

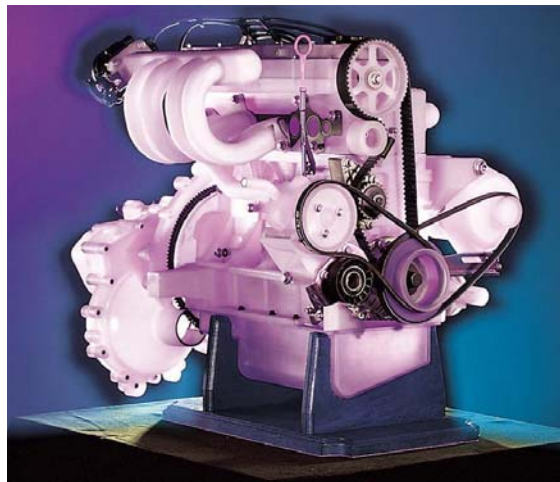


Figura 5.11 - Típico motor de combustão interna criado em uma impressora 3D – (http://www.stereolithography.com/stereolithography_images.php).

Para representação de um desenho a ser impresso em uma impressora 3D o tipo de arquivo utilizado é **.stl* (*Standard Tessellation Language*). Um arquivo **.stl* é uma representação triangular da superfície de uma geometria 3D. Este tipo de arquivo representa um objeto tridimensional criando uma malha de elementos triangulares que envolvem toda a superfície do mesmo, como pode ser visto na Figura 5.12

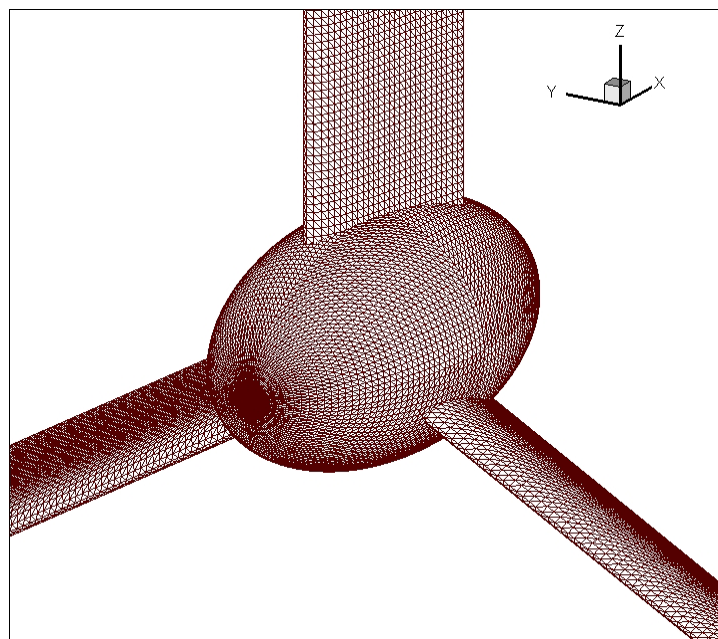


Figura 5.12 - Trecho de uma turbina eólica representada por um arquivo **.stl*.

Para cada triângulo, chamado de *face* no arquivo **.stl*, são dadas informações como a posição dos vértices e a direção do vetor normal à esta face. A Figura 5.13 mostra um pequeno trecho de um arquivo **.stl*.

```
facet normal n1 n2 n3
  outer loop
    vertex v11 v12 v13
    vertex v21 v22 v23
    vertex v31 v32 v33
  endloop
endfacet
```

Figura 5.13 - Trecho de um arquivo **.stl* . ([http://en.wikipedia.org/wiki/STL_\(file_format\)](http://en.wikipedia.org/wiki/STL_(file_format))).

Vale lembrar que em um arquivo **.stl* o trecho de texto mostrado na Figura 5.13, pode ser repetido milhares de vezes. Isto dependerá do refinamento da malha de elementos triangulares que representa uma geometria.

O fato de uma geometria qualquer poder ser representada por uma malha de elementos triangulares em um arquivo **.stl*, juntamente com o fato do código Fluids-3D representar a fronteira imersa a partir de uma malha de elementos triangulares, faz com que o uso de um arquivo **.stl* para a representação de uma geometria complexa no Fluids-3D seja totalmente adequado.

Uma outra grande vantagem do uso de arquivos **.stl*, é que este não necessita de ferramentas proprietárias para sua criação. Diversas ferramentas como o *Gmsh*, *Admesh* e *Meshlab*, por exemplo, são *softwares* livres e, portanto seu uso é gratuito (OLIVEIRA, 2007).

Com o intuito de demonstrar a capacidade do código Fluids-3D em lidar com escoamentos sobre quaisquer geometrias imersas, algumas geometrias, como a da Figura 5.14, criada com o software *Gmsh*, foram realizadas.

