa nova restrição. A equação 4-10 representa esse caso.

$$M_{n_p \times 40} \cdot X_{40\times 1} = U. \tag{4-10}$$

Os coeficientes das matrizes estão descrito abaixo.

$$\begin{array}{c} \\ p_1 \\ \\ p_i \\ \\ p_n \end{array}\!\!\begin{array}{c} \begin{array}{cccccccc} & & v_1 & & | & \dots & | & & v_4 & \end{array} \\ \begin{bmatrix} m_{1\,1} & \dots & m_{1\,10} & \dots & m_{1\,31} & \dots & m_{1\,40} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ m_{i\,1} & \dots & m_{i\,10} & \dots & m_{i\,31} & \dots & m_{i\,40} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ m_{n\,1} & \dots & m_{n\,10} & \dots & m_{n\,31} & \dots & m_{n\,40} \end{bmatrix} \end{array} \begin{bmatrix} A_1 \\ B_1 \\ \vdots \\ J_1 \\ \vdots \\ A_4 \\ B_4 \\ \vdots \\ J_4 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

Os algoritmos de mínimos quadrados testados tinham a limitação de resolver apenas sistemas do tipo não-homogêneos. Por essa razão, combinações de restrições como a que acabamos de descrever junto com a equação das normais, por exemplo, não foram testadas. Resolvemos esse problema incluindo restrições que possuem parte não-homogênea em suas equações como é o caso da inclusão de novas superfícies de nível da seção 4.6. Podemos visualizar os elementos para um sistema com diferentes superfícies de nível e J livre abaixo.

$$\begin{array}{c} \\ p_i^+ \\ p_i \\ p_i^- \\ \\ \end{array}\!\!\begin{bmatrix} \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ m_{3(i-1)+1\,1} & \dots & m_{3(i-1)+1\,10} & \dots & m_{3(i-1)+1\,31} & \dots & m_{3(i-1)+1\,40} \\ m_{3(i-1)+2\,1} & \dots & m_{3(i-1)+2\,10} & \dots & m_{3(i-1)+2\,31} & \dots & m_{3(i-1)+2\,40} \\ m_{3(i-1)+3\,1} & \dots & m_{3(i-1)+3\,10} & \dots & m_{3(i-1)+3\,31} & \dots & m_{3(i-1)+3\,40} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \end{bmatrix} \begin{bmatrix} A_1 \\ B_1 \\ \vdots \\ J_1 \\ \vdots \\ A_4 \\ B_4 \\ \vdots \\ J_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \\ \vdots \\ 1 \\ 0 \\ -1 \end{bmatrix}$$

A figura 4.12 foi construída a partir das equações do sistema formado com o J livre e as diferentes superfícies de nível. Essas duas restrições junto com a redução dos altos coeficientes geraram os melhores resultados como veremos no final desse trabalho.

Uma interpretação dessa melhoria é que quando fixamos o coeficiente J, uma compensação é feita nos coeficientes dos outros termos do polinômio tornando o sistema totalmente dependente da posição das cúbicas em relação à origem. Incluir o J no sistema diminui essa dependência pois os outros termos permanecem iguais na origem ou longe dela e apenas o valor de J é alterado. Para validar esse fato, fizemos alguns testes com os modelos *Cúbica* e *Cassini* em três cenários: $F(x, y, z) = 0$, $F(x, y, z) = 0$ com diferentes superfícies de nível e J livre com diferentes superfícies de nível. Nas figuras 4.14 e 4.15 os

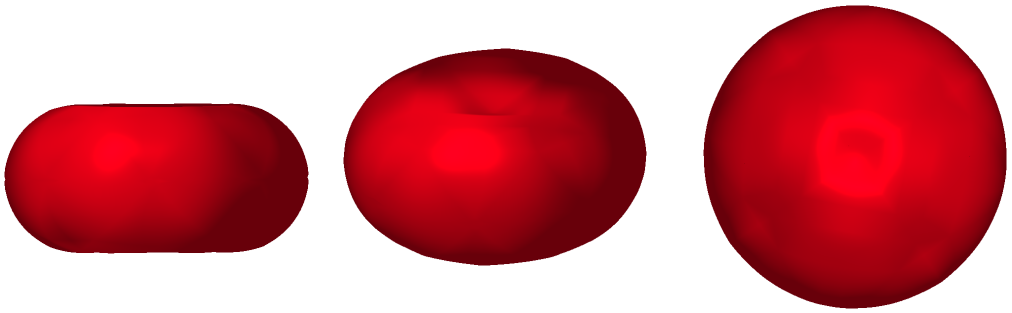


Figura 4.12: Modelo *Cassini* reconstruído com J livre e diferentes superfícies de nível.

modelos estão centrados na origem e em 4.16 e 4.17 longe da origem. O modelo *Cúbica* pode ser observado através da figura 4.13.

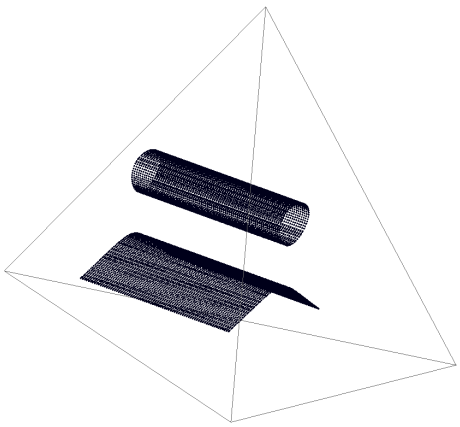
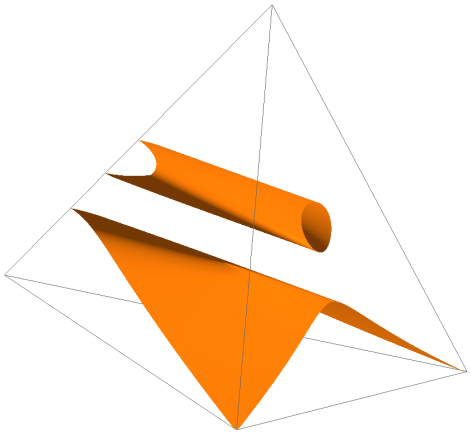
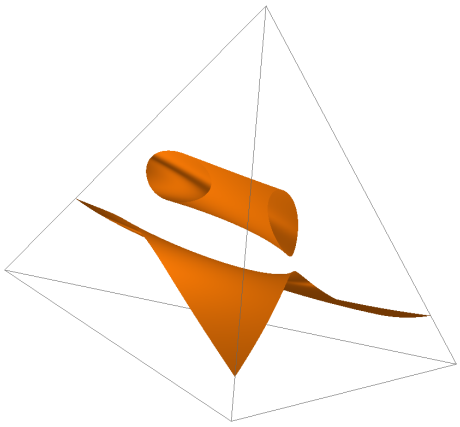


Figura 4.13: Pontos do modelo *Cúbica*.

