GUSTAVO PRADO OLIVEIRA

DESENVOLVIMENTO DO WEB SITE CFD-IB PARA O SETUP FÍSICO DO SOLVER FLUIDS 3D DE DINÂMICA DOS FLUIDOS COMPUTACIONAL

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA MECANICA
2007

GUSTAVO PRADO OLIVEIRA

DESENVOLVIMENTO DE UMA INTERFACE GRÁFICA PARA O SOLVER FLUIDS 3D DE DINÂMICA DOS FLUIDOS COMPUTACIONAL

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Mecânica da Universidade Federal de Uberlândia, como parte dos requisitos para a obtenção do título de **MESTRE EM ENGENHARIA MECÂNICA**.

Área de Concentração: Mecânica dos Fluidos

Orientador: Prof. Dr. Aristeu da SilveTc0 2eto

Dados Internacionais de Catalogação na Publicação (CIP)

O48d Oliveira, Gustavo Prado, 1978-
Desenvolvimento de uma interface gráfica para o solver fluids 3D de dinâmica dos fluidos computacional / Gustavo Prado Oliveira. - 2007.

153 f. : il.

rids 3-7.5iis de CPublie Clâm40e1(assifa Publional)Tj/TT6 1 Tf12 0425.TD5.022679.9403 Tr

GUSTAVO PRADO OLIVEIRA

DESENVOLVIMENTO DE UMA INTERFACE GRÁFICA PARA O SOLVER FLUIDS 3D DE DINÂMICA DOS FLUIDOS COMPUTACIONAL

Dissertação **APROVADA** pelo Programa de Pós-graduação em Engenharia Mecânica da Universidade Federal de Uberlândia.

Área de Concentração: Mecânica dos Fluidos

Banca Examinadora:

_____

Prof. Dr. Aristeu da Silveira Neto – UFU – Orientador

_____

Msc. Clovis Raimundo Maliska Júnior ESSS – SC

_____

Prof. Dr. Solidônio Rodrigues Carvalho – UFU

_____

Prof. Dr. Donald Mark Santee – UFG

Uberlândia, 23 de Março de 2007

Agradecimentos

Ao meu orientador e amigo Aristeu, por ser sempre um exemplo de ser humano, que me ajudou e confiou no meu empenho.

Aos meus amigos, Felipe, Sigeo, Leonardo, Tiago, João Marcelo, Millena, Priscila, Professor Gilmar e a todos os outros membros do LTCM, que de uma maneira ou outra foram indispensáveis para a realização deste trabalho.

Aos meus amigos Rubens Campregher e José dos Reis Vieira de Moura Jr, Francisco Aurilo, pelas inestimáveis idéias e ajudas quando os problemas pareciam não ter mais solução.

Ao meu amigo Professor Ricardo Fortes, que me incentivou a continuar minha jornada no mestrado, me mostrando que o importante é a satisfação pessoal

Ao meu padrasto Cizinho e minha mãe Marilene, por me oferecerem todo amparo afetivo me incentivando a nunca desistir por mais complicado que sejam os problemas. Obrigado também pelo exemplo de honestidade, trabalho, integridade que são.

Ao meu pai Allan Kardec e sua esposa Tânia, que com toda humildade compartilharam do meu crescimento como pessoa.

Aos meus irmãos Marcos Ruitter, Pablo, Guilherme e Otávio, por serem exemplos de seres humanos e amizades.

À minha irmã Fabiana, que sempre foi e sempre será uma das pessoas mais importantes da minha vida.

À minha noiva Débora, que me acompanhou durante todo meu crescimento profissional, mesmo com todas adversidades, com enorme companheirismo. Obrigado por sua paciência

e por seu amor.

Ao meu amigo Alexandre Santana (*in memorian*), que me incentivou a dar continuidade em meus estudos, tanto no mestrado quanto no doutorado, e que sem ele essa dissertação não seria possível.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) por financiar meus estudos junto ao Programa de Pós-Graduação em Engenharia Mecânica (POSMEC-UFU) pelo suporte necessário para desenvolver meu projeto.

E por último, porém não menos importante, a Deus que me abençoa a cada dia com a graça de viver, sempre me dando serenidade para discernir entre o certo e o errado, e me dando forças para seguir o caminho do bem.

Oliveira, G.P., **Desenvolvimento de uma Interface Gráfica para o Solver Fluids 3d de Dinâmica dos Fluidos Computacional.** 2007. 151f. Dissertação de Mestrado, Universidade Federal de Uberlândia - Uberlândia

Resumo

O Laboratório de Transferência de Calor e Massa e Dinâmica dos Fluidos - LTMC/FEMEC/UFU tem desenvolvido diversos trabalhos para simulações e análise computacional da dinâmica dos fluidos, porém com etapas providas por softwares proprietários, tornando onerosa a utilização de tais soluções. Além disso, a ferramenta atualmente funcional existente, que efetua tais cálculos, foi desenvolvida na linguagem Fortran, desprovida de uma interface gráfica, além de não permitir uma visualização prévia das geometrias a serem trabalhadas, o que dificulta o manuseio da mesma. Dessa forma, no presente trabalho desenvolveu-se uma metodologia para a construção de geometrias e a geração de suas malhas de elementos finitos triangulares superficiais, utilizando-se de softwares livres, para posterior importação em códigos computacionais dedicados a análise computacional da dinâmica dos fluidos, através do método de fronteira imersa. Dentre as ferramentas analisadas para a modelagem da geometria e geração da malha, optou-se pela utilização do GMSH, por ser um software livre, portável para as plataformas Linux e Windows, intuitivo e de fácil aprendizado. Já dentre as ferramentas analisadas para o desenvolvimento da Interface gráfica, optou-se pelo PHP, por ser uma linguagem script de código fonte aberto (Open Source) de uso geral, muito utilizado, e especialmente guarnecida para o desenvolvimento de aplicações Web embútivel dentro de códigos fonte de linguagem HTML. Como todo código gerado em PHP é “embutido” no código HTML, o servidor WEB fica responsável por interpretar o código e transformá-lo na interface que será vistas pelos usuários. Utilizando esta linguagem, a conexão com o servidor do LTCM é feita de forma rápida e sem complicações, através da WEB. O trabalho apresentará algumas soluções de geometrias / malhas geradas no GMSH, e a preparação dessas soluções para a utilização do método de fronteira imersa na análise computacional da dinâmica dos fluidos em geometrias como esferas, cilindros, aerofólios, buscando-se utilizar os softwares para trabalhar com formas mais complexas como aeronaves e automóveis.

Palavras chave: fronteira imersa, geração de malhas, software livre.

Oliveira, G.P., **Development of a Graphical Interface for Solver of Computational Fluid Dynamics, Fluids 3d.** 2007. 151f. M. Sc. Dissertation, Universidade Federal de Uberlândia, Uberlândia.

Abstract

The Laboratory of Heat Transfer and Mass and Fluid Dynamics of the – LTMC / FEMEC / UFU has been developing several works for simulations and computational analysis of fluid dynamics, however with stages provided by software proprietors, turning onerous the use of such solutions. Besides, the functional tool existent, that it makes such simulations, it was developed in the Fortran language, without a graphic interface, don't allowing a previous visualization of the geometries, what hinders the handling of the same. In that way, in this work a methodology for the construction of geometries and the generation of their triangular meshes of superficial finite elements, being used of free software, for subsequent import in an program dedicated to the computational analysis of fluid dynamics, through the immersed boundery method. Among the tools analyzed for the modeling and generation of the geometry and their mesh, was opted for the use of GMSH, for being free software, for being an executable program in Linux and Windows platforms, intuitive and of easy learning. Among the tools analyzed for the development of the graphical interface, was opted for PHP, for being a language script of open source of general use, very used, and especially provided for the development of applications Web inside of codes source of language HTML. As every code generated in PHP is "embedded" in the code HTML, the WEB server is responsible for to interpret the code and to transform it in the interface that will be seen by the users. Using this language, the connection with the LTCM server is made in a fast way and without complications, through the WEB. The work will present some solutions of geometries / meshes generated in GMSH, and the preparation of those solutions for the use of the immersed boundary method in the computational analysis of fluid dynamics of geometries as spheres, cylinders, trying to uses software to work with more complex forms as aircrafts and automobiles.

Keywords: border immersed, generation of meshes, free software

Lista de Figuras

Nomenclatura

Letras Latinas

C_D: coeficiente de arrasto;

C_L: coeficiente de sustentação;

C_S: coeficiente lateral;

d: distância entre os centros dos volumes;

D: diâmetro da esfera;

f: vetor força por unidade de volume, freqüência;

F: vetor força no interior do sistema, força interfacial, fluxo nas faces do volume;

i: direção cart fluxor/ana;

I: tensor unitário;

p: pressão

p : pressão no ponto lagrang/ano k ;

q: termo fonte;

ϕ

Re: número de Reynolds;

S: parte viscosa do tensor tensão, área superficial do volume de controle;

t: tempo;

u: velocidade na direção x;

U velocidade *U* na corrente livre;

v: velocidade na direção y;

w: velocidade na direção z;

Letras Gregas

μ	viscosidade dinâmica;
ρ	densidade;
τ_{ij}	tensor de Reynolds;
ω	vorticidade;
Ω	volume ocupado por uma porção de massa, tensor vorticidade;
Ω_k	volume elementar em torno do ponto lagrangiano k

Operadores

∂	derivada parcial;
∇	nabla;

Índices

i,j	ponto central, componente de tensor;
∞	corrente livre ;
max	máximo;
min	mínimo;

Superíndices

*	grandezas adimensionais, estimativa de propriedade;
n	iteração;
t	tempo atual;

Siglas

CAD	Computer Aided Design
CFD	Computational Fluid Dynamics;

FDM	Finite Difference Method;
HCI	Human-Computer Interface;
IB	Immersed Boundary;
LED	Linguagem de Especificação de Diálogos;
LTCM	Laboratório de Transferência de Calor e Massa e Dinâmica dos Fluidos;
PCs	Personal Computers;
PHP	Hypertext Preprocessor;
STL	Standard Tesselation language;
UFU	Universidade Federal de Uberlândia;
UIMS	User Interface Management Systems;
VCL	Visual Component Library;
WWW	World Wide Web.

SUMÁRIO

CAPITULO 1

Introdução

A Dinâmica dos Fluidos Computacional (CFD – Computational Fluid Dynamics) é uma área de grande interesse para a solução de muitos problemas práticos aplicados à engenharia. Como exemplos, podem ser citados problemas de aerodinâmica, termodinâmica, hidráulica, dentre outros.

Com o advento dos supercomputadores tornou-se possível efetuar simulações numéricas em Dinâmica de Fluidos Computacional empregando malhas tridimensionais com níveis de refinamentos elevados. Esta linha de trabalho pode ser entendida como "virtualização de protótipo", isto é, simular o comportamento (desempenho) de geometrias antes da construção de protótipos reais, visando reduzir o custo e o tempo de desenvolvimento (ou aperfeiçoamento).

Portanto, CFD é uma ferramenta de predição do que acontecerá, quantitativamente, quando fluidos escoam, freqüentemente com as características adicionais:

- Fluxo simultâneo de calor;
- Transferência de massa;
- Mudança de fase;
- Reação química;
- Movimento mecânico;

Para os escoamentos de fluidos, modelos matemáticos são estabelecidos com base nos princípios de balanço da quantidade de movimento, da conservação da massa e da conservação da energia.

Estas equações, quando submetidas a condições de contorno e iniciais apropriadas, representam, matematicamente, um problema particular (Shames e Irving Herman, 1973). A solução analítica destas equações somente é possível para escoamentos muito simples. Para se analisar problemas reais, lança-se mão do uso dos chamados métodos numéricos.

Nos casos de escoamentos laminares, os modelos são relativamente simples, pois as equações de Navier-Stokes, conservação da massa e conservação de energia podem ser

resolvidas à contento. Contudo, a maioria dos escoamentos que acontecem na natureza e nos meios industriais são turbulentos, apresentando um alto grau de complexidade e, portanto devendo-se lançar mão de modelos matemáticos de turbulência, acrescentando termos às equações anteriormente citadas.

Os modelos de turbulência levam em conta variável estatísticas, pois escoamentos turbulentos são altamente caóticos, e com isto há a necessidade de ferramentas estatísticas para representá-los.

Assim, propostas de análises de situações reais em escoamentos de fluidos, fora do campo puramente experimental, ainda esbarram em necessidades técnicas mais sofisticadas e na disponibilidade de poderosos recursos computacionais. Evidentemente, é de igual notoriedade o crescimento dos recursos computacionais, viabilizando a realização de simulações numéricas cada vez mais acuradas de escoamentos em domínios de topologias diversas.

Problemas de elevado grau de complexidade resultam em aumento da quantidade de variáveis físicas a manipular e controlar no processo de estabelecimento do problema, estabelecimento de hipóteses para o modelo, para solução do problema.

Usualmente problemas complexos demandam forte esforço de cálculo, pois existe grande dificuldade de se estabelecer modelos matemáticos robustos ou relações entre variáveis em diversas escalas de observação. O eixo de estabelecimento do modelo de solução de problemas complexos tem sido a observação do problema, de sua fenomenologia e a concepção do modelo físico e do modelo fenomenológico que antecede o desenvolvimento do modelo matemático, desenvolvimento do sistema de equações que regem o problema, e sua solução computacional mediante um código apropriado.

Resultados obtidos por Campregher (2005), mostram que partindo de problemas com geometrias arbitrárias tridimensionais e não deformáveis, torna-se possível de se obter resultados compatíveis com a realidade utilizando-se metodologia de Fronteira Imersa, que será citada posteriormente. Partindo desses resultados, surgiu o intuito de efetuar simulações ao redor de geometrias complexas para que se possa aplicar a problemas de engenharia mais atuais como, simulação sobre automóveis e aeronaves.

Devido ao alto custo de aquisição de supercomputadores vetoriais, diversas universidades, empresas e centros de pesquisas estão empregando a tecnologia de processamento paralelo baseada em clusters de computadores pessoais (PCs), pois como citado acima, tais problemas de virtualização de protótipos e de cálculo das forças atuantes sobre os mesmos, requer grande poder computacional.

Os Sistemas Distribuídos foram criados para distribuir as tarefas e aumentar o poder computacional através do uso de vários processadores, como também, para promover o

compartilhamento de recursos. Cada processador possui sua própria memória e a comunicação entre os processadores é feita por meio de redes de comunicação, dando a esse conjunto de computadores a denominação de multi-computadores. Esses sistemas objetivam melhorar a comunicação entre os computadores, que pode envolver a facilidade de mudanças futuras (escalabilidade), rapidez nas trocas de informações e confiabilidade na execução dos processos. Com essa configuração de estrutura computacional, o cálculo a ser executado sobre geometrias complexas torna-se mais rápido e confiável.

Para estas geometrias complexas, a escolha apropriada da malha numérica depende da metodologia de discretização do domínio contínuo. Desta forma, um algoritmo desenvolvido, por exemplo, para lidar com malhas curvilíneas ortogonais, não é capaz de lidar com malhas não-ortogonais. Assim, diversas técnicas procuraram lidar com o problema, empregando malhas que se ajustam à sua topologia, quer seja acompanhando a superfície do corpo a ser estudado, quer seja se adaptando melhor ao escoamento (o caminho preferencial das linhas de corrente é uma das opções), melhorando a qualidade da solução (Campregher, 2005).

Além da escolha apropriada da malha, outro obstáculo na virtualização do protótipo, é a ferramenta necessária para gerar as geometrias. As ferramentas de geração de malha altamente automatizadas, geralmente possuem a capacidade de geração para praticamente todas as topologias, incluindo elementos hexaédricos, tetraédricos, híbridos hex/tet, ou ainda malhas cartesianas. Elementos adicionais podem ser gerados, incluindo-se células prismáticas e piramidais. Finalmente, malhas superficiais (de casca) com elementos tri ou quadri são suportadas utilizando-se solveres existentes.

Geometrias são criadas pela definição sucessiva de pontos, linhas orientadas, superfícies orientadas e volumes, compondo grupos de entidades geométricas que podem ser definidas, baseados nestas entidades geométricas.

A utilização desse tipo de ferramenta fica limitada à comercialização desses softwares, uma vez que a maioria são softwares proprietários, requerendo registro de suas licenças e tem um custo financeiro alto.

1.1. Motivação

Atualmente, o Laboratório de Transferência de Calor e Massa e Dinâmica dos Fluidos - LTMC/FEMEC/UFU possui diversas ferramentas para simulações e análise computacional da dinâmica dos fluidos. Em especial, manter-se-á o foco centrado na ferramenta de CFD desenvolvida por Campregher (2005), que permite simular escoamentos ao redor de

geometrias arbitrárias tridimensionais e não deformáveis, podendo se movimentar sob a ação de forças induzidas pelo próprio escoamento. Esta metodologia é uma extensão da metodologia originalmente bidimensional, proposta por Lima e Silva et. al. (2002) para representar o escoamento em torno de corpos estacionários.

Por requerer recursos computacionais elevados e devido ao alto custo de aquisição de supercomputadores vetoriais a configuração escolhida para o desenvolvimento do código computacional que caracteriza este trabalho é baseada em clusters de computadores pessoais (PCs), caracterizando processamento paralelo. Para isso, será utilizado um *cluster Beowulf* de 10 microcomputadores Pentium 4 (2.8 GHz / 1.5Gb RAM), disponível no LTCM. Todo o algoritmo é escrito em FORTRAN 90, utilizando a biblioteca de paralelização MPICH 2, e compilado pelo IFC (*Intel Fortran Compiler*), ambos *softwares* livre para uso acadêmico (Figura 1.1). A forma de se configurar um cluster, por não ser objetivo desta dissertação, será incluída como Anexo II do presente trabalho.

Figura 1.1. Cluster Beowulf

A potencialidade da metodologia de Fronteira Imersa deste código foi testada, obtendo-se resultados preliminares em problemas de interação fluido-estrutura, envolvendo escoamentos ao redor de uma esfera, contribuindo-se assim, com uma alternativa aos métodos correntemente empregados.

Mesmo tendo seu código validado, para efetuar tais simulações, são necessários

softwares proprietários (requerem registro) o que dificulta o desenvolvimento de aplicações acadêmicas devido ao custo dos mesmos.

Além disto, a ferramenta de CFD em questão, não oferece uma interface amigável de forma a facilitar o manuseio da aplicação. Para tal, serão utilizadas técnicas de programação visando desenvolver uma interface baseada em símbolos visuais, como ícones, menus e janelas. Seguindo esta linha de desenvolvimento, pode-se oferecer uma ferramenta simples para resolver problemas acadêmicos com entradas paramétricas e capacidade avançada de visualização. Com isso pode-se delimitar visualmente a etapa de pré-processamento, gerando arquivos de saída que possam ser usados com o código existente no LTCM, possibilitando maior facilidade de uso. Como o solver a ser utilizado é uma aplicação local residente no Cluster LTCM, uma maneira sugerida para permitir amplo acesso ao solver é através da web, visto que o cluster tem suporte à paginas da web (Servidor Apache). Vinculando a linha de desenvolvimento supra citada, com a viabilização do solver pela Internet, o presente trabalho irá caracterizar um web site, cuja definição será discutida posteriormente.