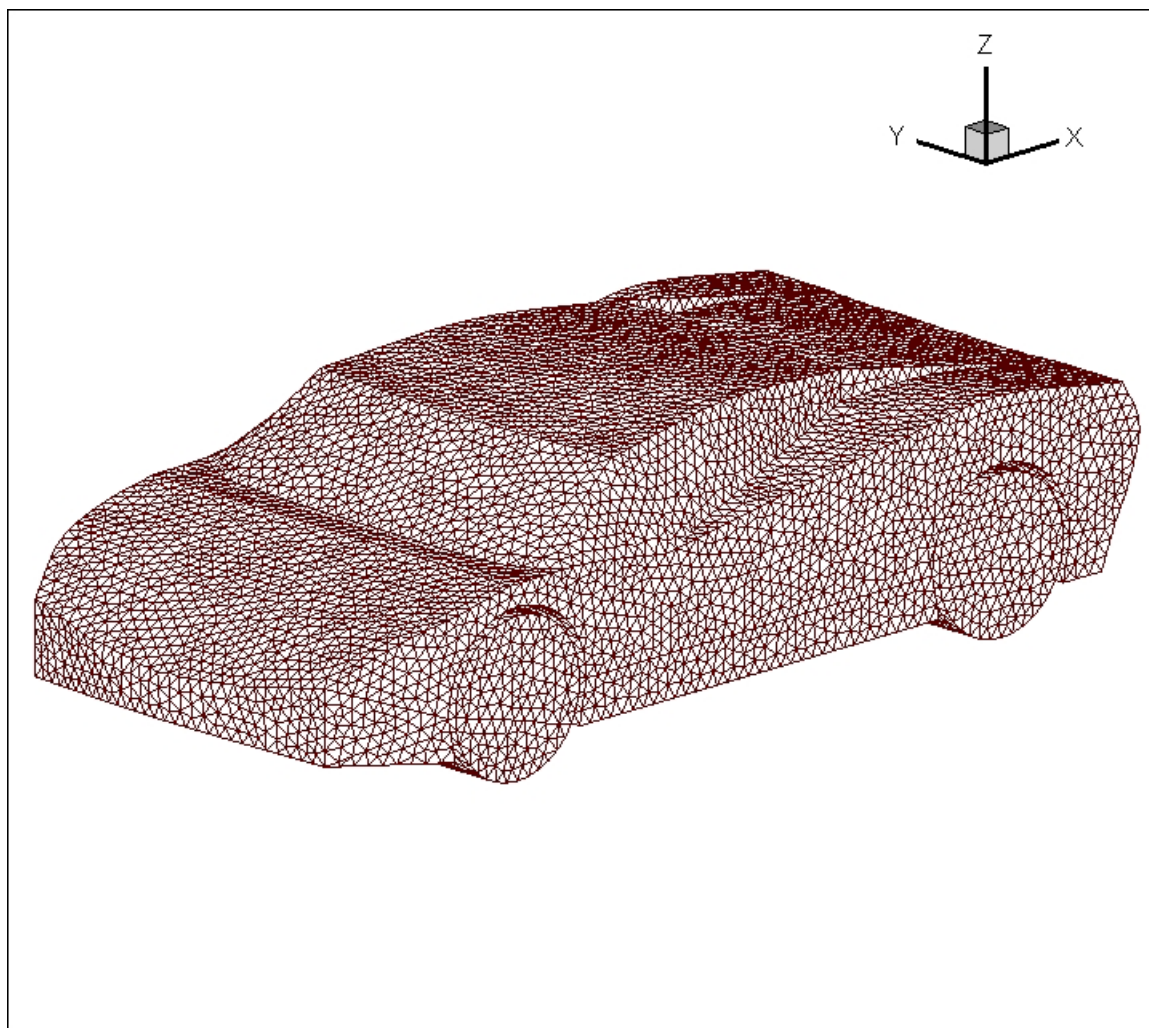


Figura 5.14 - Malha *.stl, gerada sobre um protótipo de automóvel (OLIVEIRA, 2007).

Os resultados de simulações de escoamentos sobre geometrias como a da Figura 5.14 serão apresentados e comentados no capítulo de resultados.

Capítulo VI

6 Metodologias de paralelização utilizadas no Fluids-3D

Como já mencionado em capítulos anteriores o uso de processamento paralelo em mecânica dos fluidos computacional é inevitável. Portanto este é um tópico que merece atenção especial. No presente trabalho a topologia e paralelização do código Fluids-3D foi alterada profundamente, visando tanto uma maior autonomia quanto aos domínios computacionais a serem simulados, e visando conseqüentemente, uma diminuição no tempo de processamento de uma simulação.

6.1 Tipos de paralelização possíveis no código Fluids-3D

Na paralelização escolhida no código Fluids-3D, cada processador executa basicamente o mesmo código, porém sobre uma base de dados diferente. Isto exige que se faça uma comunicação entre os processadores para que as condições de contorno sejam satisfeitas. Neste tipo de paralelização escolhido, o domínio original é subdividido em subdomínios menores, de forma a se particionar a carga de trabalho do modo mais igualitário possível entre os processadores, otimizando o uso do cluster e conseqüentemente melhorando o *speed-up*.

A topologia apresentada pela interface de comunicação entre os subdomínios pode variar de unidimensional a tridimensional, conforme mostrada da esquerda para a direita na Figura 6.1. Na topologia unidimensional, a troca de mensagens é realizada em uma única direção, que é estendida a duas direções na topologia bidimensional e três para o caso tridimensional, sempre relacionado a um volume localizado na interface. A título de exemplo, tome-se o estêncil gerado na interface de troca de mensagens entre os subdomínios: dividindo-os empregando-se a topologia unidimensional, implicaria no intercâmbio de informações ao longo de um único eixo coordenado. No caso bidimensional, os dois eixos do estêncil trocariam mensagens e, por fim, no caso tridimensional, os três eixos sofreriam intercâmbio de dados. A relação de diminuição da área superficial referente à escolha da topologia está apresentada na Figura 6.2. Cabe salientar que o volume computacional no interior de cada subdomínio permanece como sendo $V_i = i^3 / N$ independentemente do tipo de topologia escolhida, Campregher (2005).

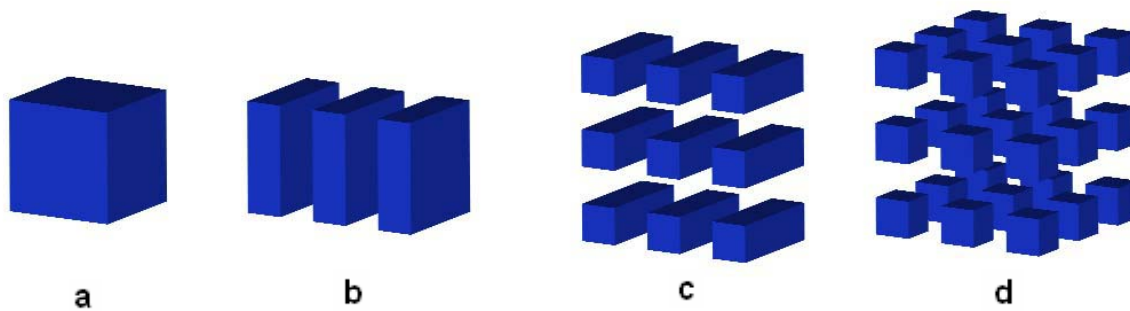


Figura 6.1 - Topologias possíveis para a troca de mensagens. a) domínio original, b) paralelização unidimensional, c) bidimensional e d) tridimensional.

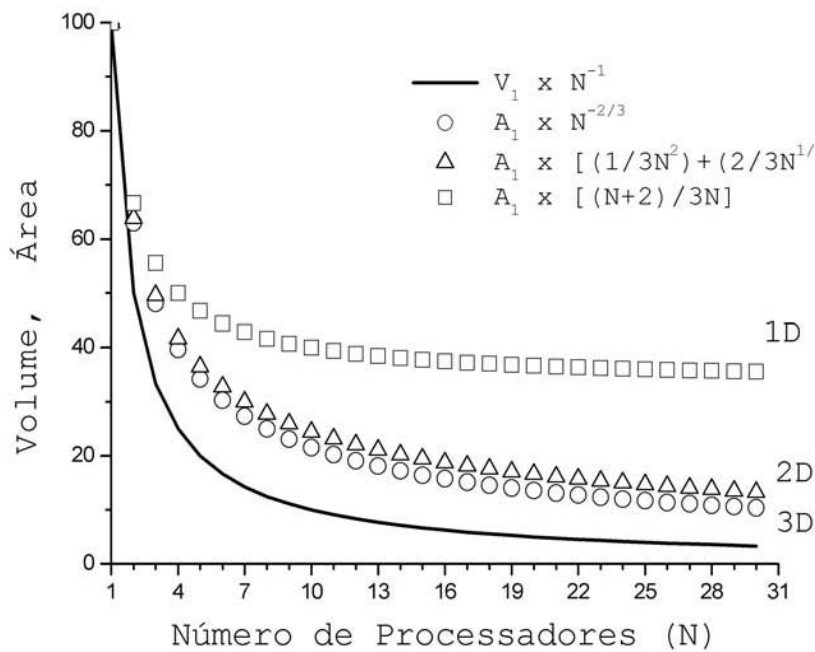


Figura 6.2 - Relação entre a diminuição do volume computacional e da área de troca de mensagens para as três topologias possíveis de particionamento (CAMPREGHER, 2005).