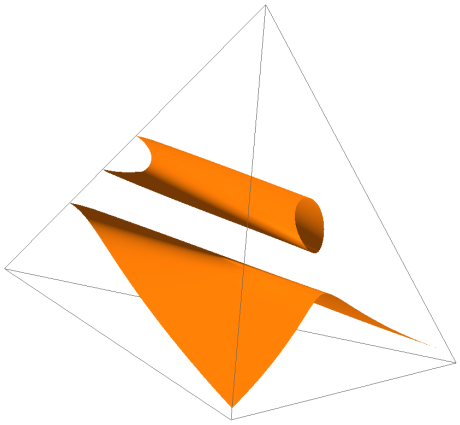
Como pode ser observado pelos testes, junto à origem, o resultado do sistema tende a ser mais comportado, uma vez que as superfícies estão mais próximas dos pontos de entrada. Longe da origem, a inclusão do coeficiente J no sistema gera resultados melhores que o caso básico. A inclusão das diferentes superfícies de nível também demonstraram ter um papel importante principalmente para reconstruções longe da origem.



4.14(a): F=0.

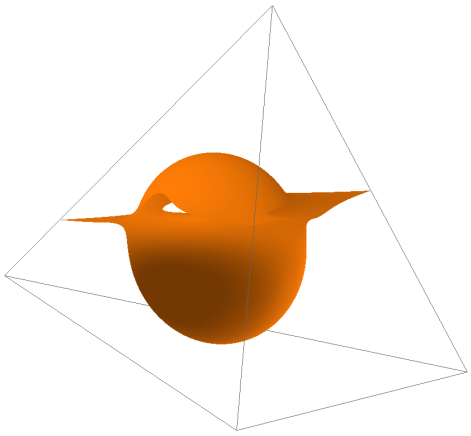


4.14(b): F=0 com superfícies de nível.

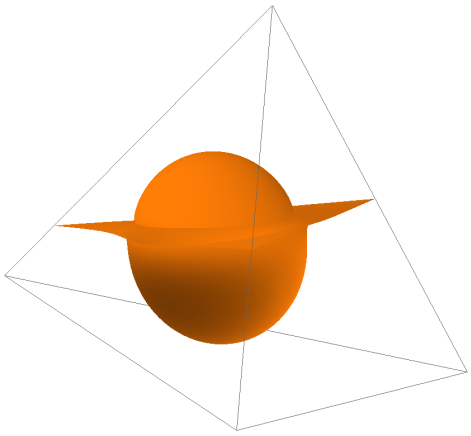


4.14(c): J livre com superfícies de nível.

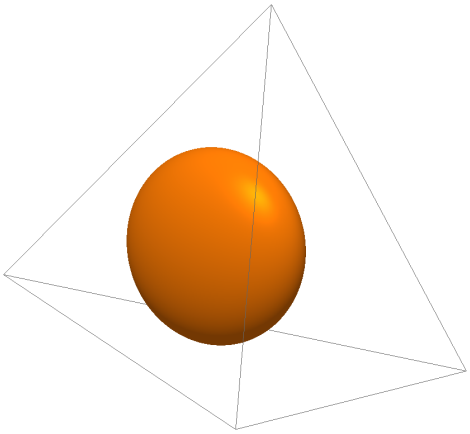
Figura 4.14: Testes com o modelo *Cúbica* centrado na origem.



4.15(a): F=0.

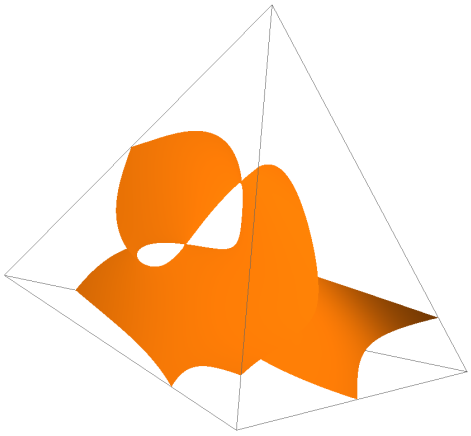


4.15(b): F=0 com superfícies de nível.

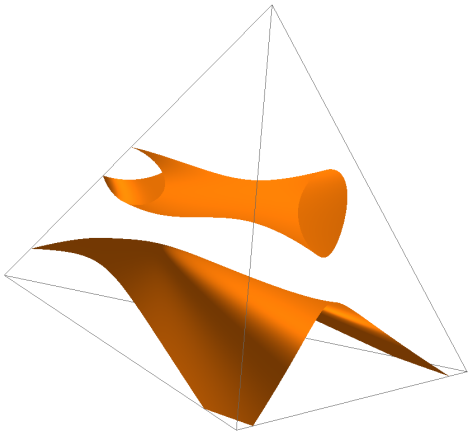


4.15(c): J livre com superfícies de nível.

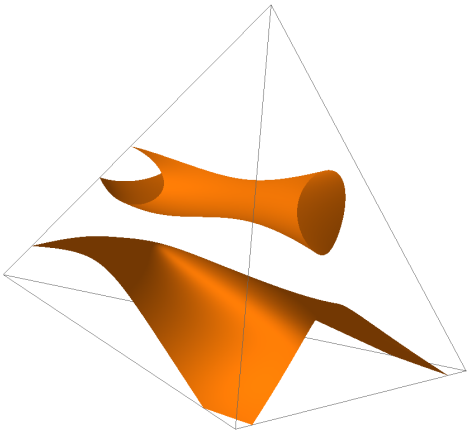
Figura 4.15: Testes com o modelo *Cassini* centrado na origem.



4.16(a): F=0.