Juntamente com o problema de interação computacional, outro grande desafio em engenharia é a solução numérica da ação de forças sobre corpos imersos em escoamentos, com suas possíveis deformações e deslocamentos. Para tal problema uma solução viável pode ser obtida através da modelagem matemática, que não vem a ser objetivo de estudo do presente projeto.

Evidentemente, é notório o crescimento dos recursos computacionais, viabilizando a realização de simulações numéricas cada vez mais aperfeiçoadas de escoamentos em domínios de topologia diversas. Em geometrias complexas, a escolha apropriada da malha computacional depende da metodologia de discretização do domínio contínuo. Assim, procura-se lidar com o problema, empregando malhas que se ajustam à sua topologia, quer seja acompanhando a superfície do corpo a ser estudado, quer seja se adaptando melhor ao escoamento, melhorando a qualidade da solução.

Escoamentos ao redor de geometrias complexas ainda representam sérias dificuldades de solução, pois as geometrias apresentam problemas na geração da malha computacional, que nem sempre são triviais, e são propensas a inserir severos erros nos balanços entre seus nós elementares. Isto acontece quando se procura ajustar a malha computacional ao objeto de estudo e ao domínio, nas discretizações.

Nos casos de discretizações em blocos, o acoplamento entre os diversos domínios, se mal construído, pode implicar em sérias inconsistências físicas. A discretização de domínios tridimensionais exige grande capacidade de armazenamento e processamento de dados que, se somados aos freqüentes refinamentos locais das malhas, torna os cálculos muito

caros.

O uso da metodologia de Fronteira imersa desenvolvida no LTCM requer apenas uma malha cartesiana para a solução de escoamentos, independentemente da complexidade e da mobilidade da fronteira imersa. No entanto, requer-se a construção de uma malha superficial para a interface fluido-sólido. Empregando-se uma malha triangular para a discretização desta interface, torna-se possível aproveitar as facilidades associadas a este tipo de malha, para a geração de geometrias complexas.

Para tanto, no presente trabalho optou-se pela busca por softwares geradores de malhas que oferecessem a facilidade de criar geometrias complexas em formatos padrões como o STL, que seja portável e acima de tudo que seja uma ferramenta gratuita. Escolhida a ferramenta, mostrar-se-á que geometrias geradas podem apresentar resultados compatíveis com outras geometrias geradas por softwares proprietários de renome.

A razão de se buscar uma ferramenta geradora de malha, tem como motivo solucionar um problema da primeira versão do código (descrita em Campregher, 2005), na qual a importação de geometrias era trabalhosa e limitada a geometrias simples. Em seu trabalho desenvolvido no LTCM (Vedovoto, 2007), objetivou-se por otimizar a capacidade de importação de geometrias complexas, de forma a facilitar o uso do programa, oferecendo ao usuário, uma nova opção de tipo de arquivo a ser importado. A otimização da estrutura de paralelização do código utilizado para as simulações, alterando as possíveis topologias de paralelização, busca redução no custo computacional inerente às simulações. Com essas alterações, o código torna-se apto a aceitar uma interface gráfica visando maior facilidade no manuseio do solver.

Apesar da busca por um software gerador de malha ser um fator marcante para o presente projeto, para facilitar o uso da ferramenta de CFD do LTCM, existe ainda o problema em se escolher uma plataforma de desenvolvimento que possa agregar linguagens de programações distintas, uma vez que o código disponível no laboratório é escrito em Fortran e é uma ferramenta executável sobre plataforma Linux. Tal característica faz com que o usuário final da ferramenta tenha que ser hábil com o sistema operacional em questão, para que possa simular qualquer caso com o código disponível. Para isso, existe a necessidade de comparar linguagens de programação, buscando a melhor forma de integrar a interface ao código já existente.

Com o código do LTCM, levando-se em consideração a fase de pré-processamento, o usuário deve efetuar uma conexão com o cluster (local onde o código está armazenado e será executado em paralelo). Feita esta conexão, é necessário conhecer a estrutura de diretórios do código para localizar os arquivos os quais devem ser editados. Para editar os arquivos deve-se fazer uso de uma ferramenta adequada, através do comando “vi” do Linux,

como mostrado na figura 1.2 abaixo.

Todas as condições de contorno são configuradas manualmente em um arquivo texto, sendo que algumas das condições não precisam ser modificadas, não existindo a necessidade de que o usuário tenha permissão para executar tal modificação, assim como para visualizar o arquivo inteiro.

Além de visualizar um arquivo ao qual não se tem a necessidade de alterar todos os seus parâmetros, como feito com o arquivo de condições de contorno, no atual código, o usuário faz alterações no arquivo do programa principal. Tal tipo de operação é de grande periculosidade, pois caso o usuário não conheça a ferramenta de CFD do laboratório, ele poderá efetuar modificações de forma a inviabilizar o funcionamento do código.

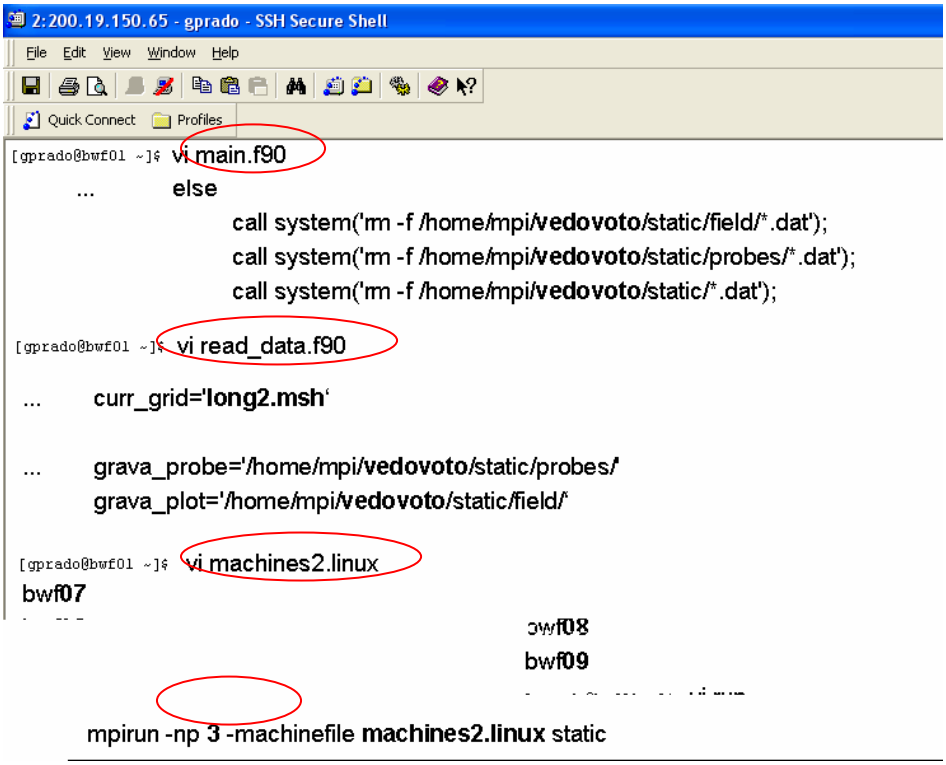


Figura 1.2. Exemplo de configuração de uma simulação com o código do LTCM, utilizando comandos do linux.

Algumas modificações no arquivo do código principal têm como função, mostrar o local (pasta de arquivos) na qual o programa irá buscar informações para executar as instruções. Uma solução viável para isso, foi adotada ao inserir uma variável global no programa principal a ser configurada pelo usuário (Vedovoto, 2007), de modo que o mesmo só indique os caminhos onde irá buscar e salvar informações apenas uma vez, e elas sejam enviadas ao código principal sem que o usuário interfira neste processo.

Juntamente com as modificações acima citadas, existe a necessidade de criar um campo

na interface gráfica para determinar quais máquinas irão executar a simulação computacional utilizando um ícone, as informações são validadas para iniciar a simulação.

Com todas essas colocações, tem-se a base para desenvolver uma interface gráfica baseada no código do LTCM, tornando amigável o solver do laboratório.

1.2. O Problema

Até o final da década de 70, os computadores eram utilizados principalmente para resolver problemas administrativos, ficando a interação com o usuário em segundo plano (Gattass–Levy, 1992). A comunicação do usuário com os programas era bastante rudimentar: a entrada dos dados era feita através de arquivos textos formatados segundo a conveniência dos programas e não dos usuários, onde era utilizado um editor de texto, linha a linha, e os resultados eram apresentados em sua maioria na forma de relatórios impressos.

Com o avanço tecnológico e a diminuição das limitações dos recursos computacionais, o uso dos computadores passou a abranger outras áreas diferentes como a engenharia, até alcançar o uso doméstico, com a difusão dos micro-computadores. A partir daí, os computadores passaram a ser limitados não pela capacidade computacional, mas sim pela capacidade dos usuários de se comunicar com os programas.

Atualmente, as simulações numéricas mais sofisticadas requerem ferramentas integradas de geração de geometrias e malhas, com o objetivo de tornar o desenvolvimento de produtos e processos os mais eficientes possíveis. Analisando a interação homem-software referente ao código do LTCM, conclui-se que, apesar de funcional, a ferramenta acaba se tornando limitada por não ser intuitiva na configuração de seus projetos, fazendo com que o usuário tenha que dedicar grande parte de seu tempo para aprender a usá-la. Este é um indicativo de que, a maioria das interfaces, quando produzidas para requerer uma interação com o usuário, nem sempre são projetadas tendo em mente os usuários.

No final da década de 70, o centro de pesquisa da Xerox criou o sistema Star baseado na metáfora de uma mesa de trabalho (*desktop*) para fazer a interação homem-máquina. Após conhecer o sistema, em 1984, Steve Jobs criou o sistema de interface que roda, até hoje, nos Macintosh da Apple. O sistema projetado utiliza a metáfora de uma mesa de trabalho, além de utilizar extensivamente recursos gráficos; suas principais características são:

- menu de barra, que pode aparecer e desaparecer sob controle do *mouse*;
- janelas que exibem o que o computador está fazendo;
- ícones que representam arquivos e diretórios (*folders*);

- caixas de diálogos, botões e muitos outros elementos gráficos de interface representando técnicas de interação.

O sistema do Macintosh popularizou o uso das interfaces gráficas com usuário (GUI) fazendo surgir outros sistemas como: Macintosh Toolbox, MS-Windows, Presentation Manager, Motif, Open Look, HP-NewWave e Gem.

A grande diversidade e a falta de um padrão de fato para sistemas de interface criam um problema para os programadores na decisão de qual metodologia deve-se utilizar para gerar uma interface. A escolha de uma metodologia errada pode limitar o potencial de uma aplicação a uma determinada plataforma. Por outro lado, se o sistema escolhido se tornar um padrão em GUI, o número de potenciais usuários aumenta, melhorando a usabilidade da ferramenta. Como é de interesse do LTCM disponibilizar a ferramenta do laboratório para uso acadêmico, esta tem que ser o mais intuitiva possível visando facilitar sua operação, sendo de extrema importância o software para concebê-la. Além disso a plataforma que irá disponibilizar o acesso do solver também é de suma importância, uma vez que se disponível via web, que é a intenção deste projeto, o solver poderá ter maior acessibilidade.

Para uma solução viável ao desenvolvimento desta interface é importante a análise de vários sistemas. Contudo, programar um aplicativo interativo que suporte vários sistemas de interface é uma tarefa penosa, pois a experiência mostra que aproximadamente metade de um programa é responsável pela interação com usuário (Marcus–Van Dam, 1991). Isto se deve principalmente à necessidade de controlar vários dispositivos de entrada simultaneamente (e.g., *mouse* e teclado) e de atender às necessidades cognitivas dos usuários. Nota-se que as primitivas gráficas necessárias na geração de um *feedback* são diferentes entre as plataformas, implicando na existência de um conjunto de funções de interface diferente para cada ambiente, aumentando significativamente o trabalho de documentação e manutenção.

Dentre as ferramentas para produção de GUI, duas classes podem ser citadas Programação Orientada a Objeto e Sistemas Gerenciadores de Interface de Usuário (UIMS – *User Interface Management Systems*). A programação orientada a objeto é um sucesso no desenvolvimento, pois os objetos de uma interface como botões e outras estruturas deste tipo, podem ser prontamente anexadas como objeto da linguagem. Um UIMS é o conjunto de programas interativos de alto nível para projetar, criar o protótipo, executar, avaliar e manter interfaces com usuários, tudo integrado sob uma interface de desenvolvimento de diálogos simples (Glenn - Chignell, 1992). O propósito dos UIMS é abstrair os detalhes de implementação de interface, para permitir ao programador especificar e manipular interfaces a um nível mais alto de abstração. Os UIMS pressupõem que o desenvolvimento de uma aplicação seja feito por dois especialistas trabalhando em conjunto: um na área do problema

em questão e outro na área de interface com o usuário. O primeiro especialista resolve o problema computacional da aplicação. O segundo se preocupa com fatores humanos, relacionados com aspectos psicológicos, cognitivos, ergonométricos e lingüísticos, de modo a proporcionar uma verdadeira interação entre usuário e aplicação (Hartson–Hix, 1990).

A interface de UIMS gerência a tela (criação) e monitora os dispositivos de entrada. Mapeia as ações do usuário sobre os dispositivos de entrada e interpreta operações de saída produzindo a tela. As entradas do usuário são tokens produzidos pelo componente de apresentação. A seqüência em que estes tokens são entregues ao controle de diálogo cria o diálogo e produz informação da interface com a aplicação, seu estado pode ser modificado e provocar comando de saída que serão interpretados pelo componente de apresentação. No sentido contrário, o controle de diálogo recebe informação da interface com a aplicação, seu estado pode ser modificado e provocar comandos de saída que serão interpretados pelo componente de apresentação. A interface de aplicação recebe pedidos do controle de diálogo que pode realizar mapeamentos antes de entrega-los à aplicação. Este componente pode responder diretamente aos pedidos do controle de diálogo quando o pedido não for sintaticamente válido, e transformar a saída da aplicação antes de transmiti-la ao controlador de diálogo. Algumas ferramentas UIMS podem ser citadas para exemplificar esta categoria, sendo elas: Tigre, Menulay, Hypercard e Internet de ViewSoft, porém uma ferramenta mais atual e de fácil uso foi utilizada neste projeto, sendo o DREAMWEAVER esta ferramenta.

Avaliando o propósito deste projeto, uma vez que o código do LTCM já se encontra funcional e parcialmente validado, o ideal então é utilizar ferramentas de classe tipo UIMS para não modificar a estrutura do código atual. Utilizar ferramentas orientadas a objeto podereia dar uma maior complexidade e consistência ao projeto, porém, o intuiuto do mesmo é que a interface gráfica a ser criada possa ser complementada caso haja ampliações no código computacional existente, então, facilitar o desenvolvimento da interface, facilita também o entendimento de seu desenvolvimento, sendo passível de ser continuada por outrem.

1.3. Objetivo

O objetivo da presente dissertação é descrever o projeto e a implementação de um sistema prático de interface com usuário que ofereça suporte ao sistema atual utilizado pelo LTCM, viabilizando o uso desta ferramenta de CFD.

O projeto da interface será efetuado para solucionar alguns problemas encontrados

na ferramenta atual, preservando a sua estrutura e acrescentando duas principais qualidades: facilidade de uso, tanto pelos programadores quanto pelos usuários finais da aplicação; portabilidade para diferentes ambientes computacionais. As principais características são:

- Permitir composição de diálogos, através da descrição de um *layout*;
- Ter uma linguagem de especificação de diálogos (LED) compilada em tempo de execução da aplicação;
- Separar o sistema de interface do código computacional;
- Ser expansível.

Além disso, para auxiliar essa interface, será elaborada uma pesquisa para selecionar um software livre capaz de gerar malhas tridimensionais, de forma a padronizar a criação de geometrias complexas, tendo como exigência principal, que as malhas sejam formadas por elementos triangulares, com a finalidade de aproveitar as facilidades oferecidas por malhas com este tipo de elemento, podendo assim ser utilizada juntamente com a metodologia de Fronteira Imersa.

CAPITULO 2

Revisão Bibliográfica

Sempre que se pretende resolver numericamente um determinado problema físico, a primeira etapa a ser realizada é a discretização do domínio contínuo original. No Método das Diferenças Finitas (*Finite Difference Method – FDM*), o domínio contínuo é discretizado por pontos conhecidos como nós, cujo conjunto forma a malha mostrada na Figura 2.1.

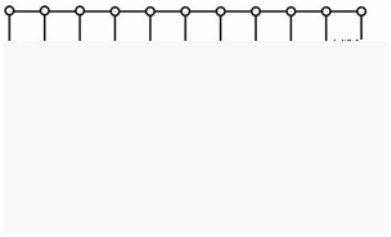


Figura 2.1. Domínio discretizado por malhas ou pontos nodais.

É possível que o domínio de cálculo de um escoamento seja representado por malhas extremamente simples (cartesianas), utilizando sem grandes dificuldades, diferenças finitas ou volumes finitos para a solução das equações de Navier-Stokes. Para metodologia de fronteira imersa, que será explicada posteriormente, os corpos imersos podem ser representados como um conjunto de pontos discretos, estes pontos representam a geometria como um campo de força, de forma que a distância entre eles não permita que o escoamento passe pelo seu interior, Figura 2.2 (Vedovoto, 2007), porém a representação em arquivo requer informações como pontos e conectividades entre os mesmos, ou no caso do arquivo STL, Pontos e as normais aos mesmos.

De acordo com estudos anteriores, uma construção do conjunto de pontos discretos, denominados pontos lagrangianos, foi realizada com a ajuda de uma malha de elementos triangulares. Como a única função desta malha é melhor posicionar os pontos lagrangianos, foram escolhidos elementos triangulares (de mais fácil manipulação numérica) nos quais se insere apenas um ponto lagrangiano (Campregher et al., 2005).