É possível observar, com relação à Figura 6.2, que o particionamento unidimensional atinge a sua capacidade máxima de diminuição da área superficial de troca de mensagens entre os subdomínios adjacentes muito antes das demais topologias. Evidentemente, esta característica, num problema de grande porte, implica num desempenho inferior do cluster. A

primeira versão do código Fluids-3D foi feita a escolha pelo particionamento unidimensional do domínio. Segundo Campregher (2005), esta escolha não comprometeu a performance, do código, principalmente por se empregarem poucos processadores nos cálculos. A vantagem mais clara de se empregar particionamento unidimensional reside na implementação bem mais simplificada do código.

A discretização dos termos espaciais no código Fluids-3D utiliza aproximações de segunda ordem centradas, o que exige um estêncil de três células, sendo uma o ponto de interesse e as outras duas a esquerda e a direita do ponto de interesse. Desta maneira, em uma topologia de paralelização unidimensional, faz-se necessária a sobreposição (*overlapping*) de apenas um plano. Por outro lado, aproximações de ordem mais elevada, exigiriam um número maior de planos e, conseqüentemente, um fluxo maior de troca de mensagens e possível sobrecarga (*overhead*) da rede. Este fato é particularmente complicante na topologia tridimensional de paralelização.

Na Figura 6.3, pode-se observar que os volumes internos ao domínio, como os volumes 2 e $n-1$ que foram calculados e estão atualizados, são copiados aos volumes n e 1, respectivamente, agindo como condição de contorno aos domínios Ω_1 e Ω_2 . Na Figura 6.4 é possível observar, conjuntamente, as sobreposições entre os domínios e a topologia unidimensional empregada

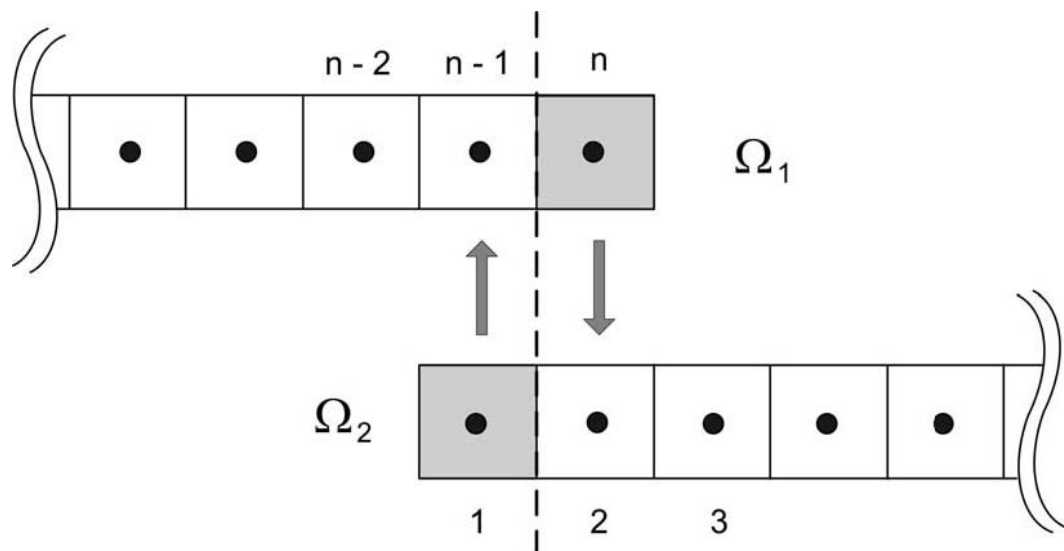


Figura 6.3 - Sobreposição necessária para aproximações espaciais de segunda ordem (CAMPREGHER, 2005).

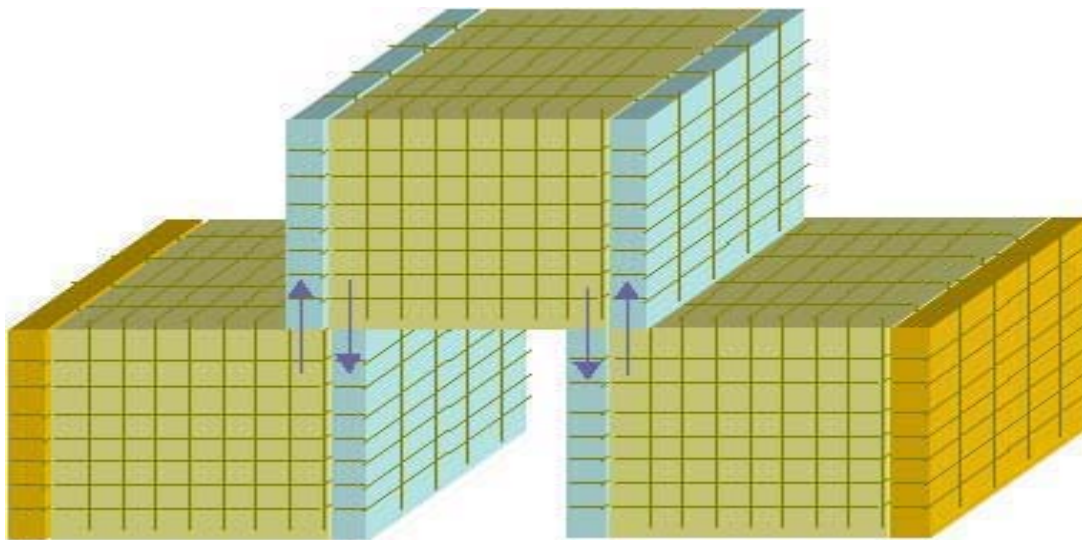


Figura 6.4 - Comunicação entre os subdomínios usando uma topologia unidimensional de divisão (MARINHO, 2004).

6.2 Implementação da topologia tridimensional de paralelização no código Fluids-3D

Existem basicamente duas maneiras de se trabalhar com o armazenamento de dados em um código numérico. A primeira, e mais comum, é trabalhar com vetores multidimensionais, como, por exemplo, uma matriz, designando um campo bidimensional de variáveis como velocidades ou qualquer outra variável de trabalho. Em um caso tridimensional, as declarações das variáveis são semelhantes a um caso bidimensional, porém, a matriz a ser alocada possui uma dimensão a mais. A Figura 6.5 mostra, como seria o formato de uma matriz tridimensional.