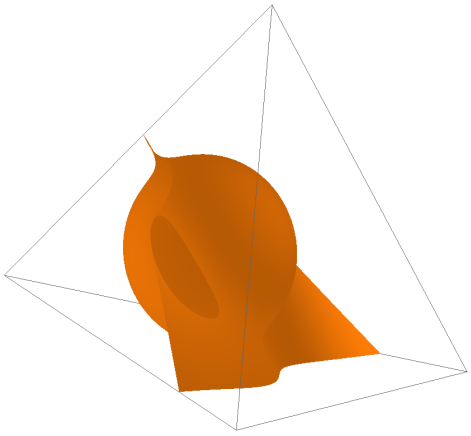


4.16(b): F=0 com superfícies de nível.

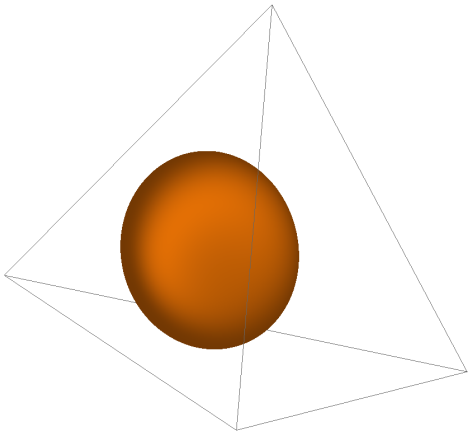


4.16(c): J livre com superfícies de nível.

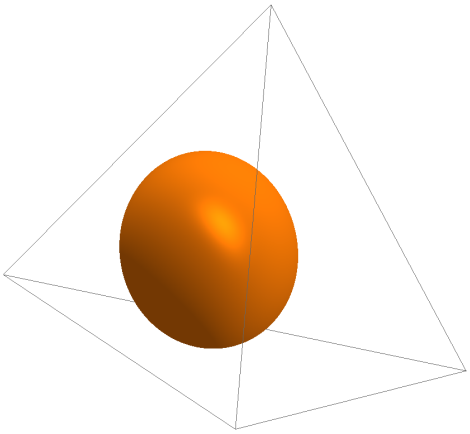
Figura 4.16: Testes com o modelo *Cúbica* afastado da origem.



4.17(a): F=0.



4.17(b): F=0 com superfícies de nível.



4.17(c): J livre com superfícies de nível.

Figura 4.17: Testes com o modelo *Cassini* afastado da origem.

4.8 Alterações na Malha

Mesmo depois de definir quais equações devem entrar no sistema para garantir uma boa solução, ainda continuamos tendo problemas na reconstrução da superfície em determinadas situações. A boa notícia é que esses problemas se concentravam em determinadas regiões do modelo, em alguns momentos a qualidade do *fitting* não era boa e em outros continuaram aparecendo componentes espúrias mesmo com uma boa reconstrução. Para minimizar ou eliminar esse tipo de problema resolvemos testar alterações na estrutura da malha de tetraedros removendo tetraedros contendo poucos pontos de entrada, alterando a posição espacial dos vértices ou aumentando o número de tetraedros.

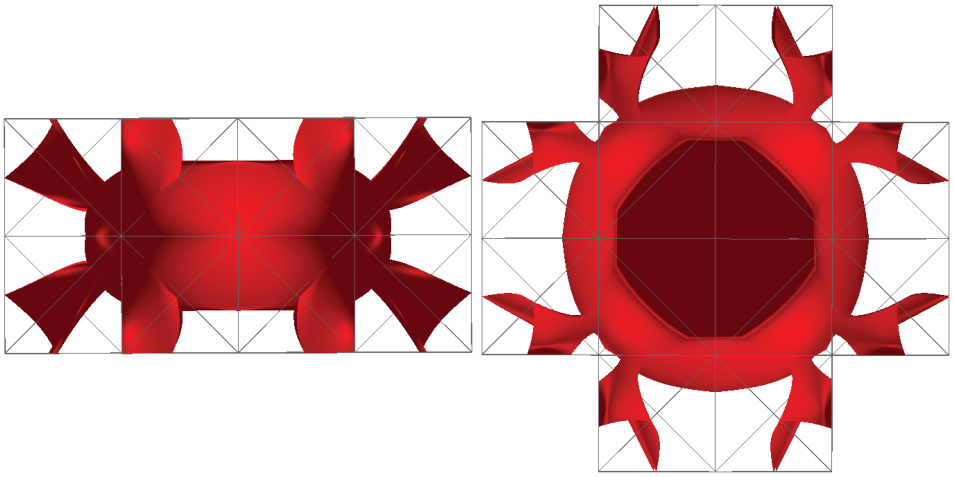
4.8.1 Mínimo de Pontos por Tetraedro

A primeira situação que observamos é que algumas componentes espúrias, como planos que aparecem destacados da superfície, surgiam na reconstrução em tetraedros com um número pequeno de pontos do modelo. Quanto menor for o número de pontos, mais difícil é de determinar a cúbica que case corretamente com a superfície utilizando nosso problema de mínimos quadrados pois a matriz M pode ficar com posto deficiente. Resolvemos, por esse motivo, determinar o número mínimo de pontos que o tetraedro deve conter para ser incluído no sistema e o resultado pode ser observado através da figura 4.18.

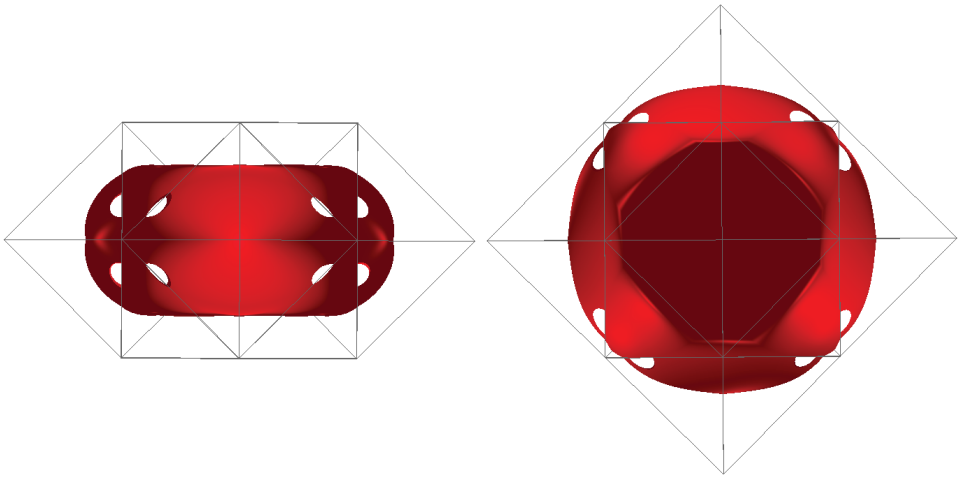
Os tetraedros que ainda persistirem com o problema podem ser excluídos posteriormente após a rescontrução para visualização. Entretanto, seja qual for o caso, eliminar esses tetraedros pode inserir buracos na superfícies em regiões em que a reconstrução correta não foi possível como pode ser observado através da figura 4.18(b). O resultados foram gerados apenas com a restrição básica ($F = 0$).