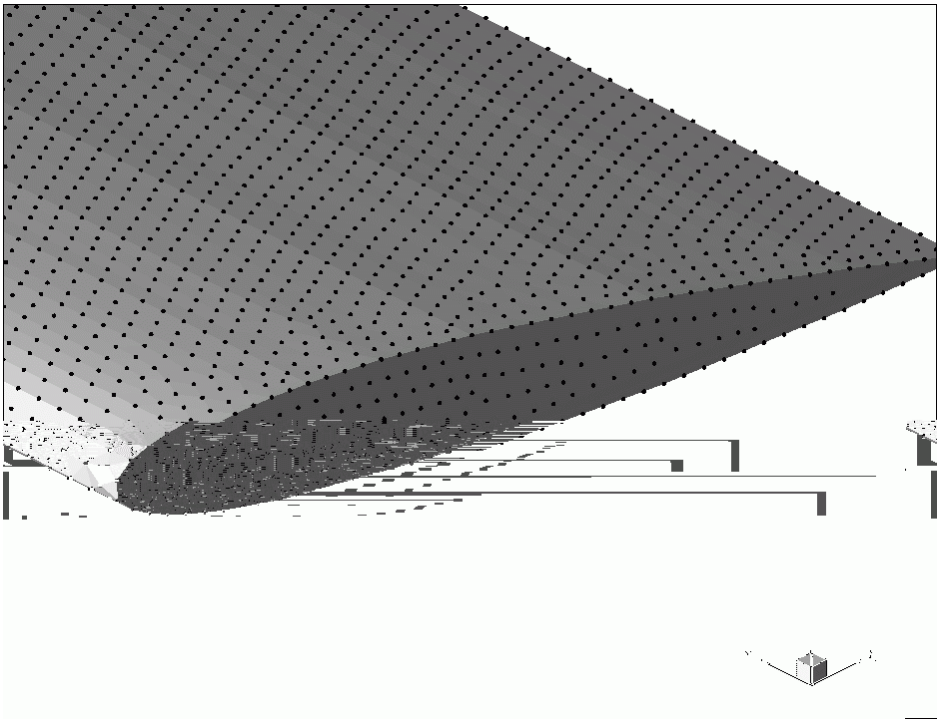


Figura 2.2. Caracterização de um corpo por pontos discretos (Vedovoto, 2007).

Em Campregher (2005), foi feita uma discretização com aproximações espaciais e temporais de segunda ordem, empregando Volumes Finitos em malhas cartesianas e com capacidade de processamento paralelo. Um domínio lagrangiano foi construído com uma malha de elementos triangulares.

Empregando-se uma malha deste tipo, acredita-se ser possível aproveitar suas facilidades na geração de geometrias complexas, o que constituiria uma importante qualidade, caso se deseje aplicar a metodologia em problemas reais de engenharia.

A partir dessas características, necessárias para se gerar uma malha, foi feita uma pesquisa a fim de encontrar uma ferramenta capaz de gerar tais geometrias e suas malhas, de forma que tal ferramenta fosse de fácil manuseio e que tivesse baixo custo, preferencialmente que seja livre de comércio.

Além de tais qualificações, o software precisa ser portável para que se possa trabalhar em uma plataforma livre, já que no Laboratório LTCM utiliza-se um cluster Beowulf base Linux, mas que também possa ser executado em plataforma Windows, já que a grande maioria de usuários de tecnologia, tem maior conhecimento de utilitários nesta plataforma.

2.1. Fronteira Imersa e Formato STL de Arquivo

O código Fluids-3D permite lidar com soluções numéricas da ação de forças sobre corpos imersos em escoamentos, com suas possíveis deformações e deslocamentos,

utilizando modelagem matemática.

Uma metodologia alternativa para se resolver escoamentos sobre geometrias complexas e móveis vem sendo desenvolvida no LTCM. Trata-se da metodologia de Fronteira Imersa (*Imersed Boundary* - *IB*). Neste sentido, a presença de uma interface fluido-sólido, ou mesmo de uma interface fluido-fluido, pode ser simulada pela adição de um termo de força (que alguns autores denominam como termo forçante) às equações do movimento do fluido. Dessa forma, o escoamento reconhece a presença de um objeto imerso, apenas pela ação de um "campo de força". A forma como este campo de força é calculado diferencia a metodologia empregada atualmente. Cabe salientar que tais forças são formadas a partir de efeitos diversos, agindo ao mesmo tempo no fluido, em decorrência de cisalhamentos e deformações, variações na quantidade de movimento e gradientes de pressão (Campregher, 2005).

Assim, como a metodologia baseia-se na ação de forças "externas", é possível que o domínio de cálculo do escoamento possa ser representado por malhas extremamente simples, utilizando apenas coordenadas cartesianas e discretizando por Volumes Finitos ou Diferenças Finitas. Enquanto isso, a representação do corpo imerso pode ser feita por um conjunto de pontos discretos qualquer, que pode assumir as mais diversas posições, no espaço e no tempo, dentro do domínio do escoamento. De forma geral, costuma-se posicionar o domínio do escoamento por um sistema euleriano de coordenadas (em que o material que está sendo estudado se move através do domínio computacional) e o objeto em estudo por um sistema lagrangiano, em que o domínio computacional se move com o material em estudo.

Esta metodologia foi aplicada por Unverdi e Tryggvason (1992) em escoamentos bifásicos, onde a força inserida no termo-fonte é modelada com base na tensão superficial presente na interface entre os dois fluidos. Com essa metodologia, os autores realizaram simulações de bolhas em domínios bi e tridimensionais, onde uma linha reta unindo os pontos nos problemas 2D é substituída por um elemento triangular nos casos 3D, nos moldes das formulações em Elementos Finitos. Geometrias tridimensionais, geradas por malhas de elementos triangulares e inseridas em malhas cartesianas, foram propostas por Gilmanov *et al*. (2003).

Conforme conclusões tiradas dos estudos mencionados acima, para o código atual do LTCM, é adotado o uso de malhas triangulares para simular a iteração fluido-sólido, com a finalidade de aproveitar as características desse tipo de malha. Para gerar a malha da geometria a ser inserida no escoamento, faz-se uso de ferramentas proprietárias como ANSYS.

Como mencionado, a principal característica da metodologia de fronteira imersa é

simular a presença de uma interface sólida ou fluida no meio do escoamento via inserção de um termo de força $\vec{f}_X$, nas equações de Navier-Stokes para o fluido.

O ANSYS gera a geometria e exporta dois arquivos, um deles contendo os pontos (nós) da geometria e outro contendo a conectividade entre cada ponto. A desvantagem deste tipo de arquivo, é que quando inseridos no solver do LTCM, necessita-se de um custo computacional adicional, pois com as informações que eles fornecem ainda é necessário que se execute cálculos sobre eles para encontrar o centróide de cada elemento triangular da malha, além de ser necessário otimizar os arquivos, retirando linhas com informações desnecessárias.

Para reduzir este esforço computacional adicional, uma das soluções já adotadas é fazer uso de arquivos no formato STL (*Standard Tessellation Language*), que é um formato nativo do software de *Stereolithography*. O formato STL é importado e exportado por muitos outros pacotes. O STL descreve somente a geometria de superfície de um objeto tridimensional sem nenhuma representação da cor, da textura ou de outros atributos comuns do modelo do CAD. O formato de STL é especificado em ASCII e representações binárias, embora os formatos binários sejam mais comuns por serem mais compactos. Um formato STL descreve uma superfície triangular não estruturada, formada pela normal e pelos vértices dos triângulos, usando um sistema cartesiano tridimensional de coordenadas.

2.1.1. ASCII STL

Um arquivo ASCII STL começa com a linha

```
solid name
```

onde o nome é uma string opcional. O arquivo continua com vários números de triângulos, representados como segue

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

e finaliza com

.

A estrutura do formato sugere que outras possibilidades existem, por exemplo, faces com mais de um *loop* ou *loop* com à exceção de três vértices mas, na prática, todas os faces são triângulos simples. O espaço branco (espaços, tabulações, novas linhas) pode ser usado em qualquer lugar no arquivo exceto dentro de números ou palavras. Os espaços entre os identificadores de estrutura, *facet* e *normal*, e entre *outer* e *loop* são necessários.

Um arquivo STL é uma representação triangular de uma superfície de geometria 3D. A superfície é logicamente desenhada em um conjunto de triângulos orientados chamados faces. Cada face é descrita pela unidade normal composta por três pontos externos e listados na ordem que representam os vértices do triângulo. Enquanto a relação de aspecto e orientação de faces individuais são determinadas pela curvatura de superfície, o tamanho das faces é definido pela tolerância que controla a qualidade da representação de superfície em termos da distância das faces da superfície. A escolha da tolerância é fortemente dependente na aplicação designada para o arquivo STL produzido. Em processo industrial onde máquinas de stereolithography executam cálculos para gerar uma camada de superfície feita de uma resina foto-sensível, a tolerância pode estar em ordem de 0,1 mm para produzir geometrias 3D com partes altamente precisas. Porém, valores maiores são tipicamente usados em pre-produção de protótipos de STL, por exemplo, para propósitos de visualização.

Hoje em dia, modelagem de elemento finito envolve discretização de objetos muito complexos em termos de geometria e topologia. Enquanto sofisticadas estruturas de dados para a descrição de topologias arbitrárias estão disponíveis, a gama de geometrias que pode ser controlada através de algoritmos existentes está bastante limitada. Particularmente malhas de superfície 3D são restritas pela complexidade associada com a descrição matemática da superfície. A maioria dos algoritmos pode controlar superfícies paramétricas 3D, porém muitas aplicações se tratam das superfícies de natureza discreta como malhas de elementos finitos deformadas, gride de pontos esquadrinhadas por tomografia de computador, representações de terrenos digitais, etc.

Uma classe importante de superfícies 3D, descritas de forma discreta, utiliza o grupo de superfícies no formato de arquivo stereolitografia. O STL tem se tornado o formato de dados padrão para transmissão de dados para rápida prototipagem industrial. Este formato aproxima superfícies 3D de um modelo sólido com triângulos orientados (faces) de tamanho diferente e forma (relação de aspecto), para alcançar uma representação bastante satisfatória para processo industrial de partes 3D que usam máquinas de stereolitografia. Produzir um protótipo STL de uma geometria pode aumentar o *design* de um produto, ao se incluir sua visualização geométrica. Inicialmente, os arquivos de STL foram introduzidos como formato de arquivo nativo do software STL, criado por companhia de Sistemas 3D.

Atualmente, muitos sistemas são capazes de produzir um arquivo de STL, tornando o STL um formato geral de representação de superfície 3D muito atraente para uso prático de pré-processamento de geometria, que é o principal objetivo para o qual é utilizado no LTCM.

A função da esteliografia é discretizar superfícies 3D, para classifica a superfície da geometria através de dados discretos no formato STL. A partir de um modelo, Figura 2.3, tal discretização, consiste em varias fases, sendo elas:

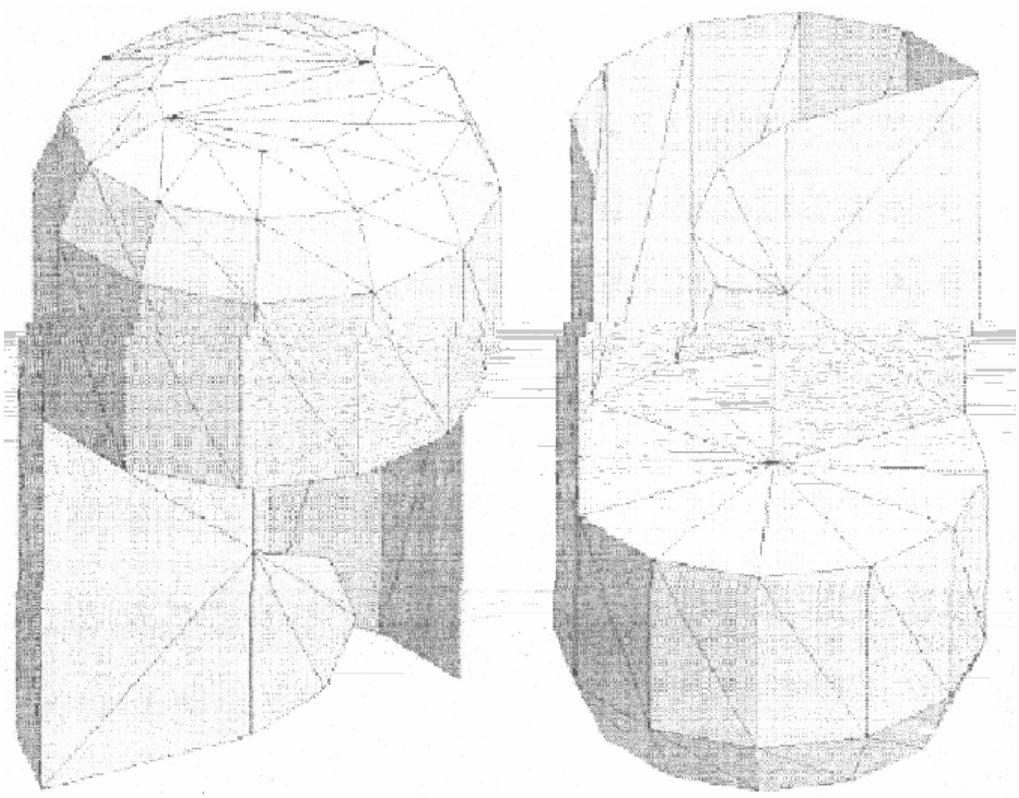


Figura 2.3 – Malha STL de um objeto Simples

1. Extração (obtenção) dos limites de representação da geometria: Inicialmente uma representação dos limites do modelo inteiro é construída no arquivo STL, que usa o reconhecimento de características topológicas e operações geométricas apropriadas. Deste modo, entidades distintas (vértices, curvas e superfícies) de natureza topológica, ou de natureza geométrica (características pontiagudas) são estabelecidas. Figura 2.4.
2. Identificação dos nós da topologia e da geometria: quando todas as bordas são identificadas, os nós correspondentes às características topológicas são identificados e classificados como vértice modelo. Estes são todos nós que não são compartilhados exatamente por duas bordas (linhas), Figura 2.5 (a). Agora os nós que possuem uma característica geométrica correspondente são procurados, sendo estas características: Nós vizinhos que formam linhas contínuas; Nós

compartilhados por duas bordas; Nós cujos ângulos entre eles foram prescritos pelo usuário. (b).

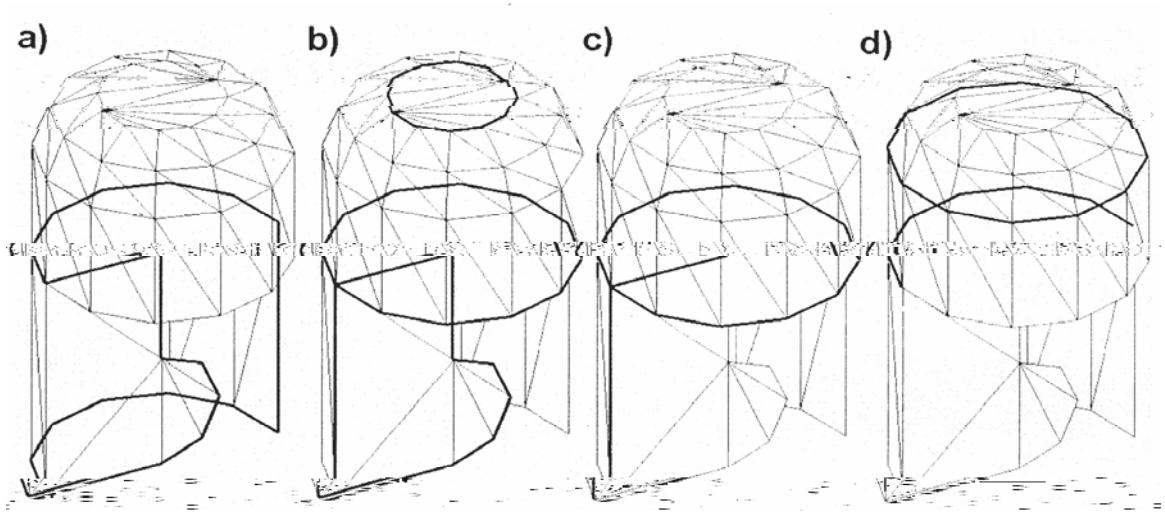


Figura 2.4 – Identificação de Topologias e bordas da geometria: a) bordas topológicas; b) bordas geométricas e planos; c) bordas geométricas entre faces; d) bordas geométricas entre faces, com diferentes raios.

a) **b)**

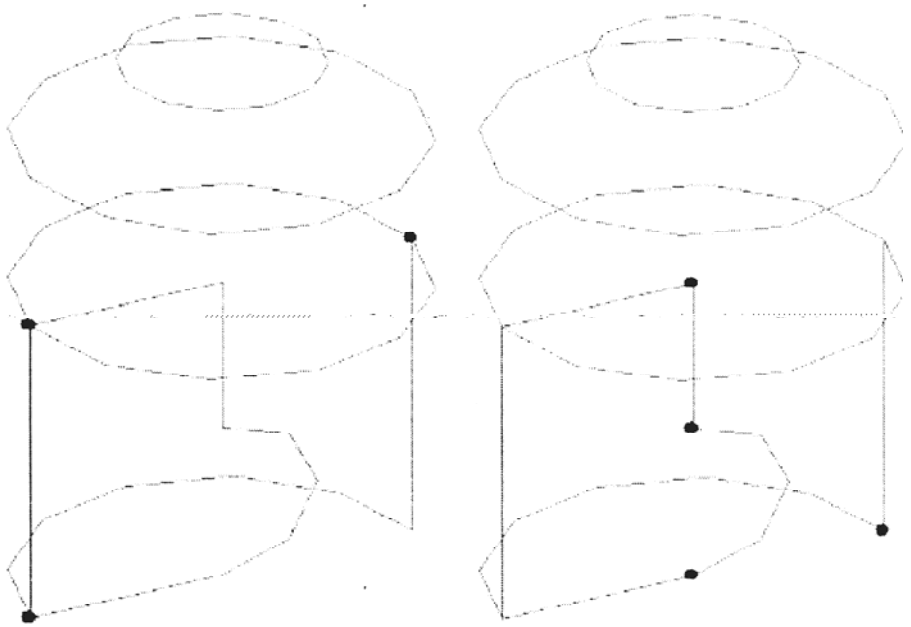


Figura 2.5 – a) Nós da topologia; b) Nós da geometria

3. Identificação das curvaturas: Aquelas formadas com a utilização de vértices, Figura 2.6 (a), e aquelas formadas por loop, Figura 2.6 (b).
4. Identificação dos modelos de superfícies através da distinção de cada superfície por tons de cinza, como mostrado na figura 2.3.

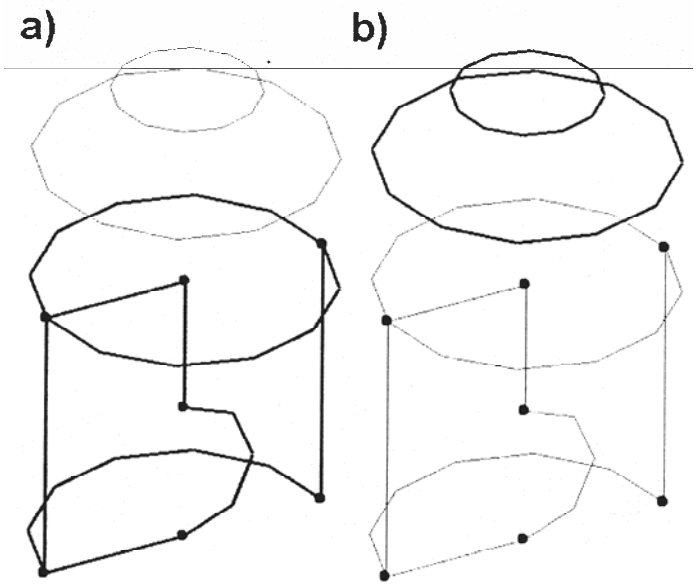


Figura 2.6 – Identificação de modelos de curvas; a) curvas com vértices; b) curvas formadas por loop

5. Na Próxima fase, uma superfície suavizada é recuperada sobre a malha da geometria. Isto é realizado através da interpolação de subdivisões utilizando uma modificação do modelo Butterfly (Zorin, 1997), que aproxima a superfície de uma interpolação de subdivisões, produzindo as curvas o mais semelhante possível da superfície. Esse processo de subdivisão anterior tem como objetivo prevenir alguns efeitos indesejáveis na recuperação da malha da superfície. Figura 2.7.