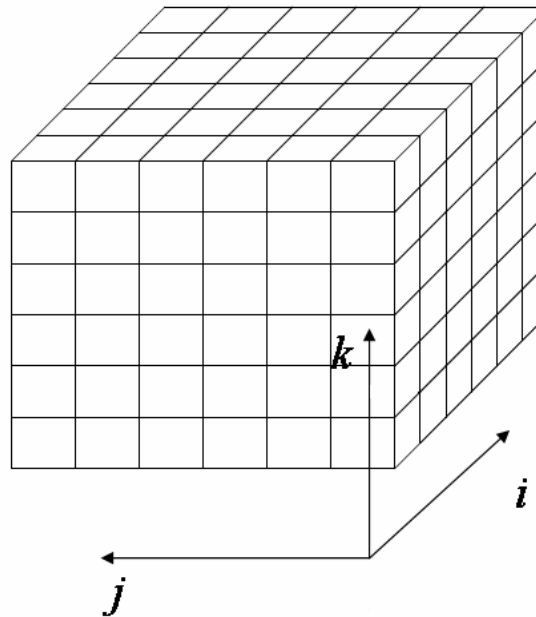


Figura 6.5 - Representação física de um domínio tridimensional.

Uma outra maneira de manipular estes dados é, ao invés de criar matrizes tridimensionais, criar vetores (*arrays* unidimensionais), porém, obviamente com o mesmo número de volumes de uma matriz tridimensional, para o armazenamento e seguindo uma dada lei de formação. No caso do atual trabalho esta lei de formação é dada por:

$$l = (i - 1) \cdot y_{tot} \cdot z_{tot} + (j - 1) \cdot z_{tot} + k, \quad (6.1)$$

onde, l é a posição no vetor, y_{tot} e z_{tot} , são quantidades máximas de volumes nas direções y e z respectivamente, e i , j e k , são contadores que variam de 1 à x_{tot} , y_{tot} e z_{tot} respectivamente, sendo x_{tot} quantidade máxima de volumes na direção x .

Nota-se que, em um armazenamento com o uso de matrizes, uma variável possui três coordenadas para localização, por exemplo, se a variável em questão estiver posicionada onde as dimensões em x , y e z sejam máximas, seu valor é dado por:

$$\phi(x_{tot}, y_{tot}, z_{tot}) = aux, \quad (6.2)$$

onde aux é um valor numérico qualquer.

No caso de um código vetorizado (onde os dados são armazenados em vetores, ou seja, a segunda opção), a mesma variável citada no parágrafo acima seria encontrada por:

$$\phi(l) = aux, \quad (6.3)$$

onde l , aplicando a eq (6.1), é dado por $xtot * ytot * ztot$.

A vantagem de se usar um código vetorizado, é que, uma vez necessário usar o valor armazenado em qualquer local de um vetor, o acesso a tal dado é mais rápido, uma vez que o processador não necessita realizar nenhum cálculo extra para encontrar tal variável. Este cálculo extra é explicado por que atualmente, mesmo declarando variáveis como matrizes, automaticamente, o processador já transforma estas matrizes em vetores. O fato de usar um código vetorizado elimina a necessidade de o processador ficar vetorizando matrizes. Isto é particularmente benéfico em grandes volumes de dados. O código Fluids-3D utiliza desta metodologia. Vale a pena salientar que o termo código vetorizado, utilizado no presente texto diz respeito ao tipo de alocação de dados realizados no código Fluids-3D, não tendo nenhuma relação com os tipos de vetorizações realizados por supercomputadores tipo Crays ou IBM's.

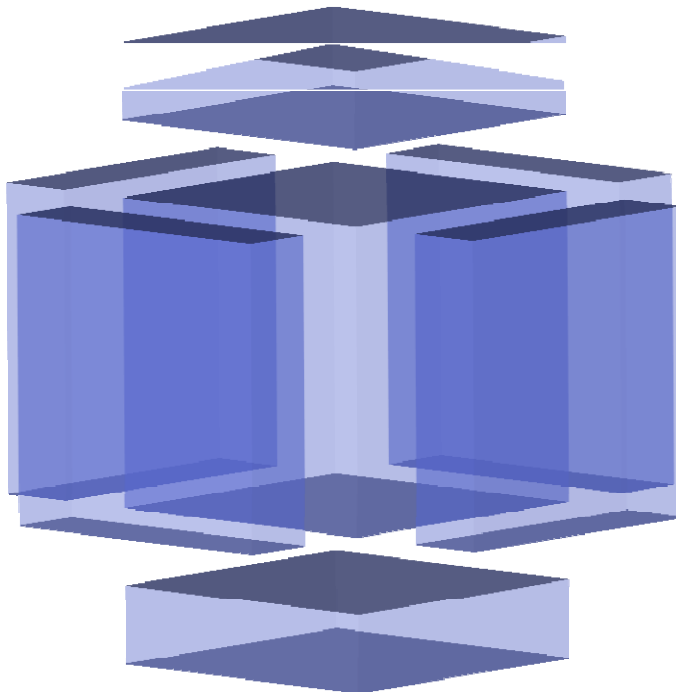


Figura 6.6 - Áreas de trocas de informações em uma topologia tridimensional de paralelização.

Apesar de ser comparativamente mais rápido do que um código matricial, a paralelização tridimensional de um código vetorizado não é algo trivial. Como já citado, a troca de informações é realizada nas fronteiras de cada subdomínio, no caso do particionamento unidimensional, a troca é mostrada na Figura 6.4. Já a Figura 6.6 mostra um diagrama de como seria a troca de informações em um particionamento tridimensional. No caso de um código matricial, a troca de informações seria basicamente igual em todas as direções e cuidados teriam que ser tomados principalmente com as condições de contorno. Por exemplo, onde, antes era uma parede com condição de não escorregamento, agora pode vir a ser um outro processador.

Devido a lei de formação mostrada na eq (6.1), no caso de um código vetorizado, somente se o particionamento for feito unidimensionalmente e na direção x, a troca de informação ocorre sem o risco de overhead, isto porque como o vetor que corresponde ao domínio computacional é criado a partir de *loops* na três direções coordenadas, sendo que este *loop* faz com que o vetor seja gerado primeiro na direção z, depois em y e depois em x, portanto cada valor de x corresponde a uma face na direção x, cada valor de y, corresponde à uma linha na direção y e conseqüentemente cada valor de z corresponde somente à um único valor de z. A Figura 6.7 dá uma idéia de como é preenchido um vetor qualquer ϕ . A Figura 6.8 mostra graficamente a área de troca em cada direção coordenada.

```

do i = 1,xtot
  do j = 1,ytot
    do k = 1,ztot
      l=(i-1)·ytot·ztot+(j-1)·ztot+k
      phi(l)=...
    enddo
  enddo
enddo

```

Figura 6.7 - Pseudo-código indicando a formação de um vetor ϕ .