4.8.2 Relaxamento dos Vértices

Dependendo de características do conjunto de pontos de entrada, como a sua proximidade em relação aos vértices da malha, a reconstrução se mostra problemática em determinados tetraedros como podemos observar através da figura 4.19. Esse é o mesmo exemplo da seção anterior, obtido apenas com a restrição básica. Queremos mostrar uma outra maneira de resolver o mesmo problema.



4.18(a): Empregando número mínimo de um ponto por tetraedro.



4.18(b): Empregando número mínimo de seis ponto por tetraedro.

Figura 4.18: Reconstruções obtidas com limitação do número de pontos por tetraedro.

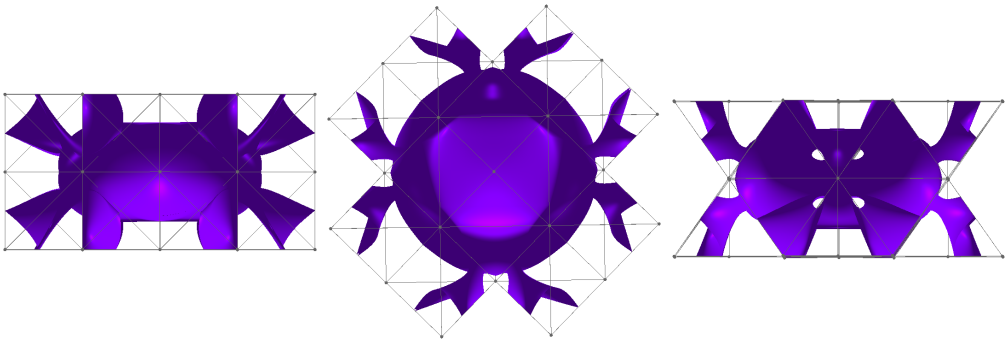


Figura 4.19: Resultado obtido com o relaxamento dos vértices.

Incentivados por casos como o da figura 4.19, em que partes da superfície que estão próximas dos vértices da malha apresentam problemas, resolvemos testar a influência da distância dos vértices em relação ao conjunto de pontos a ser reconstruído. Procuramos alterar essa distância, tornando cada vez mais o conjunto de pontos que "atravessa" as arestas equidistantes dos seus vértices. Implementamos um algoritmo que soma pesos às coordenadas dos vértices da malha de acordo com a distância até os pontos de entrada. Para simplificar nossa explicação, iremos nos deter ao caso 2D o qual pode ser ilustrado através da figura 4.20. A extensão para tetraedros ocorre de maneira direta.

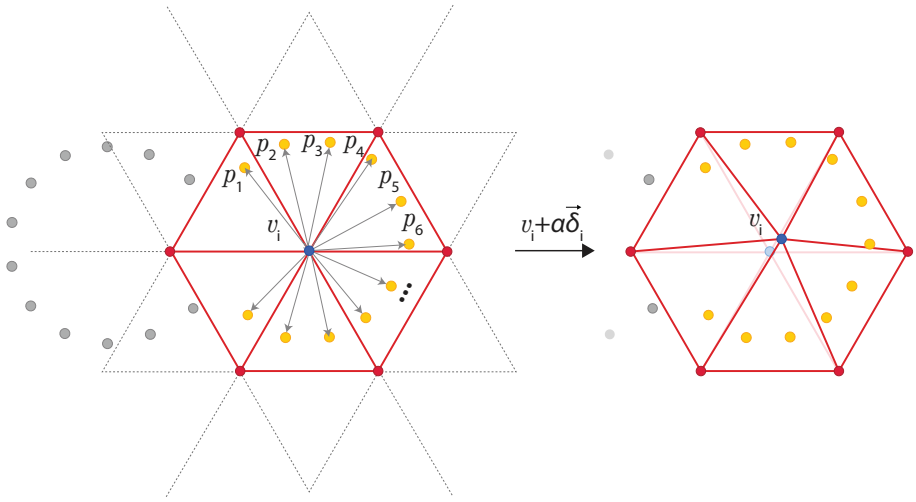


Figura 4.20: Iteração do algoritmo para o vértice v_i.

O algoritmo segue da seguinte maneira: para cada vértice v_i da malha de triângulos é calculado um vetor $\vec{\sigma}_i$ que é dado pela soma de todos os vetores $v_i\vec{p}_j$ formado pelos pontos p_j que estão dentro de triângulos dos quais v_i faz parte. Assim:

$$\vec{\sigma}_i = \sum_{p_j \in \Delta_{v_i}} v_i\vec{p}_j,$$

onde Δ_{v_i} é o conjunto de triângulos da malha que possuem o vértice v_i. A nova posição do vértice v_i é então dada por:

$$v_i = v_i + \alpha\vec{\sigma}_i.$$

O valor do escalar α é definido de acordo com a malha e esse processo pode ser repetido várias vezes até chegarmos à configuração desejada para a posição dos vértices. A única restrição é que esse procedimento não pode ser realizado para os vértices na borda pois desestrutura toda a malha.

Voltando ao exemplo do início da seção, as figuras 4.21 e 4.22 exibem a grade utilizada antes e depois de sofrer o relaxamento dos vértices. O resultado da figura 4.23 foi obtido após o relaxamento dos vértices.