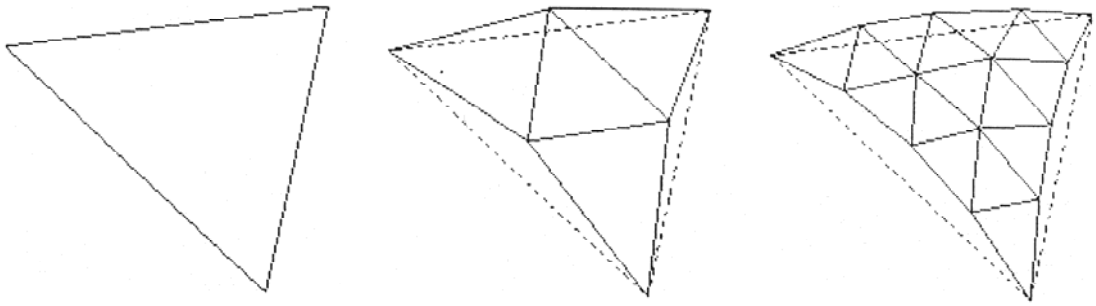


Figura 2.7 – Subdivisão da Malha.

6. Na última fase, a superfície limite reconstruída é submetida a uma triangularização. Figura 2.8.

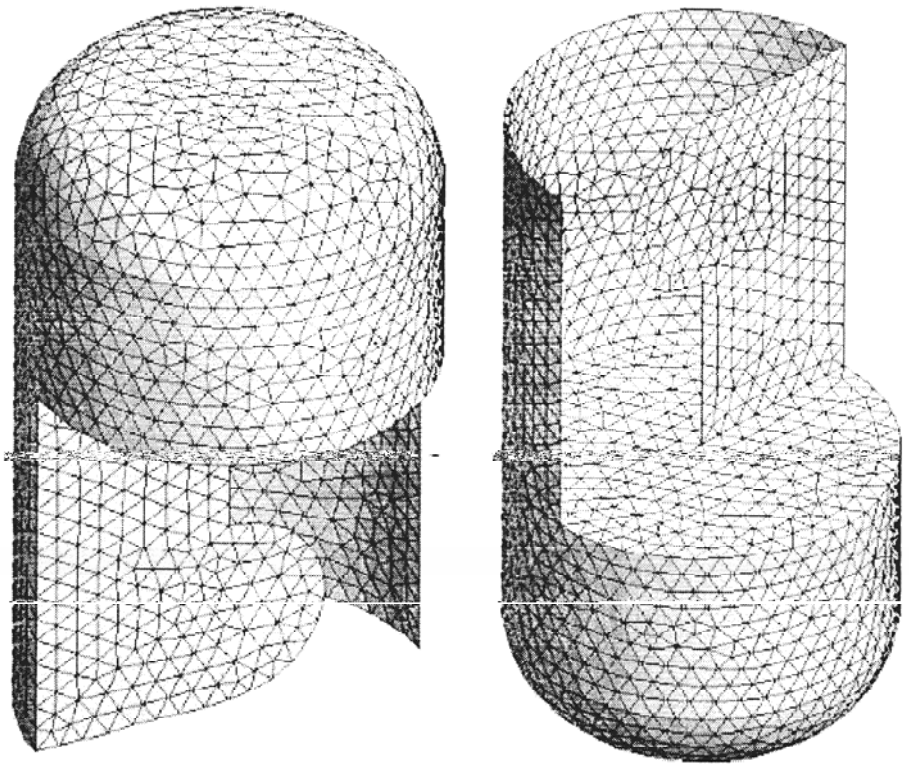


Figura 2.8 – Malha STL do objeto modelo

Analizando todos estes passos para se gerar um arquivo no formato STL, todas as características que sua estrutura armazena e retransmite, pode-se concluir que, utilizando este tipo de arquivo, obtém-se um ganho computacional em termos de tempo de processamento, já que a parte para virtualização do protótipo necessita de grande gama de cálculos computacionais.

2.1.2. A face normal do STL

No ASCII e em versões binárias de STL, a face normal de um elemento triangular deve ser um vetor unitário que aponta para fora do objeto sólido. Na maioria dos softwares isto pode ser ajustado e o software calculará automaticamente uma normal baseada na ordem dos vértices do triângulo usando a “regra da mão direita”. Os carregadores de STL (por exemplo, o STL *plugin*) certificam-se de que a normal no arquivo concorde com a normal calculada usando a regra da mão direita e advertem-se quando não concorde.

Ao obter-se a face normal de um elemento triangular, as informações contidas no formato STL auxiliam nos cálculos do centróide de cada elemento da malha triangular. Uma vez que o posicionamento do ponto central de cada triângulo da geometria é em função de seus vértices, ao se utilizar arquivos de nós e conectividades, existe a necessidade de ler

dois arquivos para saber a quais vértices o centróide fará referência. Como o STL já contém os vértices de referência e suas coordenadas, achar o centróide é apenas uma operação de média das coordenadas dos pontos. Com isso pode-se gerar a malha lagrangiana da superfície.

2.2. Aplicação local, Aplicação web e WebSite.

Levando em consideração que o solver do LTCM oferece uma solução computacional que necessita de uma arquitetura especifica (Cluster), e que esta arquitetura não se encontra disponível em qualquer lugar, neste projeto optou-se em utilizar a Internet para acessar o código do Solver Fluids3D.

A proposta do presente projeto tem como principal vantagem em relação as outras ferramentas de CFD, a não necessidade de *download* ou encaixe do software na máquina que o usuário estiver utilizando para acessar o código.

Como não há necessidade de efetuar qualquer instalação na maquina do usuário, existe então a necessidade de uma plataforma para executar este solver, portanto, para tal execução, no atual projeto utiliza-se de uma plataforma denominada cliente-servidor.

Numa aplicação cliente-servidor o cliente (aquele que instancia a interface do aplicativo) liga-se a um servidor de aplicação ou sistema de base de dados. Quando um cliente se conecta diretamente com um sistema de base de dados, ou com uma aplicação servidora monolítica, a arquitetura da aplicação é uma arquitetura duplamente ligada.

Os protocolos da Camada de Aplicação fornecem serviços do nível mais alto, tais como webservices, servidores de nomes, correio eletrônico, login remoto (telnet e SSH), do tipo cliente-servidor. Os protocolos do nível de transporte fornecem serviços que garantem uma transferência confiável de dados e aplicativos entre computadores (ou outros equipamentos) remotos. Os programas na camada de aplicação usam os protocolos de transporte para contatar outras aplicações. Para isso, a aplicação usam os protocolos de transporte para contatar outras aplicações. Para isso, a aplicação interage com o software do protocolo local que está pronto a aceitar a mensagem. A aplicação que estabelece a conexão usa os protocolos de transporte e rede para compactar o sistema que aguarda. As mensagens entre as duas aplicações são trocadas através da conexão resultante.

O paradigma cliente-servidor é usado praticamente em todos os processos distribuídos em que a aplicação servidora aguarda conexões, executa serviços e retorna resultados. Já a aplicação cliente é quem estabelece a conexão com o servidor, envia mensagens para o mesmo e aguarda pelas mensagens de resposta.

Analisando o tipo de plataforma acima citado, pode-se estabelecer que caso o usuário fosse utilizar o código computacional, diretamente na máquina onde este está

instalado, isto denominaria uma aplicação local, o que não cabe para solucionar este projeto.

Abordada a definição da plataforma cliente-servidor, dar-se-á as definições para esclarecer a diferença de aplicações web e website que são exemplos de implementações que suam a plataforma cliente servidor.

2.2.1. Aplicação Web

Aplicação Web é o termo utilizado para designar, de forma geral, sistemas de informática projetados para utilização através de um navegador, na Internet ou em redes privadas. Trata-se de um conjunto de programas que é executado em um servidor http (Web Host). O desenvolvimento da tecnologia web está relacionado, entre outros fatores, à necessidade de simplificar a atualização e manutenção mantendo o código-fonte em um mesmo local, de onde ele é acessado pelos diferentes usuários.

As aplicações Web podem ser vistas como formulários seguros que enviam mensagens pré-formatadas para usuários do servidor. Elas são desenvolvidas para interagir com bases de dados, para coletar, armazenar, organizar e disseminar informação, criando poderosas ferramentas para uso em consistentes sistemas de gerenciamento de informação.

As aplicações Web são configuradas para estarem acessíveis apenas a determinados usuários ou grupos de usuários. Após determinar a localização da URL no servidor da empresa para cada Aplicação Web ou Modelo, ela pode simplesmente ser “arrastada” para a pasta ‘includes’ de um usuário ou grupo e isso a tornará disponível. A aplicação estará disponível no console do usuário sempre que ele estiver conectado.

2.2.1. Website.

Um site ou website é um conjunto de páginas Web, isto é, de hipertextos acessíveis geralmente pelo protocolo http na Internet. O conjunto de todos os sites públicos existentes compõem a World Wide Web. As páginas num site são organizadas a partir de uma URL básica, onde fica a página principal, e geralmente residem no mesmo diretório de um servidor. As páginas são organizadas dentro do site numa hierarquia observável na URL, embora as hiperligações entre elas controlem o modo como o leitor se apercebe da estrutura global, modo esse que pode ter pouco a ver com a estrutura hierárquica dos arquivos do site.

Alguns sites, ou parte deles, exigem uma subscrição com o pagamento de uma taxa,

por exemplo, mensal, ou então apenas um registro gratuito.

Sites são escritos geralmente em HTML, ou dinamicamente convertidos para esta linguagem e acessados usando um software cliente chamado web browser ou navegador. Sites consistem de páginas HTML estáticas ou páginas criadas dinamicamente usando tecnologias como JSP, PHP ou ASP.

Existem numerosos tipos de sites, cada um especializado em um serviço ou uso em particular. Atualmente, dentre os principais tipos existentes, destacam-se os seguintes:

- Institucionais: servem como ponto de contato entre uma instituição e seus stakeholderes (clientes, fornecedores, etc.). No caso de instituições comerciais, são usados geralmente para comercio eletrônico, recrutamento de funcionários, etc. No caso de instituições sem fins lucrativos, servem principalmente para divulgarem seus trabalhos, informarem a respeito de eventos, etc. É o tipo de site mais comum na internet. Podem ainda ser sites pessoais, geralmente mantidos por profissionais liberais.
- Midiáticos: São sites informativos com atualizações freqüentes e periódicas. Nem sempre o conteúdo é baseado em texto puro, podendo conter variados elementos multimídia. Muitos deles podem ser “assinados” por meio de feeds RSS, que servem para notificar as atualizações, e muitos deles incluem espaços para comentários dos leitores. Nesta categoria também se incluem sites não necessariamente vinculados à informação, como sites de entretenimento e sites de conteúdo adulto.
- Aplicativos: são sites interativos cujo conteúdo consiste de ferramentas de automatização, produtividade e compartilhamento, substituindo aplicações de desktop. Podem ser processadores de texto, planilhas eletrônicas, editores de imagem, softwares de correio eletrônico, agendas, etc.
- Bancos de Dados: servem para catalogar registros e efetuar buscas, podendo incluir áudio, vídeo, imagens, softwares mercadorias, ou mesmo outros sites. Pode ser de dois tipos.
- Comunitários: são os sites que servem para a comunicação de usuários com outros usuários da rede. Nesta categoria se encontram os chats, fóruns e sites de relacionamento.
- Portais: servem para congregar conteúdos de diversos tipos entre os demais tipos, geralmente fornecidos por uma mesma empresa. Recebem esse nome por congregarem a grande maioria dos serviços da internet num mesmo local.

Como a interface a ser criada não irá substituir uma aplicação desktop, o projeto atual então, não pode ser denominado como uma aplicação web, e sim apenas um website com possível definição de institucional, no caso o LTCM.

2.3. Requisitos de Interface de Softwares

Quando se deseja construir um software, interface, ou algum sistema de interação Homem-Computador, deve-se ter alguma noção do que o sistema fará. Frequentemente, a interface substituirá uma já existente ou o mode de realizar tais operações, como configuração de serviços. Algumas vezes, a nova interface é um aprimoramento ou uma extensão de uma interface atual. Cada vez mais freqüente, a interface é planejada para tarefas que nunca haviam sido realizadas antes, não importanto se sua funcionalidade é nova ou antiga, cada uma tem um propósito geralmente expresso em termos do que a interface pode fornecer ao sistema. Um requisito é uma característica do sistema ou a descrição de algo que o sistema é capaz de realizar, para atingir seus objetivos.

De acordo com a definição de requisitos, algumas tarefas que o sistema é capaz de realizar podem ter seus diagramas elaborados, sendo alguns deles: Diagrama de Fluxo de Dados, Diagrama de Caso de Uso e Diagrama de Caso de Teste.

Um sistema de software é feito para servir seus usuários. Portanto, para construir um sistema de sucesso devemos saber quem são seus usuários potenciais e o que eles querem e precisam. O termo usuário representa alguém ou alguma coisa (como um outro sistema) que interage com o sistema que está sendo desenvolvido.

Um caso de uso é um pedaço de funcionalidade do sistema que dá ao usuário um resultado de valor (Martins, 1999). Casos de uso capturam requisitos funcionais e todos juntos resultam no modelo de caso de uso, o qual descreve a funcionalidade completa do sistema.

A estratégia de casos de uso pode ser caracterizada pela pergunta: o que o sistema faz para cada usuário? Estas palavras têm uma implicação muito importante. Faz com que se pense em termos dos valores dos usuários, não apenas em funções ou poderiam ser interessantes.

Os casos de uso direcionam o processo de desenvolvimento, já que, baseados no modelo de casos de uso, o analista cria um série de modelos de projeto e implementação que os realizam efetivamente. Os responsáveis pelos testes realizam seu trabalho com o propósito de garantir que os componentes do modelo de implementação cumpram corretamente os objetivos estabelecidos nos casos de uso. Desta forma, os casos de uso não somente iniciam o processo de desenvolvimento, mas também o mantém coeso. Direcionado a casos de uso significa que o processo de desenvolvimento executa uma seqüência de tarefas derivadas dos casos de uso. Eles são especificados, projetados e servem de base para a construção dos casos de teste (Martins, 1999).

Embora seja verdade que os casos de uso dirijam o processo, eles não são

selecionados isoladamente. São desenvolvidos juntamente com a arquitetura do sistema. Ou seja, os casos de uso direcionam a arquitetura do sistema, que por sua vez influencia a seleção dos casos de uso. Portanto, ambos amadurecem no decorrer do ciclo de vida do sistema.

A seguir serão abordadas as técnicas de D.F.D. – Diagrama de fluxo de dados – com o objetivo de oferecer noções básicas sobre o assunto relacionado, facilitando assim a visualização do funcionamento do projeto desta dissertação.

2.3.1. Ciclo de Vida

Um Processo Unificado consiste da repetição de uma série de ciclos durante a vida de um sistema. Cada ciclo é concluído com uma versão do produto, pronta para a distribuição. Este ciclo subdivide-se em quatro fases, concepção, elaboração, construção e transição, onde cada fase, por sua vez é subdividida em iterações que passam por vários *workflows* (passos dentro de uma interação) do processo: modelagem de negócios, requisitos, análise e projeto, implementação, teste, implantação, gerência de configuração e mudança, gerenciamento de projeto e ambiente.

Uma iteração típica realiza algumas atividades ou fluxos de atividades. Porém, a importância de cada fluxo de atividade depende da fase em que a iteração se encontra.

É importante ressaltara que o fluxo de Análise do Projeto do RUP foi alterado para considerar mais apropriadamente o desenvolvimento de aplicações E-Commerce sem perder, contudo, a sua adequabilidade a outros tipos de domínios de aplicação. As alterações foram realizadas de forma a não deturpar a natureza de adaptabilidade e genericidade do processo (Araújo, 2001). A seguir são detalhadas as descrições das atividades incluídas e adaptadas do fluxo de análise e projeto.

Descrição do Workflow de Análise e Projeto:

Atividade: Projetar Casos de uso

- Responsável: Analista de Sitemas;
- Entrada:
  1. Especificação dos requisitos de software;
  2. Modelo de Iteração;
  3. Realização dos Casos de Uso;
  4. Modelo de Arquitetura;
  5. Documento de Arquitetura de software;
- Passos:

1. Refinar as realizações de casos de uso, incluindo os elementos do projeto (dependendo das camadas da aplicação);
2. Incorporar a arquitetura necessária à implementação do caso de uso às realizações, ou seja, todas as classes surgidas na etapa anterior devem fazer parte agora dos diagramas de iteração de cada um dos casos de uso.

- Saída:
  1. Modelo de Iteração;
  2. Realização dos Casos de Uso;

Atividade: Projetar navegação

- Responsável: Web Designer;
- Entrada:
  1. Especificação dos requisitos de software;
  2. Storyboards de casos de uso;
- Passos:

usuário, tutoriais, entre outros) estão organizadas na aplicação;

3. Especificar a padronização dos elementos gráficos da aplicação mediante o padrão adotado pelo cliente;
4. Validar as definições dos casos de uso da aplicação a partir da especificação das interfaces do usuário;

- Saída:
  1. Modelo de padronização das interfaces do usuário;
  2. Especificação dos elementos de interface.

Em relação ao *workflow* acima citado, deve-se destacar que, apesar de o responsável por cada passo ser considerado de uma área distinta, neste projeto todos os passos foram realizados por uma mesma pessoa.

2.3.2. D.F.D – Diagrama de Caso de Uso.

Os casos de uso representam funções completas do produto. Um caso de uso realiza um aspecto maior da funcionalidade do produto: deve gerar um ou mais benefícios para o usuário. O conjunto de casos de uso deve descrever a funcionalidade completa do produto, sem lacunas e sem superposições; daí dizer-se que cada caso de uso representa uma “fatia de funcionalidade”.

Para o desenvolvimento do “website cfd-ib para o setup físico do solver fluids 3d de dinâmica dos fluidos computacional” temos os seguintes casos de uso:

- Gestão manual do sistema mantenedor do solver e do web site;
- Gestão Manual dos Usuários;
- Configuração do Setup físico do solver fluids-3d;
- Download de resultados.

Atores: Os papéis dos usuários do produto são modelados através dos atores. Cada ator apresenta uma classe de usuários definida, sendo assim temos como atores: Usuários de CFD da FEMEC e o Administrador do Sistema.