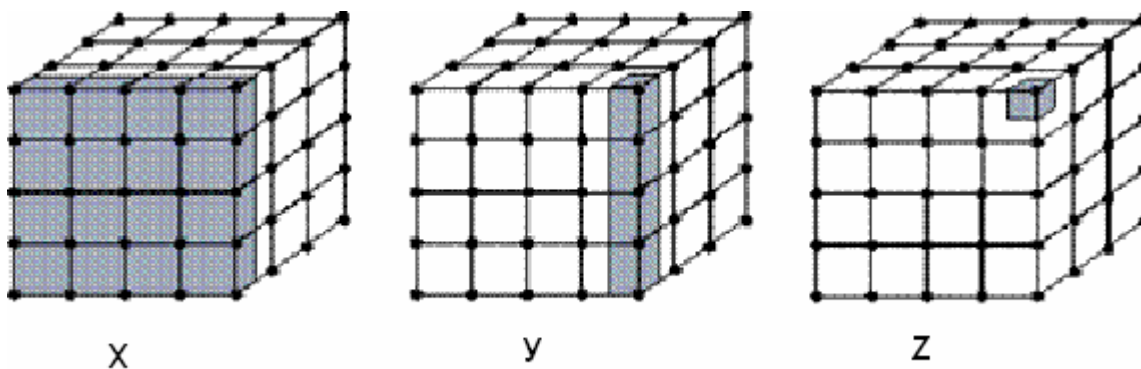


Figura 6.8 - Área de troca de informação nas três direções coordenadas.

O fato de as rotinas de paralelização serem mais eficientes quando mandam maiores volumes de dados inviabiliza a paralelização em y e z, uma vez que o overhead é inevitável. Para ainda sim paralelizar tridimensionalmente um domínio, algumas precauções podem ser tomadas, evitando o *overhead*. Neste trabalho optou-se por criar vetores auxiliares com o tamanho característico da mensagem que se deseja enviar, e então enviá-los utilizando comandos adequados das bibliotecas de paralelização, e no processador de destino “abrir” este conjunto de dados e redirecioná-los às suas posições corretas.

Esta ação de criar vetores auxiliares, apesar de ter um certo custo computacional, foi uma maneira simples de se evitar o overhead na rede, e, assim, paralelizar tridimensionalmente o código Fluids-3D

Duas considerações importantes ainda são necessárias:

- Toda a paralelização citada nesta dissertação diz respeito somente a paralelização do domínio euleriano, e não da fronteira imersa e do modelo físico virtual. No caso da simulação de geometrias complexas a fronteira imersa ainda deve estar confinada em um único processador, porém, com a utilização de topologia tridimensional, uma nova gama de geometrias podem ser simuladas, uma vez que esta metodologia de paralelização, aumenta muito a escalabilidade do código, que por sua vez pode simular domínios maiores.
- Em casos nos quais a razão de aspecto é baixa ou moderada, ou escoamentos que não necessitam de vários milhões de volumes para ficarem bem caracterizadas, a paralelização tridimensional, ou mesmo bidimensional não são indicados. Como quem dita a velocidade da simulação é sempre o processador mais lento (mais requisitado, ou com uma quantidade maior de volumes), o processador que é responsável pelos cálculos lagrangianos é

sempre quem dita a velocidade da simulação. Algumas simulações monitoradas indicam que o processador da fronteira imersa chega a 99% de uso, enquanto os processadores responsáveis pelas outras áreas do domínio não chega, a usar 7% de CPU.

Em todas as simulações realizadas no presente trabalho a razão de aspecto dos domínios eram baixas ou moderadas, assim sendo, não foi necessário utilizar o particionamento bi ou tridimensional. No entanto, o trabalho de paralelização 3D foi de suma importância, uma vez que este proporciona uma maior liberdade de particionamento do domínio euleriano, mesmo que seja em apenas uma direção. Isto é particularmente útil para se trabalhar com malhas refinadas a jusante do objeto imerso.

6.3 Bibliotecas de paralelização.

A comunicação entre os domínios, juntamente com a quantidade de cálculos processados em cada um deles, requer um cuidado especial, pois afeta não somente o desempenho da simulação, mas, também, a integridade dos resultados. Infelizmente, ainda não existe um software que seja capaz de paralelizar automaticamente, de forma eficiente e com segurança, todas as situações possíveis encontradas em processamento paralelo. Programas de maior desempenho ainda necessitam da intervenção do programador para se chegar ao resultado desejado, Campregher (2005).

A responsabilidade de transferência de informações entre os subdomínios em uma simulação fica a cargo das bibliotecas de paralelização, difundidas amplamente na Internet. Entre as mais populares, a PVM (*Parallel Virtual Machine*) e a MPI (*Message Passing Interface*) são as mais utilizadas, sendo esta última a empregada no presente trabalho.

A biblioteca PVM foi desenvolvida em 1989 por um grupo de universidades e centros de pesquisa nos EUA em um projeto chamado *Heterogeneous Network Project*. Como proposta inicial, o objetivo dos pesquisadores era desenvolver uma biblioteca que permitisse a comunicação entre máquinas de natureza heterogênea (diversos tipos de máquinas e processadores) sob o conceito de troca de mensagens (PITANGA, 2002). De forma geral, o sistema PVM consiste de uma aplicação que, uma vez inicializada, é responsável por comunicar-se com as demais máquinas da rede, formando a rede virtual e de um conjunto de bibliotecas próprias (em torno de 38) para realizar a criação e destruição de processos, troca de mensagens e sincronização de tarefas.

Pode-se dizer que a principal característica da PVM é poder construir uma rede heterogênea com eficiência, sendo controlada por uma única máquina virtual. Além disso,

permite criar e destruir processos em tempo de execução, sendo possível remover ou adicionar máquinas durante a simulação. Finalmente, a PVM possui notável tolerância à falhas, pois caso a máquina virtual não encontre comunicação com um nó da rede, ela automaticamente redireciona as tarefas para um outro nó, sem interromper o processo. Entretanto, toda esta facilidade na construção da rede virtual implica num custo alto em termos de sobrecarga de comunicação (overhead), prejudicando o desempenho do código o que tende a piorar, consideravelmente, em sistemas com grande número de processadores.