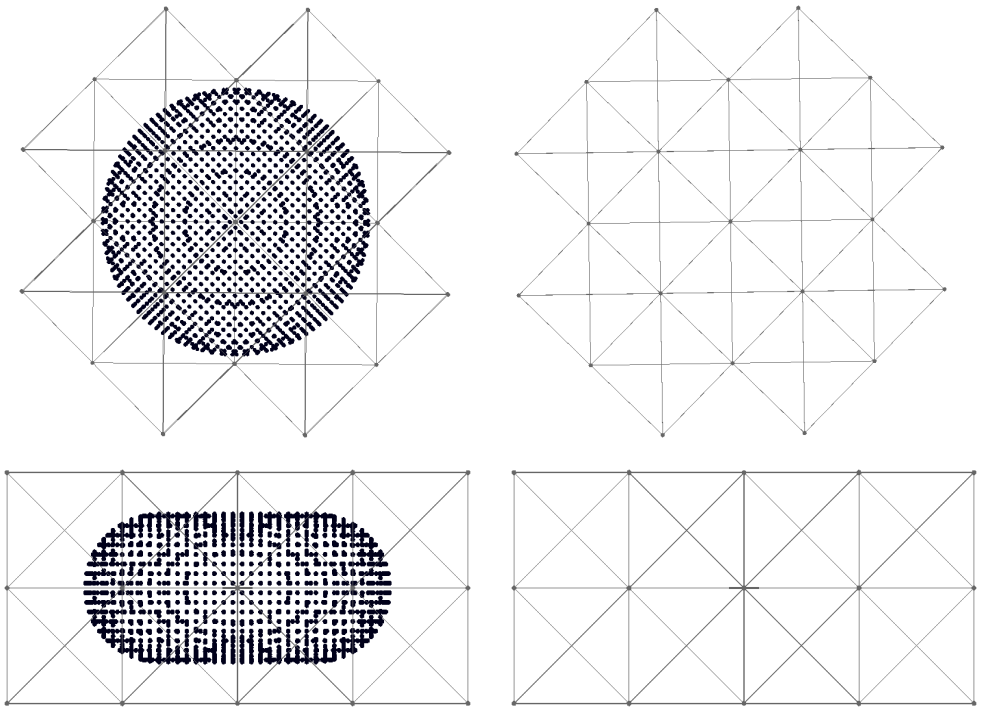


Figura 4.21: Grade sem o relaxamento dos vértices.

Através de resultados como o da figura 4.23, observamos que as arestas ideais possuem vértices equidistantes à superfície, ou seja, quando há interseção, a cúbica deve passar no centro das arestas. Passamos a levar em consideração esse fator para montagem do sistema. Como no caso anterior, do número mínimo de pontos por tetraedros, alterações posteriores à reconstrução na posição dos vértices do *TetraQuad* podem ser feitas para melhorar o resultado da visualização.

4.8.3 Aumentar a Resolução da Malha

Para melhorar a qualidade da reconstrução, uma última possibilidade é aumentar o número de tetraedros da malha. Entretanto, esbarramos em duas limitações ao aumentarmos demasiadamente a resolução da malha corremos o risco de criar muitos tetraedros com poucos pontos no interior gerando componentes espúrias (figura 4.24) ou ainda podemos inviabilizar o *TetraQuad*,

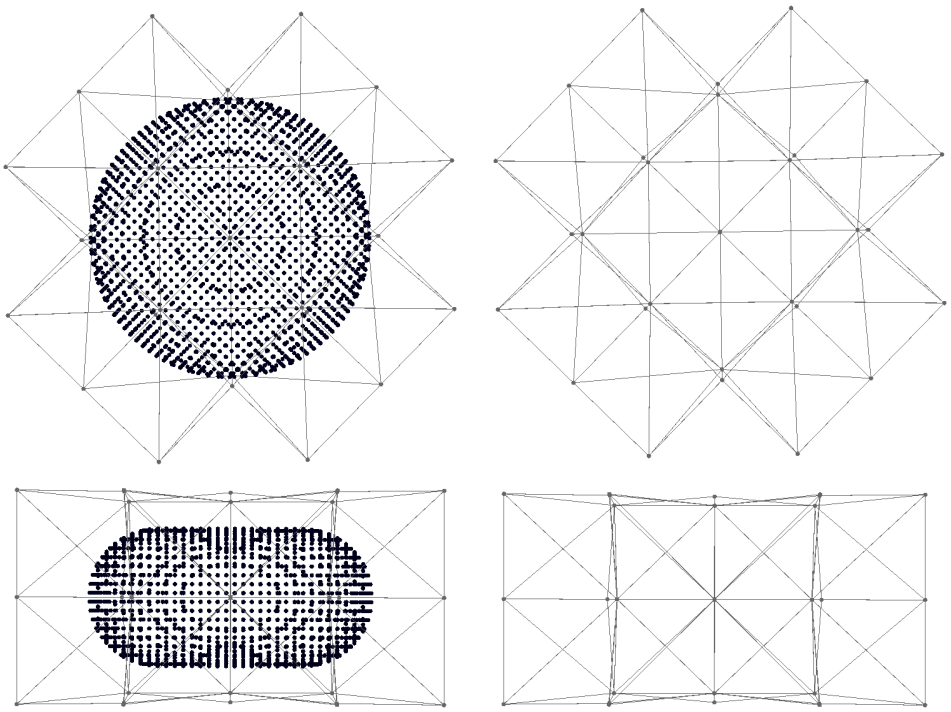


Figura 4.22: Grade com o relaxamento dos vértices.