Abaixo segue os casos de uso do Administrador do Sistema e dos usuários:

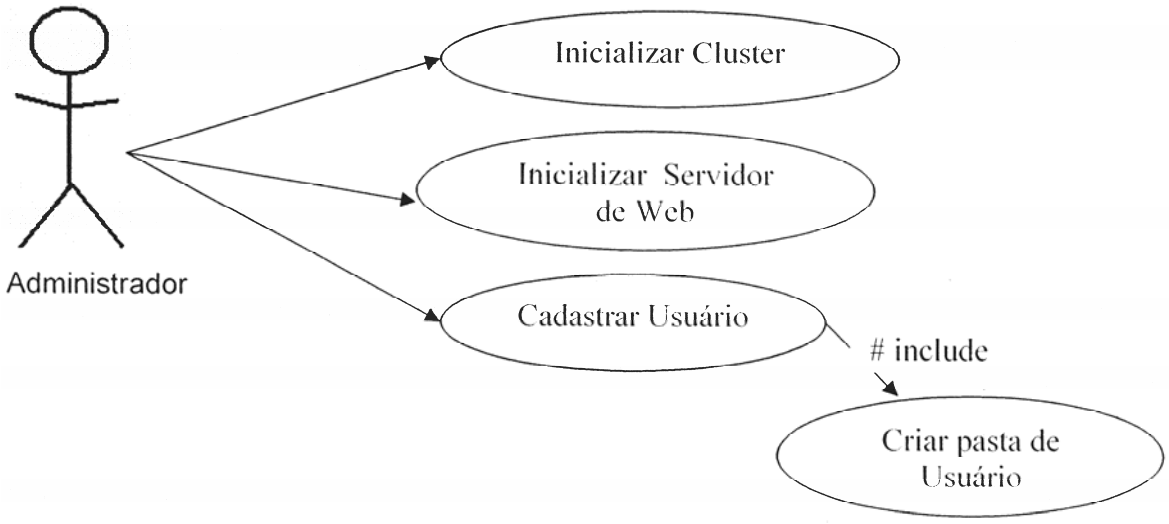


Figura 2.9 – Diagrama de Caso de uso do Administrador

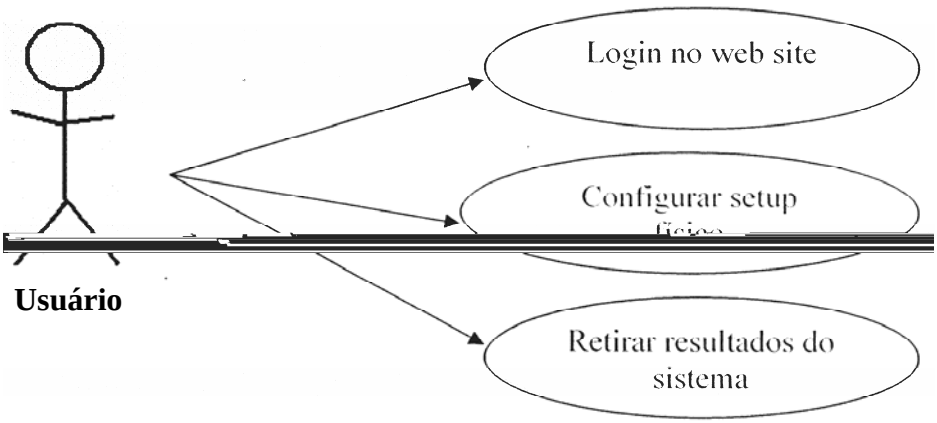


Figura 2.10 – Diagrama de Caso de uso do Usuário

Com os diagramas de caso de uso acima visualizados, pode-se concluir que cada um dos casos possui uma seqüência para o funcionamento correto do sistema.

Para o administrador existe a necessidade de o cluster do LTCM estar funcionando, com o serviço httpd (Servidor de Websites do linux) em execução. Para liberar acesso aos usuários o administrador deve efetivar o cadastro dos mesmos e criar pastas em todas as máquinas do cluster.

Para o usuário fica restrito somente o setup físico da simulação que ele deseja executar, além de abstrair os dados que lhe são de interesse.

2.3.3. D.F.D. – Diagrama de Fluxo de dados.

O D.F.D. é uma técnica usada na programação estruturada de diagramação de software que possui diversos tipos de diagramas, derivando-se em outros diagramas

subseqüentes.

Assim um D.F.D representa:

1. Imagem do sistema, projeto ou produto;
2. Modelo de organização;
3. Apresentação em etapas com aumento gradativo de detalhes;
4. Utilização dos princípios da modularização e da hierarquização.

Assim, podemos ter diversos níveis de D.F.D de forma a representar o fluxo de dados da aplicação.

a. D.F.D nível 0 – Apresenta uma visão clara do produto com todos os macro-processos, com entidades externas, fluxo de dados e depósito de dados principais.
b. D.F.D nível 1 – É uma expansão do nível zero com mais detalhes e mais completo incluindo o tratamento de exceções.

2.3.3.1. Simbologia do D.F.D.

A seguir temos uma das possíveis simbologias usadas na representação de D.F.D.:
Entidades Externas:

- São categorias lógicas de objetos ou pessoas que representam origem ou destino de dados, e que, acionam um sistema e/ou recebem informações;
- Podem ser pessoas, sistemas ou unidades departamentais;
- Possuem as seguintes regras:
    - X – letra para identividação;
    - Nome – nome da entidade: Ex.: Clientes, Sistemas de Acesso, Banco, etc.
- Como descobrir entidades externas?
    - No mínimo temos duas: quem usa o sistema (cliente) e quem opera o sistema (departamento A)

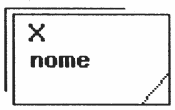


Figura 2.11 – Exemplo de Simbologia para Entidade Externa

Fluxo de dados:

- São o meio por onde os dados e as informações trafegam;
- Regras:
    - Nome – nome do dado. Ex.: Pedido, Nota Fiscal, Produto, Item, etc;

- Arg. – argumento de acesso a um deposito. Ex.: CGC, CPF, CEP, código, mátricula, etc.

Sempre envolvem processos não sendo possível o fluxo de entidade para entidade, entidade para depósito de dados, depósito de dados para depósito de dados.

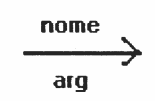


Figura 2.12 – Exemplo de Simbologia para Fluxo de Dados

Processos:

- Transformam fluxo de dados em uma atividade;
- São módulos do sistema;
- Regras:
  - N – número de referência do processo. Ex.: 0, 1, 2, 3, 1.1, 1.2
  - Função – descreve o processo, no verbo infinitivo. Ex: Cadastrar cliente, Gerar Arquivo, Imprimir Relatório, etc.
  - Loc – local físico onde se desenvolve o processo. Ex.: Almoxarifado, Contabilidade, etc.

Dica: para descobrir um processo relate os requisitos do sistema. (Cadastrar Cliente, Efetuar Logon, etc.)

Figura 2.13 – Exemplo de Simbologia para Processos

Depósito de Dados:

- São locais de armazenamento de dados;
- São arquivos físicos;
- Regras:
  - Dn – número do depósito. Ex: 0, 1, 2, 3, D1/1, D1/2.
  - Nome: nome do depósito. Ex.:Clientes, Produtos, Contas, etc.

Para tornar mais fácil identificar o diagrama de Depósito de dados, leva-se em conta dois tipos de arquivos: Cadastral e de Movimento (Movimento de itens, etc.).

Figura 2.14 – Exemplo de Simbologia para Entidade Externa

A fim de exibir os requisitos para o fluxo de dados, pode-se utilizar os diagramas de fluxo de dados. Assim, como acontece em algumas técnicas, à hierarquia é expressa em camadas, de tal modo que diferentes níveis de detalhes sejam mostrados em diferentes camadas. Para iniciar tal fase, deve-se considerar o sistema como um transformador de dados. O diagrama mostra os dados que fluem para dentro do sistema, como eles são transformados e como eles saem do sistema. A ênfase é sempre no fluxo de dados, não no fluxo de controle. Abaixo segue o Diagrama de Fluxo de Dados da Interface, referente ao projeto desta dissertação.

Ao analisar o diagrama da figura 2.15, pode-se visualizar a interação do usuário com cada um dos formulários da interface. Após se realizar a consulta ao banco de dados para validar o usuário no sistema, este irá passar informações para cada formulário que será visualizado e, caso haja algum erro, poderão ser corrigidas. Tal interação segue de maneira semelhante até o fim da configuração de uma simulação, com exceção de quando o usuário deve indicar arquivos de sua máquina para efetuar upload para o sistema. Após todos arquivos terem sido gerados e armazenados no Cluster, o usuário envia o comando de execução da simulação sobre os arquivos gerados, através de um botão no formulário final.

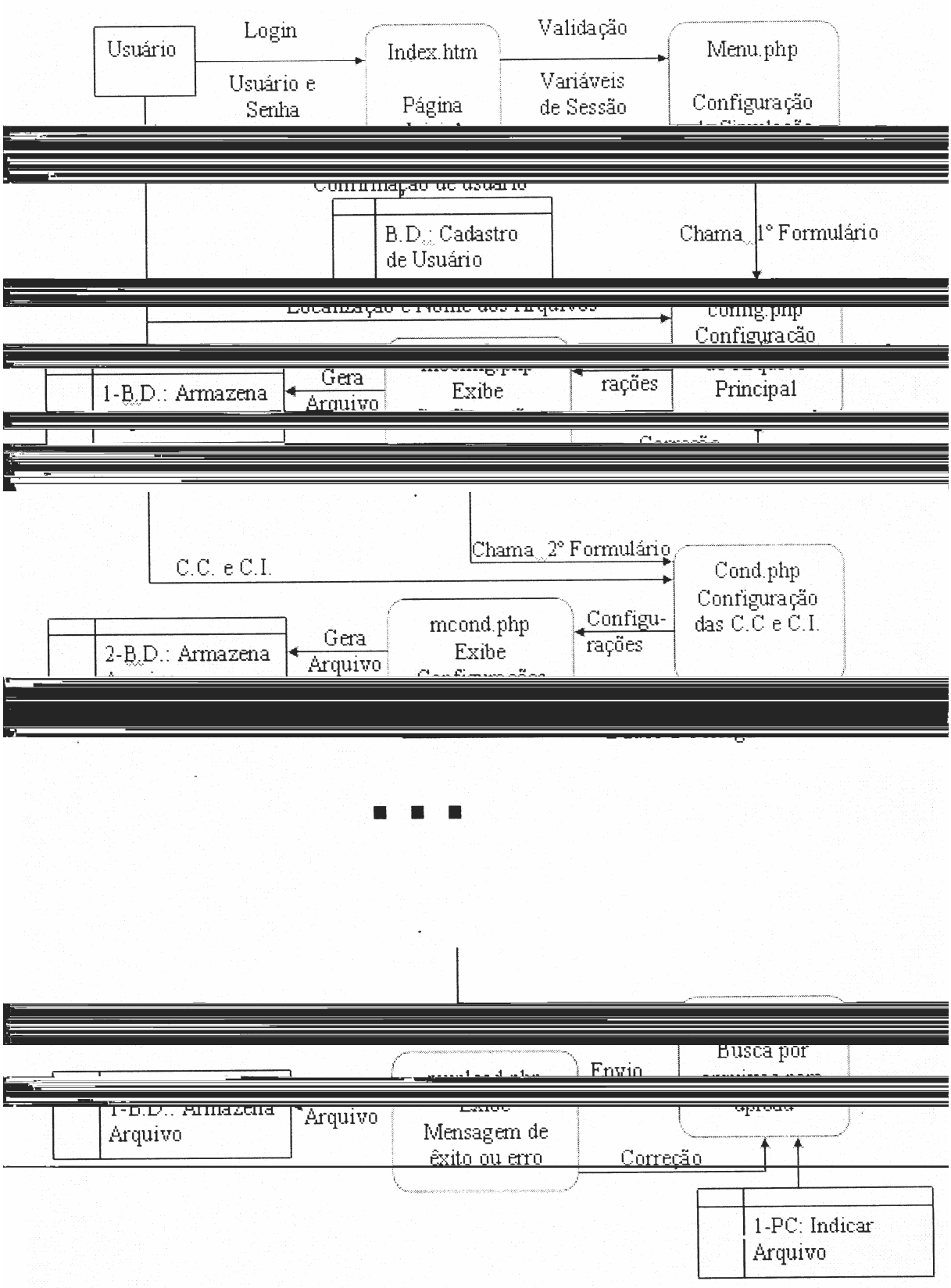


Figura 2.15 – Diagrama de Fluxo de Dados da Interface do atual projeto

2.4. Softwares Geradores de Malhas

2.4.1. Triangle

O TRIANGLE é um programa em plataforma C para geração de malhas Bidimensionais e construção de triangularização de Delaunay e Diagramas de Voronoi. Ele é rápido, com buffer robusto e capaz de calcular a triangularização de Delaunay armazenando a malha de forma exata.

As malhas de qualidade (sem ângulos agudos), são geradas usando o algoritmo de refinamento de Delaunay. Suas características incluem armazenamento dos ângulos na área do triangulo, buracos e concavidades especificados pelo usuário, além de uso econômico de aritmética exata para melhorar a robustez.

Diagrama de Voronoi é formado com base na seguinte regra: dadp i, conjunto de locais (pontos) num plano, associar a cada local a região do plano mais próximo dele do que de qualquer outro. Uma definição mais formal seria a seguinte: Dado um conjunto S de n pontos no plano, queremos determinar para cada ponto p de S qual é a região V(p) dos pontos do plano que estão mais próximos de p do que de qualquer outro ponto em S. As regiões determinadas por cada ponto formam uma partição do plano chamada de Diagrama de Voronoi, Figura (2.16). Como Geometria computacional não é objeto de estudo dessa dissertação, maiores informações sobre Diagrama de Voronoi e triangularização de Delaunay pode ser melhores explicadas por Toussaint, em http://cgm.cs.megil.ca/~godfried/teaching/cg-web.html.

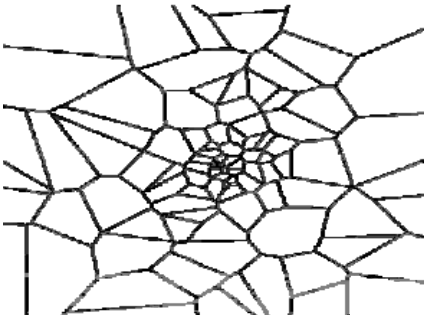


Figura 2.16. Exemplo de diagrama de Voronoi de um conjunto de vértices.

Uma triangularização de um conjunto de pontos consiste em encontrarmos segmentos de reta que conectam estes pontos de tal modo que nenhum desses segmentos cruze com nenhum outro e que cada ponto seja vértice de pelo menos um triangulo formado

por esses segmentos. Esses segmentos particionam o conjunto de pontos em triângulos, daí o nome de Triangularização.

O grafo dual de um Diagrama de Voronoi constitui uma triangularização cujos pontos são os pontos construtores do diagrama de Voronoi. A esta triangularização particular dá-se o nome de Triangularização de Delaunay. De fato existe um algoritmo que, dado um Diagrama de Voronoi, obtém-se a Triangularização de Delaunay em tempo linear. Outro algoritmo faz o serviço inverso, também em tempo O(n). Porém, existem algoritmos que produzem a Triangularização de Delaynay diretamente do conjunto de pontos.

Um Triangulo da Triangularização de Delaunay tem a seguinte propriedade: ele determina um círculo cujo interior não contém nenhum outro ponto de conjunto de pontos a não ser os três pontos que determinam o triângulo, mostrado na figura 2.17

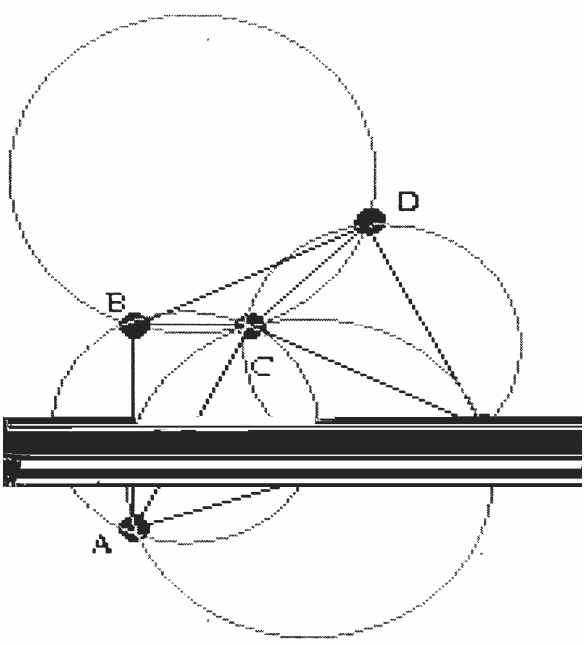


Figura 2.17 – Exemplo de triangularização de Delaunay de um conjunto de vértices.

Apesar de o *Triangle* ser um software de plataforma linux, ele se torna desqualificado para nossos objetivos, pois:

- Trabalha somente com malhas bidimensionais, sendo, portanto, impossível de gerar a malha os objetos de interesse.
- Apesar de ser um software robusto criado em plataforma C, o programa não é portável, não sendo passível de uso na plataforma Windows.

2.2.2. NETGEN

NETGEN é uma ferramenta geradora de malha para duas ou três dimensões e possui código fonte aberto. Ele nos oferece um padrão de programa com uma interface gráfica de usuário, ou uma Biblioteca C++ de ligação com outros aplicativos. NETGEN é avaliável para Unix/Linux e Windows98/NT.

NETGEN gera malhas triangulares ou quadrilaterais em 2D e malhas tetraédricas em 3D. A entrada para 2D é descrita por splines curvas e a entrada para problemas 3D é qualquer uma definida pela geometria de sólidos construtivos ou em arquivos de formato padrão STL. Esta ferramenta possui módulos para otimização de malhas e refinamento hierárquico de malhas.

Apesar de ser um software de plataforma linux e ser portável para plataforma Windows, existem alguns motivos pelos quais o NETGEN não será utilizado para nossos objetivos:

- Sua malha 3D é de formato tetraédrico, sendo que na metodologia da Fronteira Imersa restringe-se ao uso de malha triangular;
- Pode-se sugerir a malha interna para representar a geometria, porém, para utilização na Fronteira Imersa, o conhecimento necessário que se deve ter sobre a geometria é somente sua superfície (casca), não fazendo sentido malhar o volume e carregar o programa com informações desnecessárias.

Porém o NETGEN é citado como ótimo pacote auxiliar de outros softwares geradores de malha, funcionando como ferramenta suporte na geração de malhas 3D.

2.2.3. GMSH

GMSH é um gerador automático de malhas tridimensionais em elementos finitos, primariamente de Delaunay, com facilidades de pré e pós-processamento. Ele foi desenvolvido para oferecer uma ferramenta simples para resolver problemas acadêmicos com entradas paramétricas e capacidade avançada de visualização. Uma de suas características é que um comprimento é preservado na geração de malhas adaptativas em linhas, superfícies e volumes, ou seja, ao se estabelecer um tamanho característico para o elemento da malha (triângulo), ele permanecerá fixo durante todo o processo de malhagem,

até a geração do arquivo STL (casca da geometria). Ele permite também mixar estas malhas em uma simples estrutura.