Na primeira versão do código Fluids-3D foi construída a implementação MPICH (MPICHamaleon) do padrão MPI-2. Nos desenvolvimentos do atual trabalho, a biblioteca escolhida foi a MPICH-2. Desenvolvida a partir de melhoramentos necessários nas primeiras versões do MPI, esta biblioteca chega a ser 30 % mais rápida durante o processo de troca de mensagens, se comparada a versões antigas do MPICH, Gropp et al. (1999). Além de ser considerada uma das únicas versões que combinam portabilidade, interoperabilidade e alta performance, possui ainda capacidade de lidar com máquinas heterogêneas e SMP (*Symmetric Multi Processors*), ou seja, máquinas individuais com dois ou mais processadores. Outras vantagens importantes são a disponibilidade de diversos recursos, que podem ser explorados visando melhorar o rendimento dos códigos e a sua excelente portabilidade. Finalmente, pode ser dito que o MPI apresenta-se como um padrão para processamento paralelo, o que garante constante manutenção e atualização, Gropp et al. (1999).

CAPÍTULO VII

7 Resultados e Discussão

Como já mencionado em capítulos anteriores, as mudanças realizadas no pré-processamento do código Fluids-3D viabilizaram uma maior facilidade de *set-up* e importação de geometrias, facilitando assim, o estudo de escoamentos sobre diversas geometrias. Neste capítulo serão apresentados resultados de re-validação deste código, bem como resultados de aplicações de CFD a escoamentos sobre geometrias complexas.

Existem três possibilidades de condições de contorno nas direções y e z do domínio euleriano, são elas:

- parede: todas as componentes das velocidades (normais e tangenciais) possuem valor nulo;
- fronteira livre: derivadas nulas para os valores das velocidades tangenciais e velocidade nula na direção normal;
- periodicidade: os fluxos de massa e de quantidade de movimento linear que saem de uma superfície, entram pela superfície oposta.

Na entrada do domínio, adota-se um perfil uniforme de velocidade com $u=U_{\infty}$, $v=w=0$. Com relação às condições de contorno na saída do domínio utiliza-se um artifício de correção da componente u no último plano do domínio (em $i=xtot$, onde i representa a posição de um dado plano na direção x), de modo a garantir a conservação de massa global. Este artifício consiste em atribuir um valor equivalente ao valor do penúltimo plano (em $i=xtot-1$) multiplicado por uma razão entre as massas totais que entram e que saem do domínio (VERSTEEG; MALALASEKERA, 1995, CAMPREGHER, 2005).

7.1 Resultados de re-validação

Esta seção é dividida em duas partes. Primeiramente serão apresentados resultados para escoamentos sobre uma esfera estática, com o intuito de validar a sub-rotina de importação de geometrias geradas na forma de arquivos *.stl. O processo de geração de um arquivo *.stl pode ser encontrado mais detalhadamente em Oliveira (2007). Com o intuito de re-

validar a importação de geometrias provenientes da importação de malhas geradas em geradores de malhas comerciais, o escoamento sobre um cubo estático será apresentado.

7.1.1 Resultados de validação da importação de geometrias via *.stl

A validação da sub-rotina de leitura de arquivos *.stl foi realizada comparando o escoamento sobre uma esfera, utilizando a primeira versão do código Fluids-3D, que foi amplamente validada em Campregher (2005), e a versão atual do código Fluids-3D. A Figura 7.1 mostra um exemplo de malha de uma esfera representada por uma arquivo *.stl.

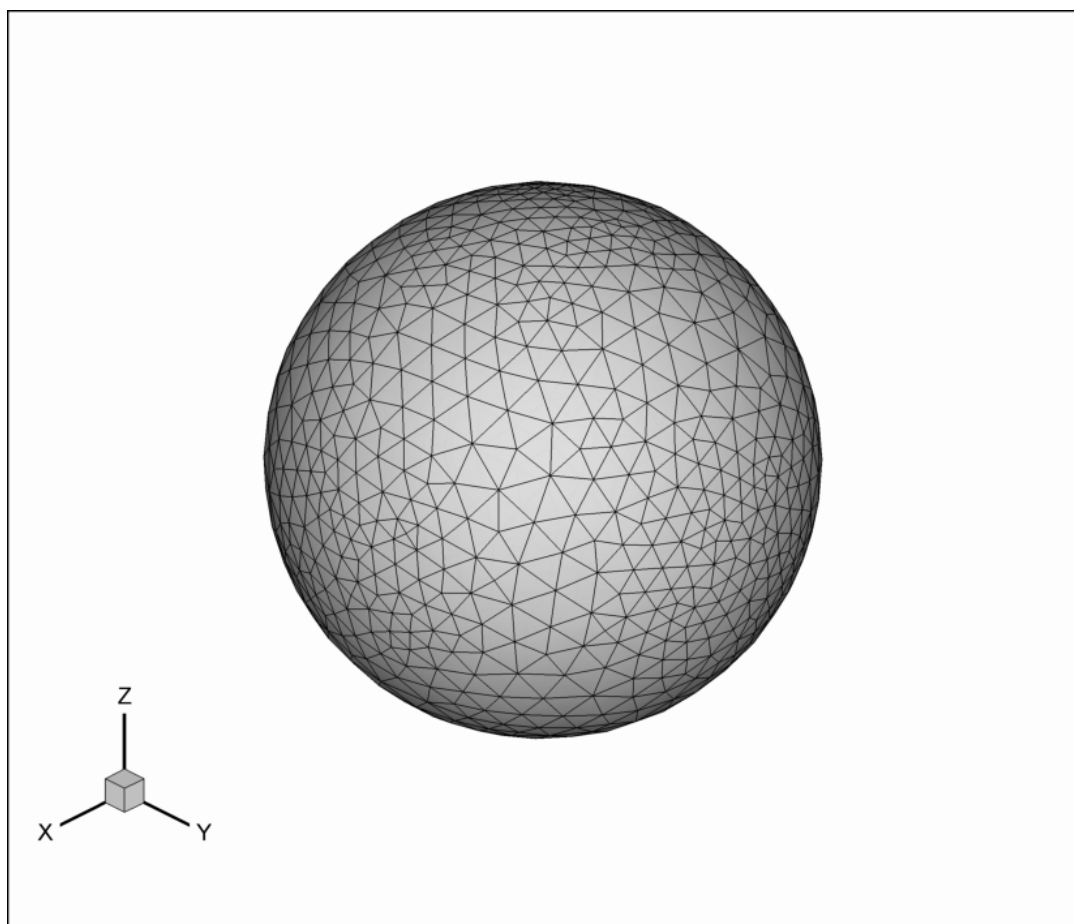


Figura 7.1 - Visualização de uma malha de elementos triangulares provenientes de um arquivo *.stl.