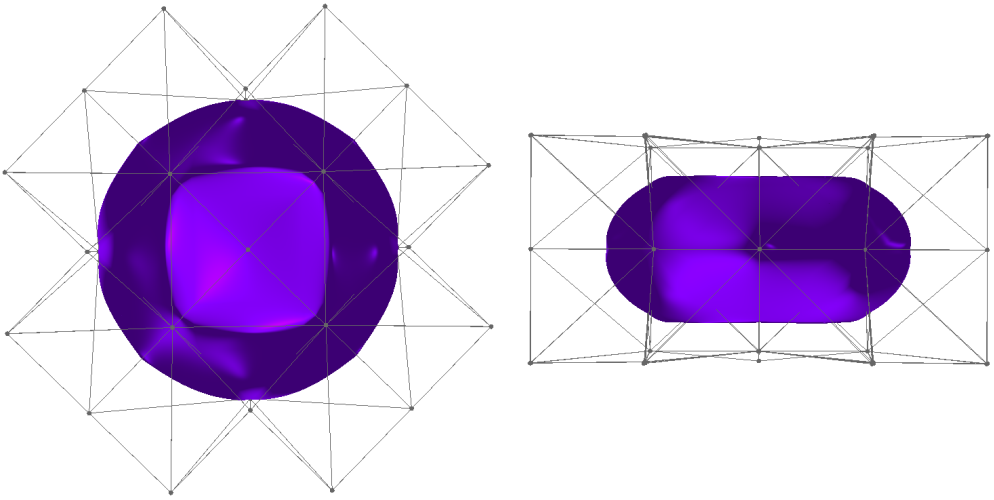


Figura 4.23: Resultado com o relaxamento dos vértices.

tornando a estrutura mais difícil de processar que a malha de triângulos original do modelo.

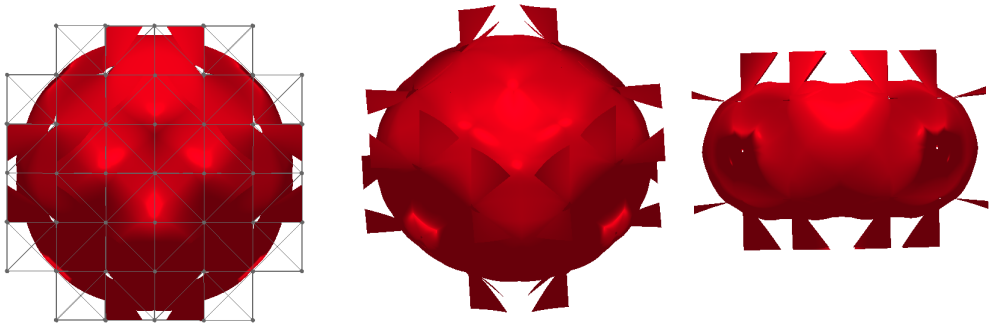


Figura 4.24: Reconstrução empregando malha com resolução de $8 \times 8 \times 8$.

Na figura 4.25 reduzimos a resolução da malha para $6 \times 6 \times 6$ com 272 tetraedros. Um último exemplo aparece na figura 4.26 combinando o aumento de resolução da malha, relaxamento dos vértices, redução dos altos coeficientes, superfícies de nível e o coeficiente J livre. Resultados melhores podem ser vistos no próximo capítulo onde descrevemos uma ferramenta para medir a qualidade das reconstruções.

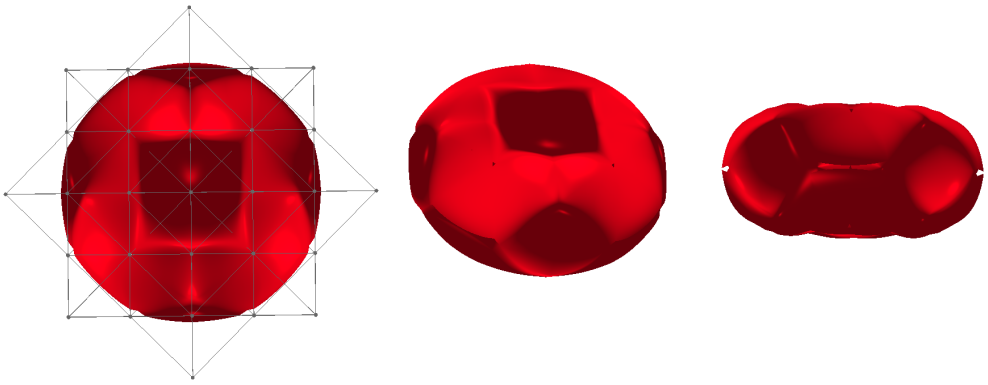


Figura 4.25: Exemplo com malha de resolução $6 \times 6 \times 6$.

4.9 Métodos para Resolução de Mínimos Quadrados

Testamos os algoritmos implementados em algumas bibliotecas de álgebra linear para solução do problema de mínimos quadrados. Uma dessas bibliotecas possui implementação em GPU, garantindo um processamento mais rápido se comparada as mesmas implementações em CPU. Uma aferição da qualidade

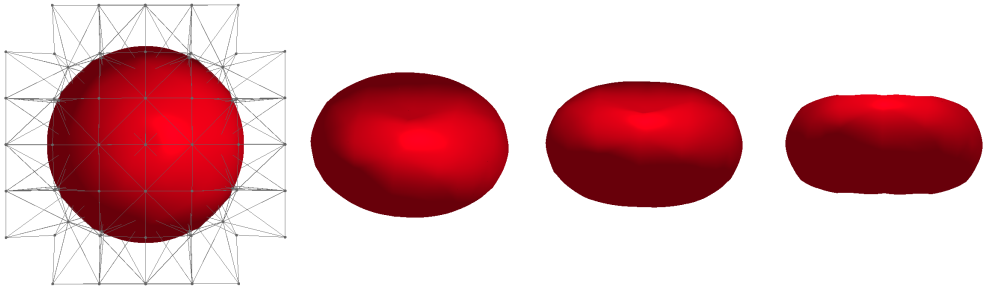


Figura 4.26: Resultado com aumento de resolução da malha, relaxamento dos vértices, redução dos altos coeficientes, superfícies de nível e coeficiente J livre.

de resultados e performance dessas implementações pode ser encontrada no capítulo de resultados. Nesta seção iremos fazer uma breve caracterização dos algoritmos utilizados. Uma explanação mais completa dessas rotinas e alguns detalhes de implementação podem ser encontrados no livro de Trefethen e Bau III (19). Mas antes de passarmos adiante, vejamos qual a ideia em comum entre eles.