GMSH é composto de quatro módulos: geometria malha, solver e pós-processamento. Todas essas funções são prescritas usando a interface gráfica de usuário (GUI) ou em arquivos de dados ASCII usando a própria linguagem de script do GMSH. Ações interativas geram bits de linguagens em arquivos de saída e vice-versa. Isto torna possível automatizar todo tratamento, utilizando loops, chamadas de sistemas externos e condicionais.

Geometrias são criadas pela definição sucessiva de pontos, linhas orientadas, superfícies orientadas e volumes, compondo grupos de entidades geométricas que podem ser definidas, baseados nestas entidades geométricas. A linguagem script do GMSH permite que toda entidade geométrica seja bem parametrizada.

Malha de elementos finitos é a tecelagem de um dado subconjunto de espaços tridimensionais por elementos de várias formas geométricas elementares (linhas, triângulos, quadrados, tetraedros, prismas, hexaedros, e pirâmides), arranjados de tal forma que se duas destas formas se interceptam, eles formam uma face, uma borda ou um nó. A geração de malha segue a mesma performance da criação de geometria: primeiro discretizamos as linhas e a malha das linhas são usados para malha das superfícies. Portanto, neste processo, a malha de uma entidade é somente construída pela malha de seus limites. Maiores detalhes sobre como utilizar o GMSH pode ser encontrado em anexo I, Tutorial do GMSH.

Solvers externos podem ser inter-relacionados com GMSH através do Unix ou sockets TCP/IP, o que permite encaminhar facilmente cálculos externos para coletar e aproveitar resultados de simulação dentro do módulo de pós-processamento do GMSH.

Exemplos de pós-processamento podem ser carregados e manipulados nos módulos geometria e malha. Funções de pós-processamento incluem seção de cálculos computacionais, offset, elevação, extração de componentes e limites, mapas de cores e suas variações, animação, vetores gráficos de saída, etc. Todas opções de pós-processamento podem ser acessadas pelo modo interativo ou através de entrada de arquivo texto ASCII.

GMSH foi considerado o mais aproximado dentre os geradores de malha para o presente trabalho, por possuir as seguintes características:

- Descrição simples e rápida de geometrias, graças às funções de usuário definidas, *loops*, funções condicionais e bibliotecas;
- Parametrização de geometrias. A linguagem script do GMSH faz com que todos os comandos e seus argumentos dependam de cálculos prévios;

- Gera com simplicidade 1D, 2D e 3D (Figura 2.5);
- GMSH oferece vários mecanismos para controlar o tamanho dos elementos na malha final: através de comprimento característico de pontos geométricos;
- Cria exportações simples de geometrias e malhas;
- Interage com *solvers* externos. GMSH oferece interfaces C, C++, Perl, Python, e outras que podem ser facilmente adicionadas;
- Visualização de resultados computacionais de grande variedade;
- Exporta plotagem em diferentes formatos: Látex, PNG, JPEG, STL e vários outros;
- Gera animações complexas;
- Roda em várias plataformas (Windows, Mac e Unix), sendo gratuito.

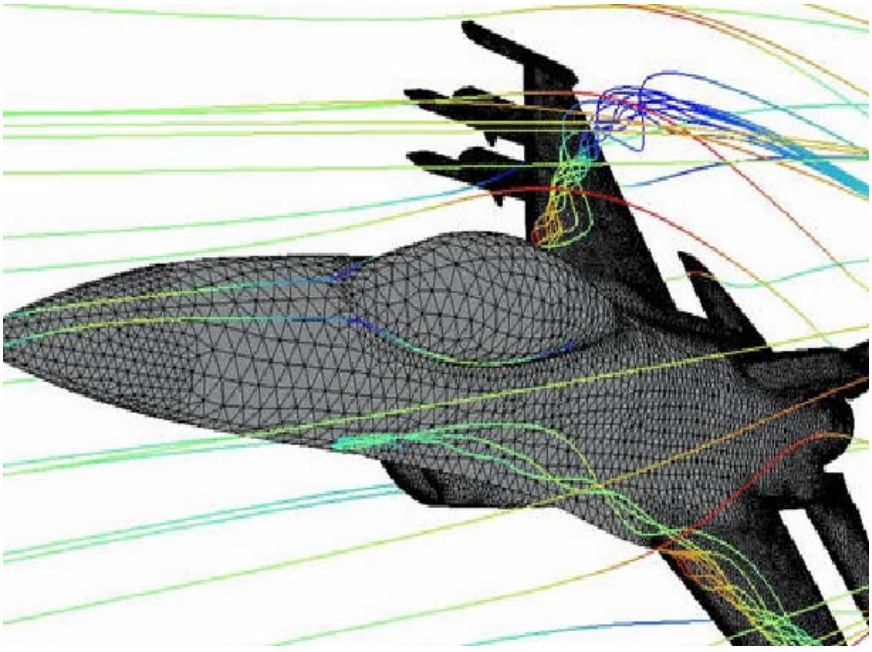


Figura 2.18 – Exemplo de malha 3D gerada pelo GMSH.

2.4. LINGUAGENS DE PROGRAMAÇÃO

Como nossa principal preocupação, de selecionar um gerador de malha que acatasse as características necessárias para resolver os problemas físicos foi resolvida, resta escolher uma ferramenta de programação capaz de dar suporte à ferramenta geradora de malha escolhida, para que possamos desenvolver uma ferramenta capaz de efetuar os cálculos necessários para problemas de interação fluido-estrutura.

Para tanto, é necessário ressalvar que, para desenvolver uma interface de uso amigável e de boa aparência, os melhores tipos de linguagem de programação, são aquelas

que trabalham com desenvolvimento de aplicações visuais de interface com o usuário. Para isso, tal ferramenta deve se valer de uma biblioteca de componentes visuais (VCL – *Visual Component Library*) consistindo de objetos reutilizáveis, incluindo objetos padrões de interface com o usuário, gerenciamento de dados, gráficos e multimídia, gerenciamento de arquivos e quadros de dialogo padrão.

Como as aplicações terão que ser realizadas na plataforma linux, pode-se descartar facilmente a linguagem Delphi, que apesar de se enquadrar nas características acima citadas, esta linguagem encontra-se disponível somente para Windows, além de não ser uma ferramenta gratuita. Com isso, uma opção que se apresenta é o Kylix, que é um Delphi criado para plataforma Linux, porém seus direitos autorais são propriedade da Borland, o mesmo proprietário do Delphi, sendo assim, Kylix não é uma ferramenta livre.

Na busca por ferramentas visuais que tenham como plataforma o sistema operacional Linux, foram encontradas algumas ferramentas que apesar de possuírem boas qualificações, não preencheram alguns dos requisitos citados, dentre eles, ser uma ferramenta livre. Dentre as ferramentas freeware encontradas a princípio, teve-se uma tendência pela utilização do QT Designer.

2.4.1. QT

QT é uma aplicação compreensiva para desenvolvimento de *framework* em C++. Isto inclui uma biblioteca de classes e ferramentas para o desenvolvimento e migração de plataformas.

Sua biblioteca de classes possui cerca de mais de 400 classes C++, a qual encapsula toda infra-estrutura necessária para o desenvolvimento de aplicações fim-a-fim. O QT API inclui um modelo de objeto amadurecido, uma rica coleção de classes e funcionalidade para programação GUI (Interface Gráfica de Usuário), layout, programação de base de dados, redes, XML, integração com OpenGL.

Qt possui uma ferramenta gráfica chamada QTDesigner que é uma construtora de *layout* e *forms* com características GUI, a qual possibilita o desenvolvimento rápido de interfaces de usuários com alta performance com visual nativo e suportado em várias plataformas (Windows, Linux/Unix, Mac OS X), sendo esse um dos motivos pelo qual esta ferramenta foi estudada como possibilidade para ser aplicada ao presente projeto. A portabilidade é um objetivo muito importante, pois poderá tornar a futura ferramenta para a solução de problemas de interação fluido-estrutura, acessível para o público alvo, fazendo com que o Sistema Operacional não seja um obstáculo para divulgação desta ferramenta, já que para se obter a mesma, basta recompilar o código no sistema operacional desejado.

Por si só, ou integrado com outras ferramentas como Microsoft Visual Studio.NET, o QTDesigner inclui características como modulo de pré-visualização, layout automático dos *widgets* (caixas de diálogo GUI), suporte para customização dos *widgets* e avançadas propriedades de edição.

QTDesigner torna fácil a visualização do design avançado da interface de usuário – a qualquer hora, pode-se gerar o código produzir ou pré-visualizar sua interface, mudando e ajustando o design de seu projeto de acordo com seu interesse.

Apesar de todas características que o qualificam, o QTDesing é uma ferramenta de programação orientada a objeto, e como citado anteriormente, os objetos de uma interface como botões e outras estruturas deste tipo, são prontamente anexadas como objeto da linguagem. Como o código do LTCM já se encontra funcional e validado, o ideal então é utilizar ferramentas de classe tipo UIMS para não modificar a estrutura do código atual.

O código computacional para o qual se criará a interface está em constante atualização, módulos novos estão sendo criados, e à medida que isso acontece a interface tem que ser modificada. Portanto, recomenda-se optar por utilizar linguagens e ferramentas de mais simples aprendizado, para que no decorrer do tempo a interface possa continuar a ser implementada por outras pessoas, haja visto que a interface esta sendo criada na área de engenharia mecânica, a qual geralmente não foca sua formação em desenvolvimento de software.

Além desse fator que o desqualifica, uma questão muito importante a ser mencionada, é que para utilizar o QtDesign, juntamente com a interface, ainda seria necessário implementar um código para efetuar uma conexão segura via rede com o servidor do Cluster no LTCM, sendo que para este tipo de software é necessário um teor computacional muito complexo, além de quase não se encontrar referências sobre tal assunto, que por si só exigiria uma dissertação.

2.4.2. PHP & DREAMWEAVER

PHP (um acrônimo recursivo para "PHP: Hypertext Preprocessor") é uma linguagem script de código fonte aberto (*Open Source*) de uso geral, muito utilizado, e especialmente guarnecida para o desenvolvimento de aplicações Web embútivel dentro de códigos fonte de linguagem HTML. Como todo código gerado em PHP é “embutido” no código HTML, o servidor WEB fica responsável por interpretar o código, como mostrado na figura 2.19 abaixo, e transformá-lo nas páginas que serão vistas pelos usuários. Deste modo, a conexão com o servidor do LTCM deixa de ser um problema complexo, se o

desenvolvimento do código for realizado nesta linguagem.

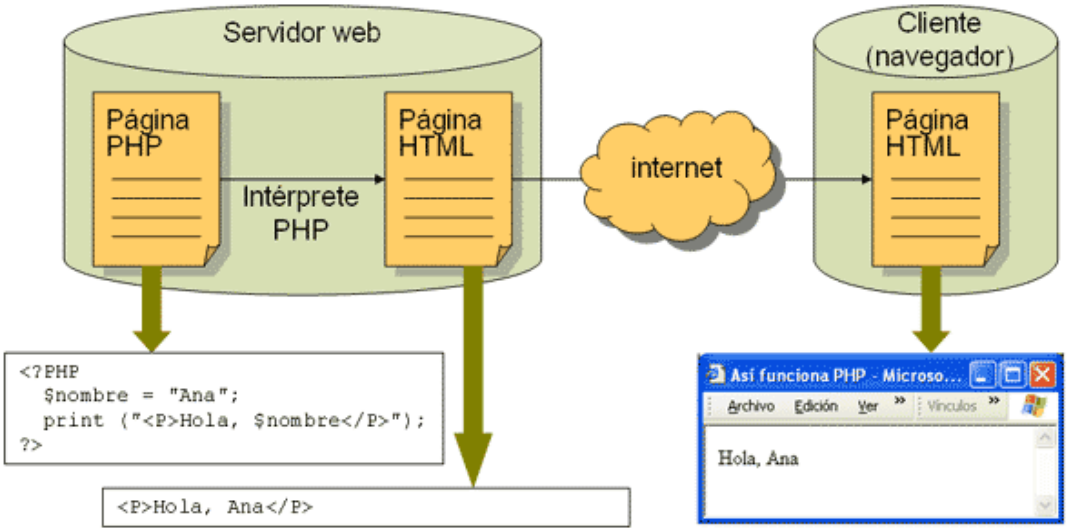


Figura 2.19. Funcionamento do PHP.

O funcionamento de PHP é simples: o usuário faz a solicitação da página; o servidor *Web* busca a página e interpreta o código; o usuário visualiza a página HTML.

O cluster do LTCM possui uma ferramenta instalada no servidor chamada Ganglia, que é um sistema de monitoramento distribuído, utilizado para sistemas computacionais de alta performance. Tal ferramenta reporta características como uso de memória nos nós do cluster, espaço livre em disco, quantidade de processos em execução por máquina, etc. Por ser implementado em PHP, para o seu funcionamento o Ganglia necessita de uma plataforma de servidor de páginas, para trocar mensagens entre as máquinas nós e a máquina servidor, além de disponibilizar estas informações em uma página acessível pela *internet*.

Como o servidor do cluster do LTCM oferece uma estrutura de servidor de páginas que é utilizada pelo GANGLIA, todo requisito necessário para desenvolver um código em PHP já se encontra disponível. Com esta estrutura já disposta no servidor, a conexão com o software se torna mais fácil, dependendo apenas do funcionamento da *web*.

Note como isso é diferente de scripts CGI escritos em outras linguagens como Perl ou C, ao invés de escrever um programa com vários comandos para imprimir HTML, escreve-se um arquivo HTML com um código inserido para fazer qualquer operação (nesse caso, imprimir um texto). O código PHP é delimitado por caracteres iniciais e finais, chamados de *tags*, que lhe permitem movimentar para dentro e para fora do "modo PHP".

O que distingui o PHP de algo como Javascript, no lado do cliente, é que o código é executado no servidor. Tendo-se um script similar, como mostrado na figura acima, em seu

servidor, o cliente receberia os resultados da execução desse script, sem nenhum modo de determinar o que é código fonte e o que não é. Pode-se inclusive configurar um servidor para processar todos os arquivos HTML como PHP, sem se identificar o tipo de linguagem utilizada.

A melhor coisa em usar PHP está no fato de ser extremamente simples para um iniciante, mas oferece muitos recursos para o programador profissional, podendo começar a escrever scripts em poucas horas.

Apesar do desenvolvimento do PHP ser focado nos scripts do lado do servidor, pode-se fazer muito mais com ele. O PHP pode ser utilizado na maioria dos sistemas operacionais, incluindo Linux, várias variantes Unix (incluindo HP-UX, Solaris e OpenBSD), Microsoft Windows, Mac OS X, RISC OS, dentre outros. O PHP também é suportado pela maioria dos servidores web atuais, incluindo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape and iPlanet Servers, Oreilly Website Pro Server, Caudium, Xitami, OmniHTTPd, e muitos outros. O PHP pode ser configurado como módulo para a maioria dos servidores, e para os outros como um CGI comum. Tais características solucionam o problema da portabilidade, facilitando o uso de qualquer aplicação.

Com o PHP, portanto, tem-se a liberdade para escolher o sistema operacional e o servidor web. Do mesmo modo, pode-se escolher entre utilizar programação estrutural ou programação orientada a objeto, ou ainda uma mistura deles. Mesmo sem todos os recursos da POO (Programação Orientada a Objetos) implementados no PHP 4, muitas bibliotecas de código e grandes aplicações (incluindo a biblioteca PEAR) são escritas somente em código POO. O PHP 5 corrige as deficiências da POO do PHP 4, e introduz um modelo de objetos completo.

Com PHP, não se está limitado a gerar somente HTML. As habilidades do PHP incluem geração de imagens, arquivos PDF e animações Flash (utilizando libswf ou Ming) criados dinamicamente. Pode-se escrever qualquer padrão texto, como XHTML e outros arquivos XML. O PHP pode gerar esses padrões e os salvar no sistema de arquivos, em vez de imprimí-los, formando um cache dinâmico de suas informações no lado do servidor.

Uma das características mais significativa do PHP é seu suporte a uma ampla variedade de banco de dados. Escrever uma página que consulte um banco de dados é bem simples. Os seguintes bancos de dados são atualmente suportados (Tabela 2.1):

Adabas D	InterBase	PostgreSQL
dBase	FrontBase	SQLite
Empress	mSQL	Solid

FilePro (read-only)	Direct MS-SQL	Sybase
Hyperwave	MySQL	Velocis
IBM DB2	ODBC	Unix dbm
Informix	Oracle (OCI7 and OCI8)	
Ingres	Ovrimos	

Tabela 2.1. Bancos de Dados suportados por PHP

PHP oferece uma abstração de banco de dados DBX permitindo-se utilizar qualquer banco de dados transparentemente com sua extensão. Adicionalmente, o PHP suporta ODBC (*Open Database Connection*, ou Padrão Aberto de Conexão com Bancos de Dados), permitindo-se utilizar qualquer outro banco de dados que suporte esse padrão mundial, toda essa iteratividade com vários tipos de banco de dados permite armazenar várias informações referentes aos projetos executados no cluster para que se possa ter informações para pós-processamento.

O PHP também tem suporte para comunicação com outros serviços utilizando protocolos como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (em Windows) e outros. Pode-se abrir *sockets* de rede e interagir diretamente com qualquer protocolo, o que caso necessário, facilita a iteração entre as máquinas do cluster. PHP também suporta o intercâmbio de dados complexos, utilizado virtualmente em todas as linguagens de programação para *web*. O PHP permite implementar a instanciação de objetos Java e os utiliza transparentemente como objetos PHP.

O PHP é extremamente útil em recursos de processamento de texto, do POSIX estendido ou expressões regulares Perl até como interpretador para documentos XML. No processamento de XML, o PHP 4 suporta os padrões SAX e DOM, além de se poder utilizar a extensão XSL para transformar documentos XML. O PHP 5 padroniza toda a extensão XML a partir da base sólida da libxml2, além de estender os recursos com o acréscimo ao SimpleXML e XMLReader. Todas essas formas de trabalhar arquivos textos são muito úteis, uma vez que a maioria dos arquivos a serem utilizados para simular um caso de mecânica dos fluidos, são arquivos textos.

Por último, têm-se também outras extensões interessantes: funções para Gateway, vários utilitários de compressão (gzip, bz2), calendário e conversões de datas, tradução, etc.

Como se pode ver, os recursos e benefícios que o PHP pode oferecer são muitos, porém uma questão ainda não solucionada para utilizar esta ferramenta de programação tão potente, é como resolver o problema da criação da interface gráfica. Para tal problema uma solução simples e de fácil acesso é o DREAMWEAVER.

O DREAMWEAVER MX é um editor de HTML profissional para desenhar, codificar e desenvolver *sites*, páginas e aplicativos para a *Web*. Os recursos de edição visual do DREAMWEAVER permitem criar páginas, de modo rápido, sem que seja necessário escrever uma única linha de código. É possível visualizar todos os elementos ou propriedades do *site* e arrastá-los de um painel fácil de usar diretamente para um documento.

Com isso finaliza-se a busca por todas as ferramentas necessárias para desenvolver a interface base para o código do LTCM (CFD-IB).