A malha euleriana utilizada em ambos os casos é mostrada na Figura 7.2. Tal malha tem dimensões $1,0 \times 0,68 \times 0,68 m$. A esfera está posicionada com seu centro na posição (x, y,

$z) = (0,38, 0,34, 0,34)$ m. O domínio euleriano foi discretizado por volumes finitos, como mencionado anteriormente. Com base no diâmetro D da esfera, o domínio possui dimensões de $9D$ à montante, $15D$ à jusante e largura de $17D$ em ambas as direções y e z . A razão de bloqueio, em termos de área foi de $0,27\%$, Campregher (2005). Para estas simulações foram utilizados três processadores em paralelo, como mostrado na Figura 7.3.

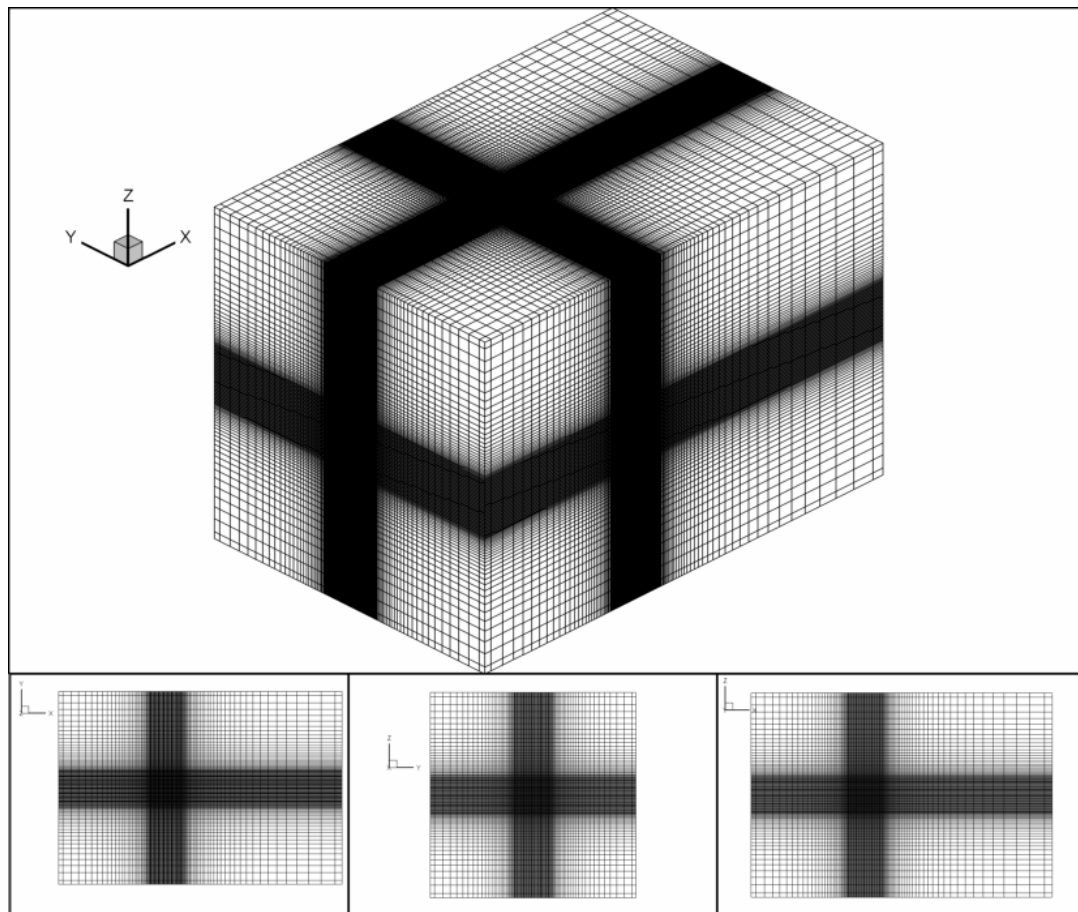


Figura 7.2 - Malha euleriana utilizada nas simulações dos escoamentos sobre esferas.

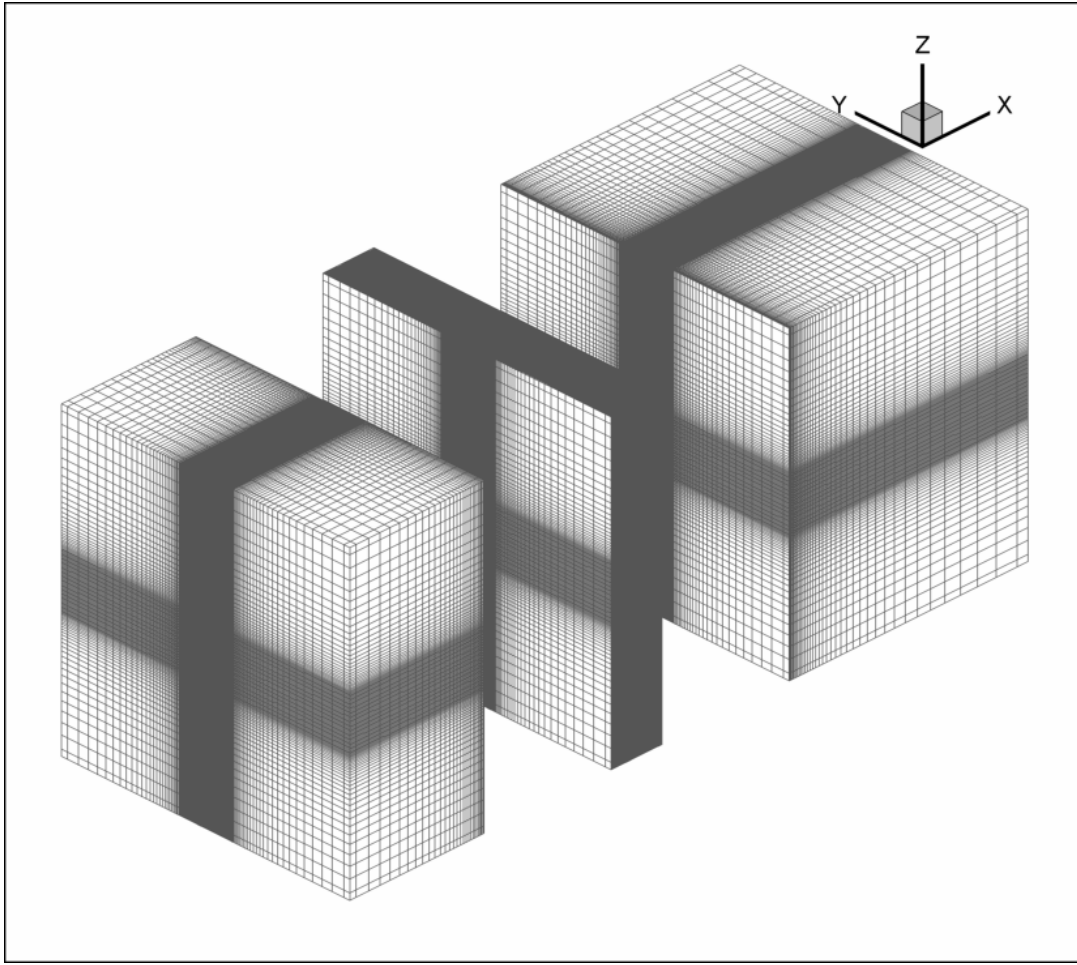


Figura 7.3 - Particionamento do domínio euleriano para escoamentos sobre esferas.