Como vimos na seção 4.1, ao resolvermos o problema de mínimos quadrados queremos encontrar o valor $x \in \mathbb{R}^n$ que minimiza a norma $||r|| = ||Ax - b||$ do resíduo r. Isso equivale a achar o vetor Ax que está mais próximo de b levando em consideração a norma euclidiana. Isso ocorre quando

$$Ax = Pb, \tag{4-11}$$

onde $P \in \mathbb{R}^{m \times m}$ é o projetor ortogonal que mapeia vetores em R^m no espaço gerado pelas colunas da matriz A chamado de $range(A)$ que é obtido resolvendo $A^*r = 0$. Isso nos leva a uma outra equação equivalente chamada de equação normal dada por:

$$A^*Ax = A^*b. \tag{4-12}$$

Em todos os casos, A^* indica a conjugada transposta da matriz A, que nesse caso, como estamos considerando apenas matrizes com entradas reais, coincide com a sua transposta. Por fim, se A tem posto máximo, esse sistema é não singular e pode ser resolvido utilizando a fatoração de Cholesky.

Para evitar confusão de notação, ao falar do problema de mínimos quadrados, utilizamos a forma clássica $Ax = b$ como no início do capítulo. A matriz A equivale à matriz M e os vetores x e b equivalem, respectivamente, aos vetores ou matrizes coluna X e U.

4.9.1 Fatoração QR

Um dos algoritmos mais empregados na resolução de mínimos quadrados utiliza a versão reduzida da fatoração QR para facilitar a solução do problema o descrevendo como um sistema triangular superior. Aplicar a fatoração ao nosso problema consiste em definir a equação (4-1) da seguinte maneira:

$$\hat{R}x = \hat{Q}b, \tag{4-13}$$

onde $\hat{Q}$ e $\hat{R}$ são as matrizes obtidas encontrando a fatoração QR reduzida da matriz A obtidas através de reflexões de Householder.

Essa formulação é ideal para ser utilizada quando a matriz A com $m \geq n$ tem posto máximo n e pode ser descrita como combinação linear dos elementos das colunas $q_1, q_2, ..., q_n$ da matriz ortonormal $\hat{Q}$ e da matriz triangular superior $\hat{R}$ da seguinte maneira

$$\begin{bmatrix} a_1 & | & a_2 & | & \dots & | & a_n \end{bmatrix} = \begin{bmatrix} q_1 & | & q_2 & | & \dots & | & q_n \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ & r_{22} & & \vdots \\ & & \ddots & \\ & & & r_{nn} \end{bmatrix},$$

que nos leva a seguinte equação:

$$A = \hat{Q}\hat{R}, \tag{4-14}$$

que é chamada de versão reduzida da fatoração QR. Podemos chegar à equação (4-13) utilizando o projetor $P = \hat{Q}\hat{Q}^*$ e a equação (4-14).

Algoritmos que utilizam a fatoração QR para a solução de mínimos quadrados estão entre os mais comumente encontrados nas bibliotecas de álgebra linear e tem complexidade $O(mn^2 + n^3)$. Utilizamos versão em GPU desse algoritmo da biblioteca ViennaCL (20) que reduz consideravelmente a quantidade de interações necessárias por ser implementado empregando programação paralela e com aproveitamento da esparsidade da matriz A.

4.9.2 SVD

Quando a matriz A não tem posto máximo, o método QR fica ambíguo. Nesse caso, podemos usar o SVD, um dos algoritmos de decomposição mais importantes. O SVD tem uma série de interpretações, dentre as quais a de que uma matriz $A \in \mathbb{R}^{m \times n}$, $m \geq n$, pode ser vista como uma transformação que leva a esfera unitária $S \in \mathbb{R}^m$ em uma hiperelipse $AS \in \mathbb{R}^m$ (figura 4.27).

Considerando que A tem posto n podemos vê-la como uma composição de escalas por fatores $\sigma_1, \sigma_2, ..., \sigma_n$ não nulos nas direções ortogonais dadas por vetores unitários $u_1, u_2, ..., u_n \in \mathbb{R}^m$. Por conveniência, podemos ordenar os fatores tais que $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n$ e numerar os vetores u_i's de forma a coincidir com essa ordenação.

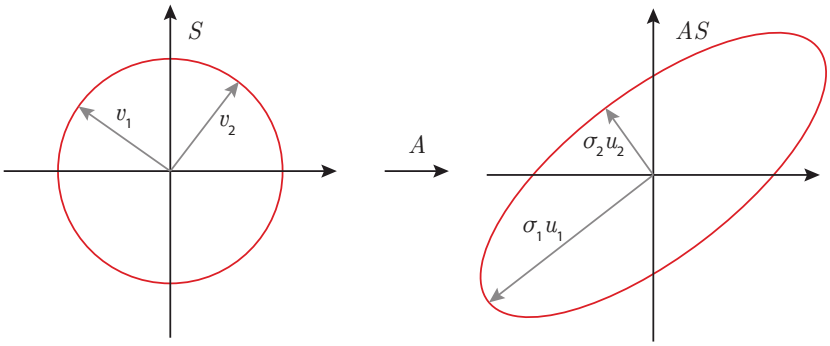


Figura 4.27: Exemplo em $\mathbb{R}^2$.

Por último, vamos nomear $\{v_1, v_2, ..., v_n\} \in S$ os vetores ortonormais tais que $Av_i = \sigma_i u_i$. Assim, podemos decompor a matriz A da seguinte maneira:

$$A \begin{bmatrix} v_1 & | & v_2 & | & \dots & | & v_n \end{bmatrix} = \begin{bmatrix} u_1 & | & u_2 & | & \dots & | & u_n \end{bmatrix} , \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} .$$

Se chamarmos de $\hat{U} \in \mathbb{R}^{m \times n}$ a matriz de colunas ortonormais u_i's, $\hat{\Sigma} \in \mathbb{R}^{n \times n}$ a matriz diagonal formada pelos valores σ_i's e $\hat{V} \in \mathbb{R}^{n \times n}$ a matriz unitária podemos redefinir A da seguinte maneira

$$A = \hat{U}\hat{\Sigma}\hat{V}^*, \tag{4-15}$$

onde $\hat{V}^*$ é a transposta da matriz $\hat{V}$ que coincide com sua inversa. A equação (4-15) é conhecida como SVD reduzido da matriz A pois descarta os valores singulares nulos (ou numericamente pequenos), permitindo trabalhar com matrizes A com posto deficiente.

É a versão reduzida da decomposição em valores singulares que é utilizada para resolver problema de mínimos quadrados reduzindo-o a um sistema diagonal de equações trivialmente resolvido dado pela equação:

$$\hat{\Sigma}V^*x = \hat{U}^*b, \tag{4-16}$$

que é obtida a partir da equação (4-15) utilizando o operador $P = \hat{U}\hat{U}^*$.