CAPITULO 3

Metodologia

Como mostrado no capitulo anterior, a busca pela ferramenta para auxiliar no desenvolvimento da interface gráfica, teve como resultado, não só um software de programação, mas também encaminhou esse projeto a um ambiente de desenvolvimento muito amplo conhecido como World Wide Web (W.W.W.) ou simplesmente Internet.

Humberto Eco (Veja Vida Digital, Dez/20001) escreveu sobre a Internet que "(...) pela primeira vez, a humanidade dispõe de uma enorme quantidade de informação a um baixo custo". Portanto, ao se dirigir para esse ambiente tão vasto de iteratividade, o ideal é fazer que as estruturas virtuais, chamadas *sites*, permitam auxiliar o meio acadêmico na área de CFD, rompendo as barreiras da distância entre o aluno e o laboratório LTCM, disponibilizando o conteúdo do solver para Dinâmica dos Fluidos Computacional pela Internet.

Embora a *internet* tenha surgido em 1992, as preocupações com interfaces, sugiram juntamente com o desenvolvimento dos softwares. A partir desse momento, o conceito de interface surgiu no campo da informática, com importância central para o campo da computação gráfica, multimídia, realidade virtual e teleconferência e fornece uma base sólida para ao *design* industrial e *design* gráfico (Bonsiepe, 1997).

A interface é uma superfície de uso viabilizada através do computador. Nela se opera em um terreno intermediário entre o *design* gráfico e o *design* industrial, priorizando as funções visuais, táteis e acústicas, de maneira que permitam a fácil inteligibilidade do usuário (Bürdek, 1999).

Dentre muitas definições, tem-se que design é o elemento fundamental para agregar valor e criar identidade gráfica para produtos, serviços e empresas. Alguns aspectos incorporados pelo *design* são a inovação, confiabilidade, evolução tecnológica, padrão estético, rápida percepção da função – uso de produtos, adequação às características sócio-econômicas e culturais do usuário. Peruzzi (1998) concluiu com isso que o *design* está intimamente ligado ao *marketing*, assim como à engenharia, ergonomia e outras facilidades

multifacetadas para gerar novos produtos e serviços.

3.1. Metodologias para Desenvolvimento de Sites e suas Interfaces

Cada designer tem uma metodologia própria e particular, mas todas condizem mais ou menos a mesma coisa, segundo Radfahrer (1999).

Dentre os primeiros estudos metodológicos destinados às interfaces gráficas, destacou-se o de Bürdek (1999), com uma metodologia geral. Antes de chegar nesta metodologia, Burdek exemplificou a diferença projetual entre o design tradicional e o design de interfaces, conforme a figura 3.1.

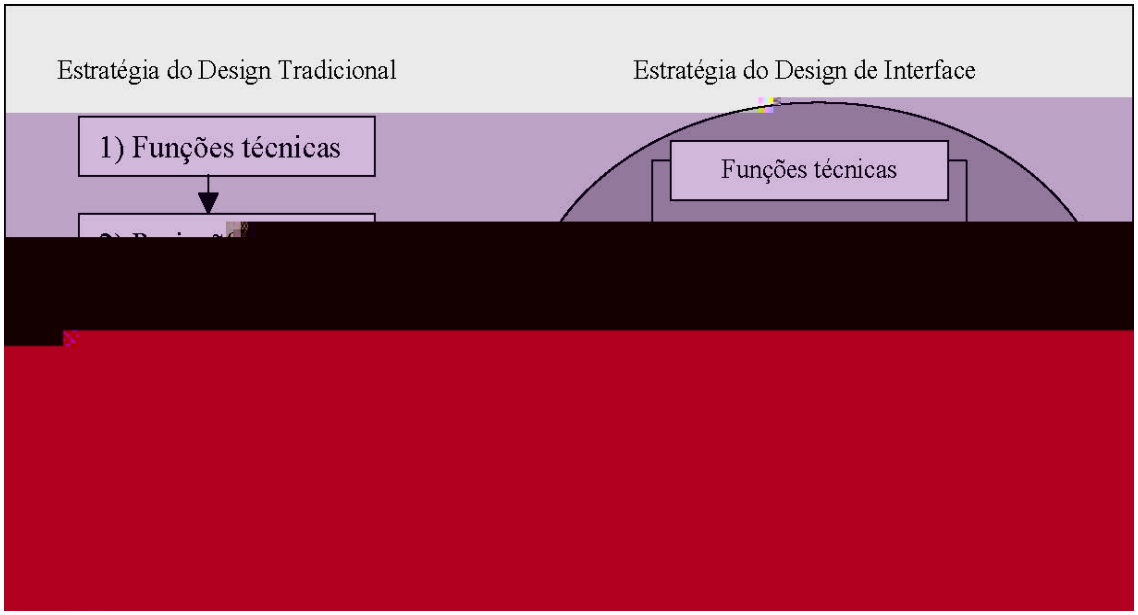


Figura 3.1. Novas estratégias projetuais (Bürdek, 1999).

A preocupação com o usuário em relação às interfaces serviu para levantar os seguintes questionamentos: como? por quem? em que contexto, entre outras – Bürdek (1999). Com isso destaca-se que o projeto é interdisciplinar e dependente do entorno em que se encontra o usuário. Com aquisição de maiores experiências, Bürdek (1999) definiu em 1990 sua metodologia geral, apresentada na figura 3.2.

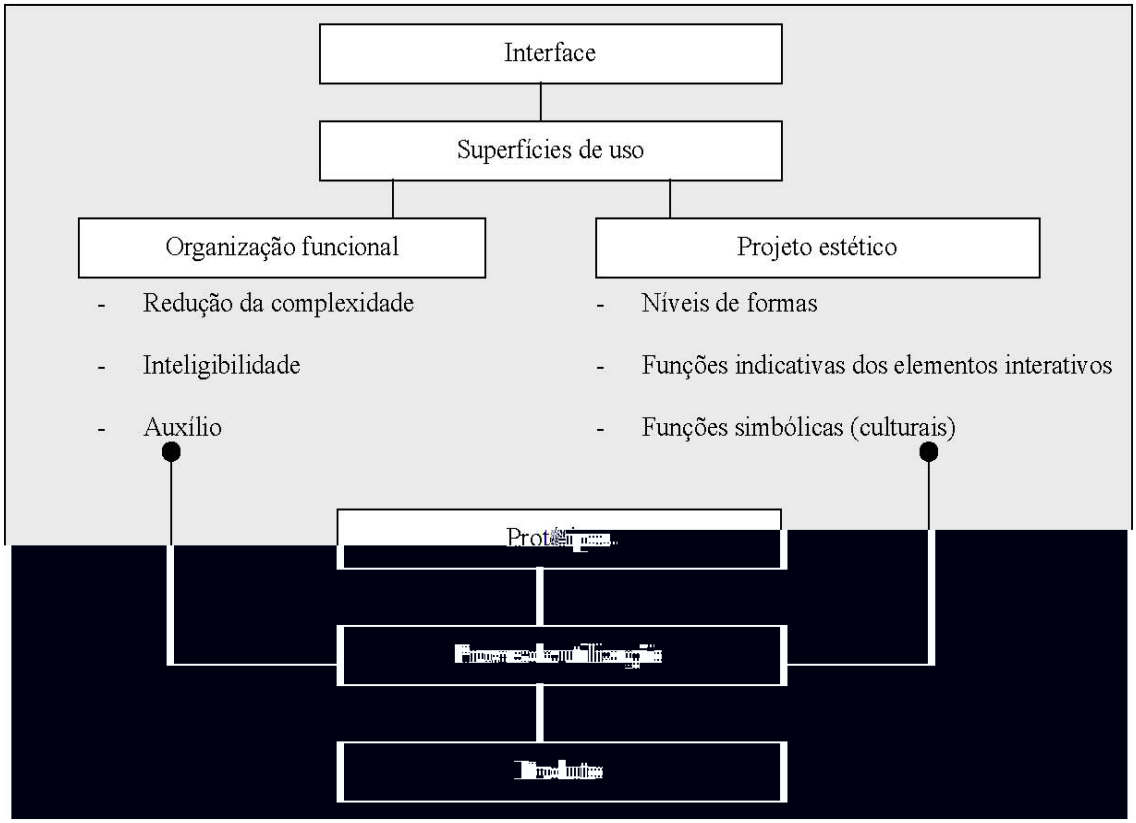


Figura 3.2 - Metodologia projetual para interfaces gráficas (Bürdek, 1999).

Afirma-se ainda que na WEB se criou um círculo vicioso, em que cada profissional cria sua própria metodologia conforme suas habilidades técnicas específicas (Radfhrer, 1999). Destaque para as metodologias de Clement Mok (1996) e Roger Black (1997), apresentadas nas figuras 3.3 e 3.4, respectivamente:

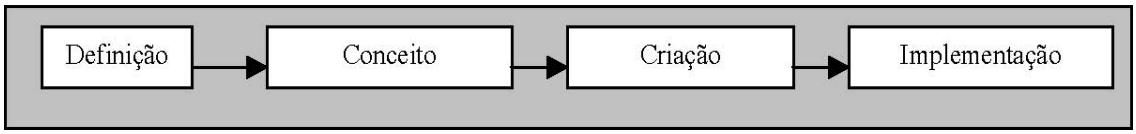


Figura 3.3 - Metodologia Projetual (Mok, 1996).

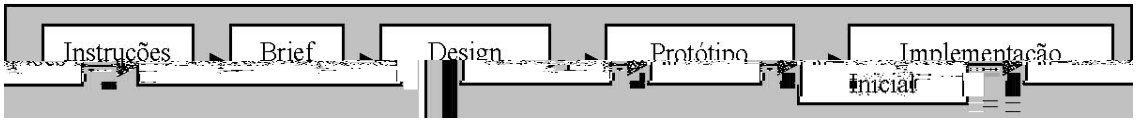


Figura 3.4 - Metodologia Projetual (Black, 1996).

As metodologias se apresentam organizadas de forma a deixar bem claro que, ambas são adaptações de metodologias projetuais de produtos e metodologias projetuais gráficas.

Diante dessa constatação, o designer brasileiro Luli Radfahrer (1999), apresentou sua metodologia projetual segundo sua definição: “(...) usamos uma mistura dos dois estilos (Mok e Black), com algumas sacadas do dia a dia e uma pitada de tempero (...)”. A figura 3.5 apresenta sua estrutura metodológica de criação e desenvolvimento de um site nos seguintes passos genéricos:

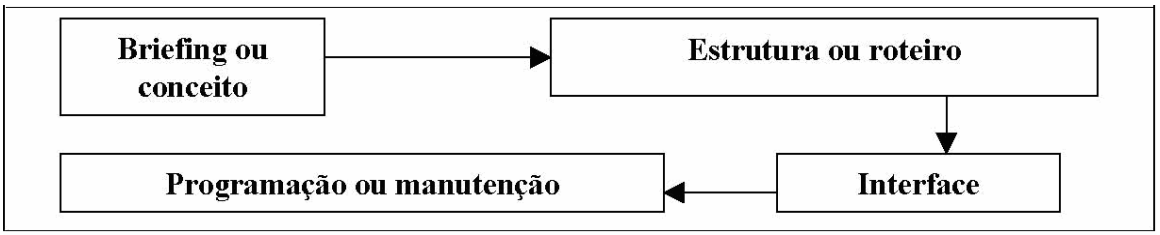


Figura 3.5 – Estrutura metodológica básica para projeto de sites (Radfahrer, 1999).

As etapas de projeto dentro da estrutura apresentada foram definidas da seguinte forma:

1. Desenvolvimento do conceito (comercial, criativo, técnico);
2. Especificações, planejamento e cronograma;
3. Organização dos grupos de informações e seu fluxo;
4. Direção de criação;
5. Produção de conteúdo;
6. Integração de conteúdo;
7. Programação e integração de software;
8. Teste e controle de qualidade;
9. Lançamento de marketing;
10. Manutenção;
11. Acompanhamento do usuário.

A disciplina “Comunicação no Ciberespaço” (*Communication in Cyberspace* – MIT, 2001), ministrada no Massachusetts *Institute of Technology* – MIT, acontece desde 1996 sob a orientação do Professor Edward Barrett. Nesta disciplina, foi desenvolvida uma metodologia própria (Barrett, 2001) que orienta futuros *web* designers e interessados na área, para a relação direta e importante existente entre o tipo de interface e a comunicação dos conteúdos presentes no *site* de forma original e complexa, enfatizando o uso de mensagem-chave e metáfora conceitual. A figura 3.6 demonstra a metodologia de Barrett (2001).

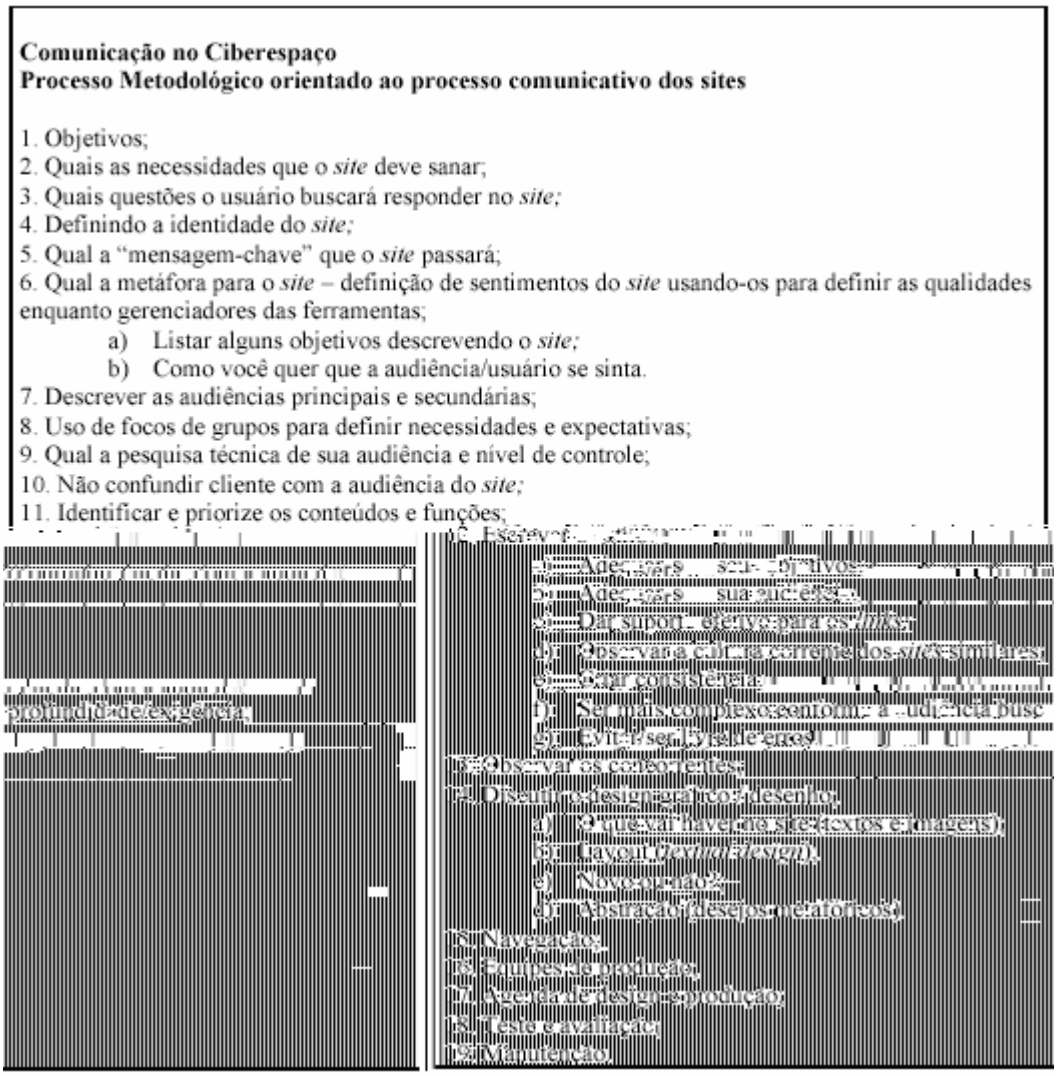


Figura 3.6. Processo metodológico para criação de *sites* (Barrett/ MIT-2001).

3.2. Justificativa para o modelo metodológico escolhido

Embora muitas empresas e *web* designers tenham suas metodologias próprias, a grande maioria utiliza, segundo levantamento bibliográfico, métodos com características lineares, como visto em Radfahrer (1999), Bürdek (1990), Mok (1996), Black (1997) ou Barrett (2001).

Metodologias simples e pobres são apresentadas na maioria dos livros que orientam processos de criação de *websites*, tentando evidenciar mais as linguagens de programação.

Diante das metodologias vistas no item anterior, pode-se analisar que, a maioria delas são derivações uma das outras ou complemento uma das outras, que objetivam suprirem falhas encontradas durante o processo de produção da interface gráfica na

individualidade de cada *site*.

Através da comparação feita, torna-se claro que a metodologia proposta e utilizada no MIT (Barret, 2001), expressam o conhecimento que há de mais atual e é utilizado no campo metodológico para *sites*. Tal conclusão se deve ao fato de o modelo metodológico abranger questões que, percorrem conceitos importantes no desenvolvimento gráfico para sanar problemas como: Informação e Comunicação, Ergonomia e Usabilidade, Marketing, Layout, dentre outros.

Além de essa metodologia possuir um procedimento compatível com a necessidade do LTCM, para se evitar erros comuns existentes em qualquer processo de desenvolvimento, faz-se necessária a reunião dos componentes do laboratório para analisar os pontos positivos e negativos do *site* em fase de criação e desenvolvimento.

3.3. Abordagem geral sobre a metodologia do projeto

Nesta abordagem, a metodologia em questão tem o intuito de orientar no desenvolvimento da interface, vinculando o desenvolvimento do site ao processo de criação, através da inserção de conhecimentos específicos da área de mecânica dos fluidos.

A organização destes conhecimentos será feita em tabelas de recomendações técnicas que aparece também como etapa bem caracterizada dentro de uma das fases do processo metodológico.

Como este trabalho não tem como objetivo desenvolver um processo metodológico para projetos de interface gráficas de CFD, esses limites foram rompidos ao se escolher ferramentas tecnológicas, referentes à programação e softwares de criação, editoração e design gráfico, como mostrado no capítulo anterior.

Deve-se salientar que, devido a complexidade exigida pelo teor de informações a serem dispostas no site, a metodologia é aplicada utilizando o auxilio específico de mestrandos e doutorandos, além do auxilio por parte do orientador, para suprir as necessidades na área de mecânica dos fluidos e correlacioná-la multi-disciplinarmente com a área computacional.

3.4. Aplicando o modelo metodológico

O modelo metodológico proposto usará simultaneamente dois métodos, sendo o

primeiro, o método da “caixa-preta” (método de segunda geração), e as demais fases da metodologia seguindo um processo linear através da manipulação direta de dados.