Para um número de Reynolds igual a 1000, são comparadas tanto propriedades qualitativas, como o critério Q (JEONG; HUSSAIN, 1995), quanto valores quantitativos, como valores dos coeficientes de arrasto, lateral e de sustentação.

O critério Q é definido na como:

$$Q = \frac{1}{2} \left[|\Omega|^2 - |S|^2 \right] > 0, \quad (7.1)$$

ou seja, a norma Euclidiana para a qual o tensor vorticidade sobrepuja o tensor deformação, definidos respectivamente como (Campregher, 2005):

$$\bar{\Omega} = \frac{1}{2} \left[\bar{\nabla} \vec{v} - (\bar{\nabla} \vec{v})^T \right], \quad (7.2)$$

$$\bar{\bar{S}} = \frac{1}{2} \left[\vec{\nabla} \vec{V} + (\vec{\nabla} \vec{V})^T \right]. \quad (7.3)$$

Os coeficientes de arrasto, lateral e de sustentação e o número de Strouhal, são definidos respectivamente como:

$$C_D = \frac{F_x}{\frac{1}{2} \rho U_\infty^2 A}, \quad (7.4)$$

$$C_s = \frac{F_y}{\frac{1}{2} \rho U_\infty^2 A}, \quad (7.5)$$

$$C_L = \frac{F_z}{\frac{1}{2} \rho U_\infty^2 A}, \quad (7.6)$$

$$St = \frac{f L}{U_\infty}, \quad (7.7)$$

onde F_x , F_y e F_z , são respectivamente a somatória das forças de cada ponto lagrangiano nas direções x, y e z. No caso da esfera, A é a projeção do objeto imerso na direção transversal ao escoamento ($A = \pi D^2/4$) e D é o diâmetro da esfera, que em todas as simulações foi mantido em 0,04m, U_∞ é a velocidade de entrada do fluido no início do domínio ($x=0$), e L é o comprimento característico do objeto imerso.

A Figura 7.4 abaixo mostra estruturas turbilhonares através da visualização de isosuperfícies $Q = 10$, para o escoamento gerado a partir de uma geometria gerada por um gerador de malhas comercial, para tempos físicos de 5,0 s, 5,5 s, 6,0 s, 6,5 s e 7,0 segundos. Em todas as simulações sobre esferas as condições de contorno do domínio euleriano foram impostas como superfícies livres.

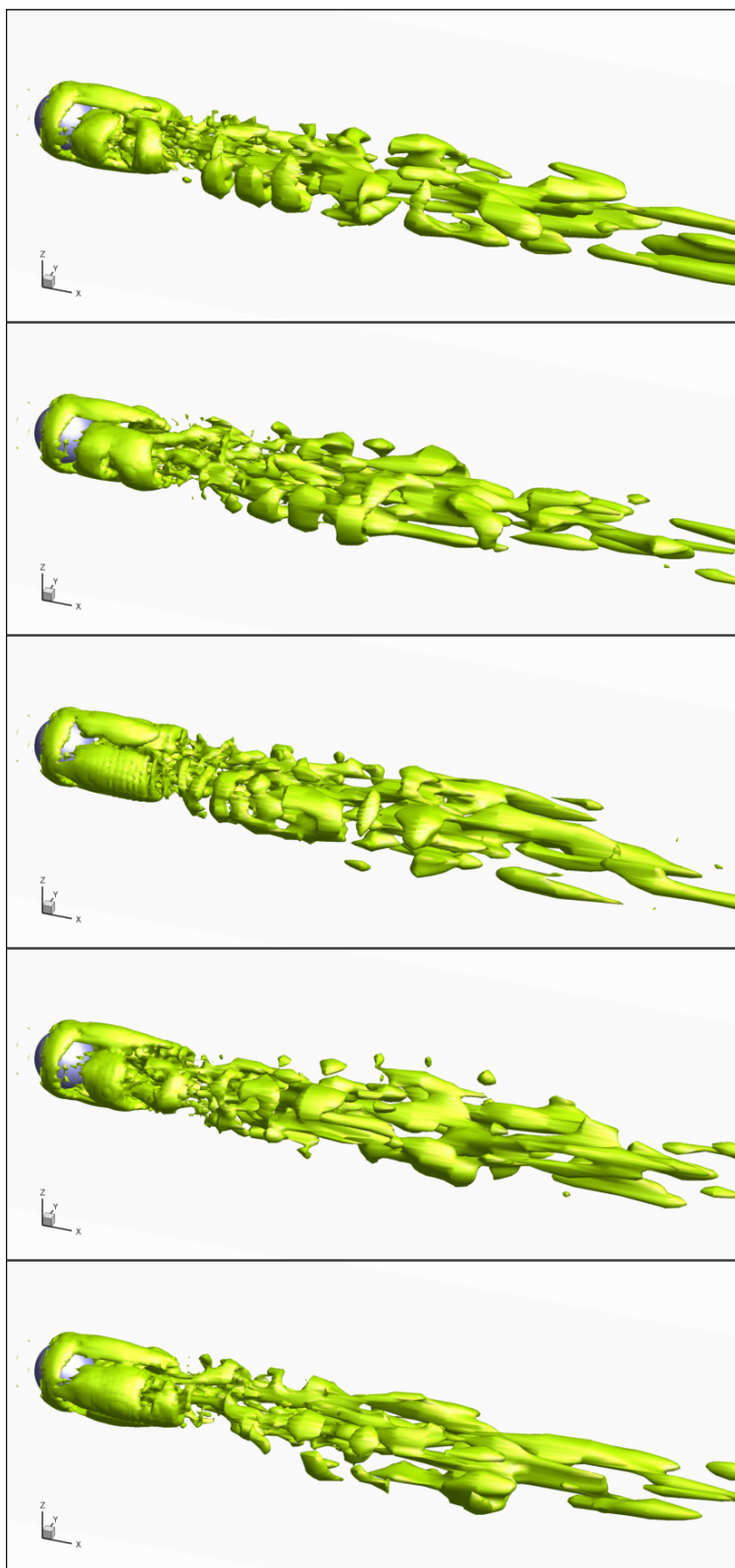


Figura 7.4 - Estruturas turbilhonares a jusante de uma esfera, importada de geradores de malhas comerciais ($Re=1.000$).