Esse método funciona adequadamente para sistemas bem comportados já que trabalha com resolução de sistemas de equações não singulares e em alguns aspectos demonstra-se mais preciso que a fatoração QR com a mesma complexidade $O(mn^2+n^3)$ (19), mas com um número maior de iterações. Para este trabalho foi utilizada uma versão do algoritmo disponível na biblioteca GSL (21) que não se beneficia do fato da matriz A ser esparsa.

4.9.3 Método do Gradiente Conjugado

Os métodos que acabamos de descrever exigem grande esforço computacional. Esse inconveniente é sentido no tempo de execução a cada vez que aumentamos o número de pontos de entrada ou o número de tetraedros do sistema. Em alguns casos os algoritmos nem chegam a convergir para uma solução. Uma alternativa a essa abordagem são os chamados métodos iterativos. Esses métodos aproveitam melhor características das matrizes como esparsidade, que é o caso do *TetraQuad*, reduzindo o número de passos necessários para solucionarmos o problema de mínimos quadrados. Nesta seção, descreveremos o método iterativo utilizado neste trabalho chamado de método do gradiente conjugado. Nas próximas linhas daremos uma breve descrição do método.

Comecemos por considerar o conjunto de vetores p_i's mutuamente conjugados em relação à matriz A definido por:

$$P = \{p_i \in \mathbb{R}^n | \forall p_i \neq p_j, i, j \in [1, n], p_i^\top A p_j = 0\}.$$

O conjunto P forma uma base para $\mathbb{R}^n$ e consequentemente podemos obter a solução x_* para o nosso sistema como combinação dos seus vetores da seguinte maneira:

$$x_* = \sum_{i=1}^{n} \alpha_i p_i. \tag{4-17}$$

O método do gradiente conjugado consiste em acharmos uma aproximação x_n para o vetor solução x_* que minimiza o resíduo

$$||r_n|| = ||b - Ax_n|| \tag{4-18}$$

ou equivalentemente que minimiza o erro

$$||e_n||_A = ||x_* - x_n||_A, \tag{4-19}$$

onde $||\cdot||_A$ é a norma dada por $||x||_A = \sqrt{x^\top Ax}$.

Encontramos essa aproximação através de iterações sucessivas nas di-

reções mutuamente conjugadas dadas pelos vetores do conjunto P. Podemos resumir o método através do algoritmo clássico de Hestenes e Stiefel (22), que também mostra como podemos obter os primeiros vetores do conjunto P a cada iteração para garantir a convergência. O algoritmo encerra quando o resíduo r_n é suficientemente pequeno.

Algorithm 1 Algoritmo do Gradiente Conjugado.

$x_0 = 0$, $r_0 = b$, $p_0 = r_0$

for $n = 1, 2, 3, ...$

$\alpha_n = (r_{n-1}^\top r_{n-1})/(p_{n-1}^\top A p_{n-1})$	passo
$x_n = x_{n-1} + \alpha_n p_{n-1}$	solução aproximada
$r_n = r_{n-1} - \alpha_n A p_{n-1}$	resíduo
$\beta_n = (r_n^\top r_n)/(r_{n-1}^\top r_{n-1})$	
$p_n = (r_n)/(\beta_n p_{n-1})$	nova direção de procura

Nosso objetivo foi reduzir o tempo de espera e melhorar a qualidade dos resultados aumentando o volume de dados de entrada para o sistema. Considerando que estamos trabalhando com matrizes quadradas $m \times m$, conseguimos reduzir a complexidade dos algoritmos anteriores que era da grau de $O(m^3)$ para $O(m^2)$. Entretanto, o método é mais sensível à configuração da malha de tetraedros, à densidade da nuvem de pontos de entrada e às restrições do sistema do que os métodos anteriores pois esses fatores aumentam a dependência do resultado da solução inicial.

Existe uma série de métodos do tipo gradiente conjugado, um deles é o LSQR. Utilizamos a implementação de Barré desenvolvida para uma ferramenta de biomedicina (23) utilizando a biblioteca Eigen (24). Mais detalhes sobre métodos iterativos podem ser encontrados no livro de Trefethen e Bau III (19) e sobre o método LSQR no artigo de Saunders (25).

Uma comparação de performance[1] entre os três métodos ao aumentarmos a resolução da malha pode ser vista no gráfico da figura 4.28. Comparamos apenas o tempo de convergência do método (medido em minutos). Discussões sobre a qualidade da reconstrução podem ser vistas no próximo capítulo.

[1]Teste realizado em um iMac com processador Intel Core i5 de 2.9GHz e 8GB de RAM.

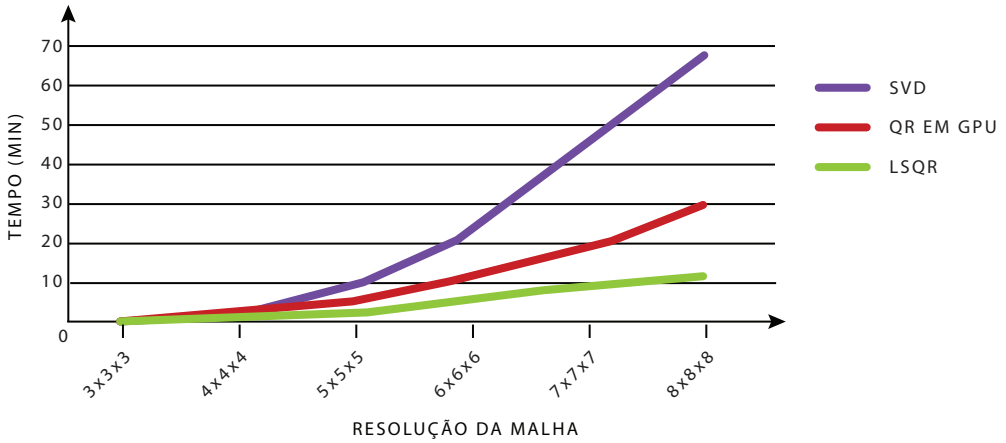


Figura 4.28: Comparação do tempo de execução dos métodos SVD, QR em GPU e LSQR.