Com o método da caixa preta, as características complexas de manipulação de dados, onde os processos não são descritíveis, as fases e dados de manipulação para obtenção de resultados desejados, são omitidas. Esta etapa é composta do código de CFD existente no LTCM, o qual será vinculado à interface gráfica. O código é dividido em módulos, de forma que suas etapas dentro da “caixa preta” tenham uma sistemática linear tornando-se um método de primeira geração, como mostrado na figura 3.7 abaixo. Nesta fase, para definir algo elaborado, os dados de entrada passarão pelo processamento das informações no código de CFD do LTCM, após terem sido vinculado à interface gráfica elaborada.

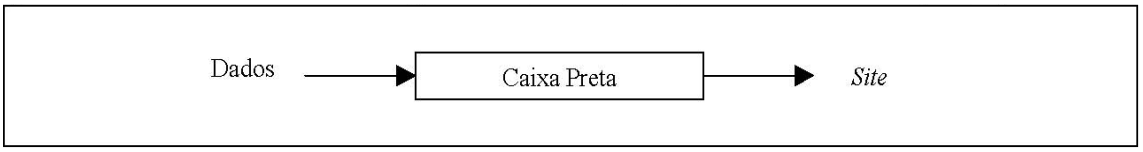


Figura 3.7. Método Caixa-Preta, adaptado para o modelo metodológico proposto.

O diferencial desta metodologia aparece na inserção da fase organizacional, onde conhecimentos específicos de mecânica dos fluidos são acrescentados a implementação da interface, através do aprendizado obtido na disciplina de Mecânica dos Fluidos, de forma a deixar a interface o mais organizada possível. Utilizando essas informações, dados serão passados da interface para o código, e este, terá informações suficientes para executar simulações computacionais em mecânica dos fluidos. Na figura 3.8, segue o fluxograma referente ao desenvolvimento da metodologia.

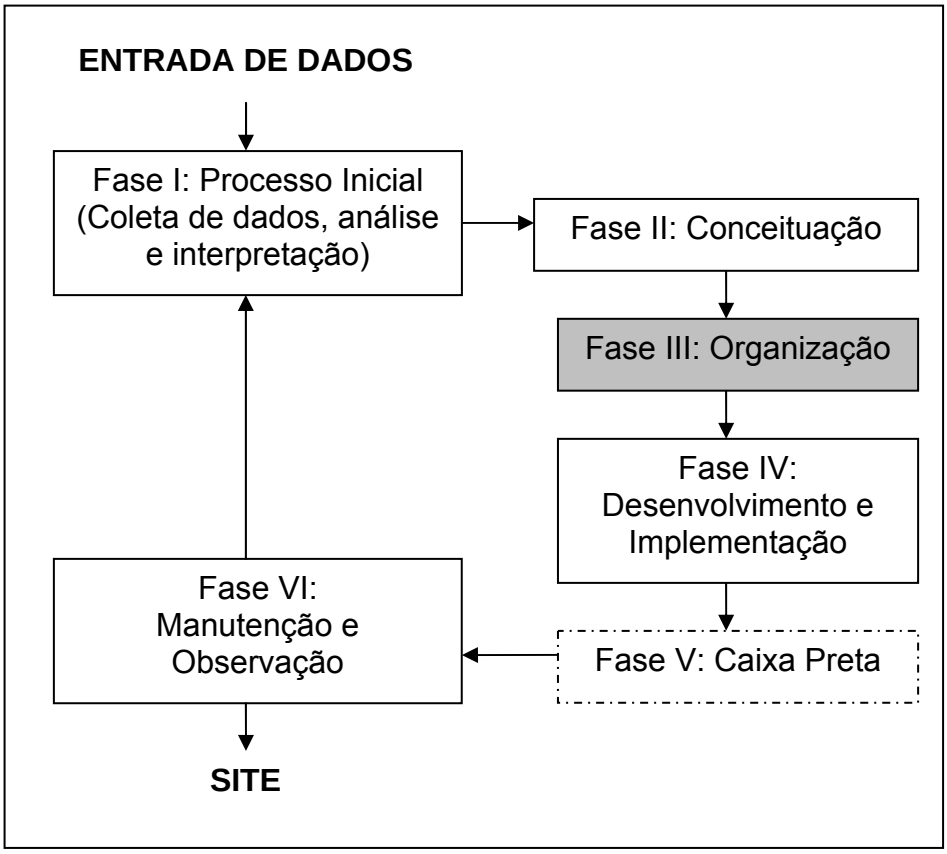


Figura 3.8. Modelo metodológico Proposto.

3.4. As fases da metodologia

Nesta seção serão explicitadas todas as fases da metodologia, esclarecendo em detalhes o que ocorre nas mesmas, seguindo o modelo metodológico proposto, abordando também as fases do processo metodológico de Barret (2001).

3.4.1. Fase I: Processo inicial (Coleta de dados do LTCM e dos usuários; análise e interpretação)

Tal fase é também conhecida como a fase de *briefing*, onde se resumirá a entrada de dados, através da coleta de informações sobre o código existente no LTCM e informações obtidas com os usuários do código. Este relacionamento de informações, laboratório – usuários, é feito justamente porque não se pode simplesmente construir um *site* baseado nos interesse único de uma das partes.

3.4.1.1.Informações sobre o código e Objetivo

Para estabelecer os objetivos que se pretende alcançar com essa interface gráfica, algumas informações foram coletadas. Dentre elas temos as seguintes informações, baseadas no estudo do código do laboratório:

1. Objetivos que se pretende alcançar com o *site*: O *site* proposto tem como objetivo principal sanar o problema de ausência da interface gráfica referente ao código disposto no LTCM. Para isso, o *site* será caracterizado como educacional visando auxiliar na evolução do aprendizado de mecânica dos fluidos, através da interatividade entre usuário (aluno) e pacote CFD;
2. Influência da WEB na audiência da interface: ao inverso de criar uma interface gráfica cujas bibliotecas padrões devem ser disponibilizadas por pacotes instalados na máquina local, viabilizando esta interface pela Internet, elimina-se a fase de instalação por parte do usuário, uma vez que o código central já se encontra validado e funcional, e elimina-se também as dificuldades referentes à conexão do usuário com o Cluster disposto no laboratório do LTCM;
3. Aspectos positivos: além de ser todo baseado em plataforma livre, gerando custo zero para o laboratório, a interface mantém um teor de escalabilidade, pois a medida que mais funções são acrescentadas ao código existente, a interface referente ao pacote pode facilmente ser implementada e vinculada a interface atual.
4. Organização do *site*: para facilitar o manuseio do *site*, aumentando sua usabilidade, a interface gráfica está disposta em etapas, assim como o código do LTCM, possibilitando ao usuário trabalhar de forma organizada.

3.4.1.2. A interface de acordo com interesse do usuário

A análise das informações contidas nesta seção tem como finalidade descobrir as necessidades do usuário, buscando conhecer o mundo dos mesmos e tentando vivenciar o que seriam suas atividades. Para tal abordagem, foram escolhidos alguns tópicos, contidos em um formulário (figura 3.9) de pesquisa para audiência de usuário proposto por Sterne (2003), para obter um caráter básico e conhecer a audiência do *site*, identificando os interesses dos usuários com intuito de mapear estas informações para análise e interpretação.

Sugestão de questionário a ser apresentado para possíveis usuários

(tem caráter básico de conhecer quem é a audiência do *site,* identificando seus interesses, com intuito de mapear as informações para análise e interpretação):

1. Quem são os usuário do site? (gênero, idade, profissão, características predominantes; tendências de gostos; etc.) ;
2. O que fazem, onde e como? - histórico, se necessário;
3. Quais as possíveis necessidades que este usuário pode ter/tem?
4. Que tipos de sites estes usuário costumam acessar? Quais as características destes *sites*, de que tipo são (homepages pessoais, portais, etc.) e quais os interesses que os levam a buscar estes sites? (Observar as características da organização dos conteúdos, da interface e de que forma são trabalhadas);
5. Quais as diferenças existentes, se houver, dentro da variedade do público-alvo?
6. Quais as vantagens para esse público usar a internet?
7. Quais as vantagens que o *site* do cliente poderia oferecer ao público-alvo?
8. Outras perguntas que possam ser interessantes para o cliente (usar a imaginação neste momento).

Figura 3.9. Questionário para audiência.

Partindo do principio que para ter acesso ao *site* de CFD, o usuário deve estar cadastrado no laboratório, uma seleção natural dos usuários é feita, levando-se em consideração que, o mesmo deve ser pelo menos aluno de graduação com conhecimento em mecânica dos fluidos. Além de tal restrição, existe ainda a limitação quanto ao Cluster do laboratório não ter número de máquinas suficientes para estar liberando acesso para uma grande quantidade de alunos.

Haja visto que para efetuar simulações em mecânica dos fluidos são necessárias ferramentas com poder computacional robusto e softwares apropriados, e como a maioria dos usuários não tem acesso a estes recursos, pode-se idealizar um perfil padrão de usuário.

Por se tratar de uma interface para uma ferramenta complexa que exige um conhecimento específico, uma outra abordagem importante é questionar quais as necessidades que os usuários têm, referente à mecânica dos fluidos computacional. Assim sendo, para solucionar tal questionamento, elaborada uma pesquisa dentro do laboratório, concluiu-se que a maior necessidade do usuário é, dispor de uma ferramenta computacional que permita o estudo da dinâmica dos fluidos, usando CFD para criar modelos

computacionais que representem um sistema ou dispositivo que se precisa estudar. Com isso, dando as propriedades físicas do fluido e aplicando-as a um protótipo virtual; a metodologia CFD deve fornecer resultados de predição do comportamento do escoamento.

De acordo com as necessidades dos usuários acima citadas, pôde-se verificar que são poucas as interfaces via *web* que disponibilizam esse tipo de ferramenta, um dos poucos *sites* que se conseguiu localizar foi o CFDnet, o qual é usado através da Internet, baseado na interatividade do usuário, que propõe resolver rotinas de visualização da solução, resolvendo problemas práticos da engenharia no escoamento de fluidos. Assim como o proposto por este projeto, no CFDnet nenhum *downloads* ou encaixe do software são requeridos (Alé e Romero, 2003).

Apesar de CFDnet ser um pacote intuitivo, o mesmo possui restrições que, se comparado ao código existente no LTCM, faz com que o CFDnet deixe a desejar em alguns pontos. Dentre eles temos a problemática de que CFDnet trabalha somente em 2D, como mostrado na figura 3.10 abaixo. Apesar de se poder desenhar o canal pelo qual o escoamento passará (podendo representar uma geometria de forma complexa), este pacote fica restrito a definição dos blocos geométrico do grid, sondas e não se pode fazer um *upload* de geometrias, mesmo que 2D, como mostrado na figura 2.11. O maior atrativo explicitado no CFDnet é a possibilidade de realizar pós-processamento dos resultados do escoamento configurado (figura 3.12), o que no caso da interface projetada para o LTCM, foi deixado para estudos e implementações futuras.

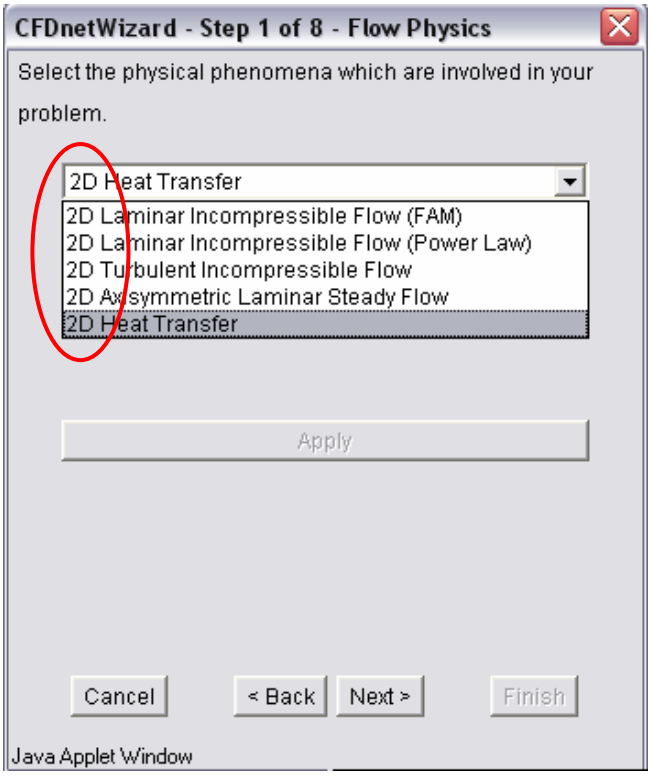
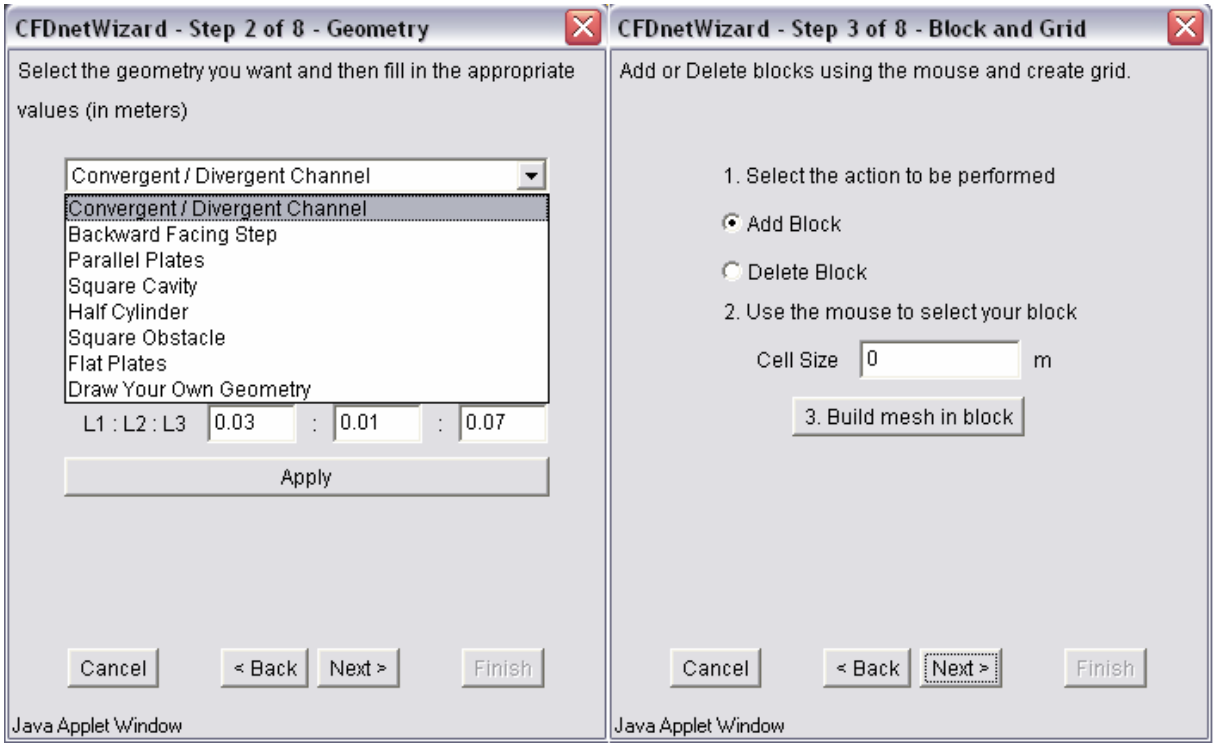
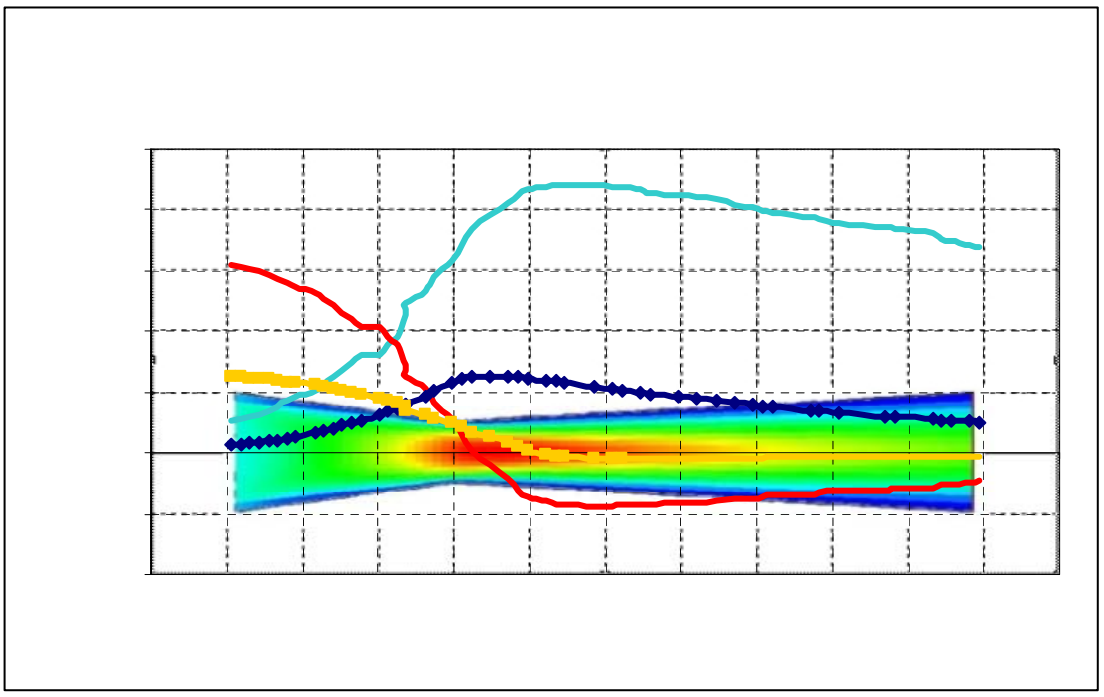


Figura 3.10. Escolha do tipo de escoamento e problema físico.



(a) (b)

Figura 3.11 – a)Criação da geometria do canal de escoamento do Fluido; b) Inserção de blocos.



tipo de solução, com o pesar de o mesmo ter algumas restrições, a solução proposta no presente projeto torna-se um passo importante para sucesso de tal interface gráfica, visto que uma das principais vantagens em relação a outras ferramentas de CFD é que, nenhum *download* ou encaixe do software são requeridos. O projeto disponibilizará uma versão livre via *Web*, tendo seu acesso facilitado, e não se restringirá a geometrias simples no domínio 2D.

3.4.2 – Fase II: Conceituação da Interface

Com os objetivos bem definidos, define-se nesta fase o tipo de *site* e o conceito do mesmo. Este irá orientar, na fase seguinte, a organização dos conteúdos e o planejamento da interface através da(s) mensagem(s) que se pretende transmitir, permitindo o processo de comunicação entre o usuário e a interface.

Para conceituar o *site*, todas as informações anteriores, relativas ao código de CFD e os usuários, devem ser utilizadas. Apoiando-se na metodologia criada por Barrett - MIT (2001), visualizada na figura 3.13, os objetivos claramente delimitados enunciam uma lista de necessidades reais que devem ser atendidas pelo *site* - que irão definir sua identidade - possibilitando a geração de uma (ou mais de uma) mensagem-chave.

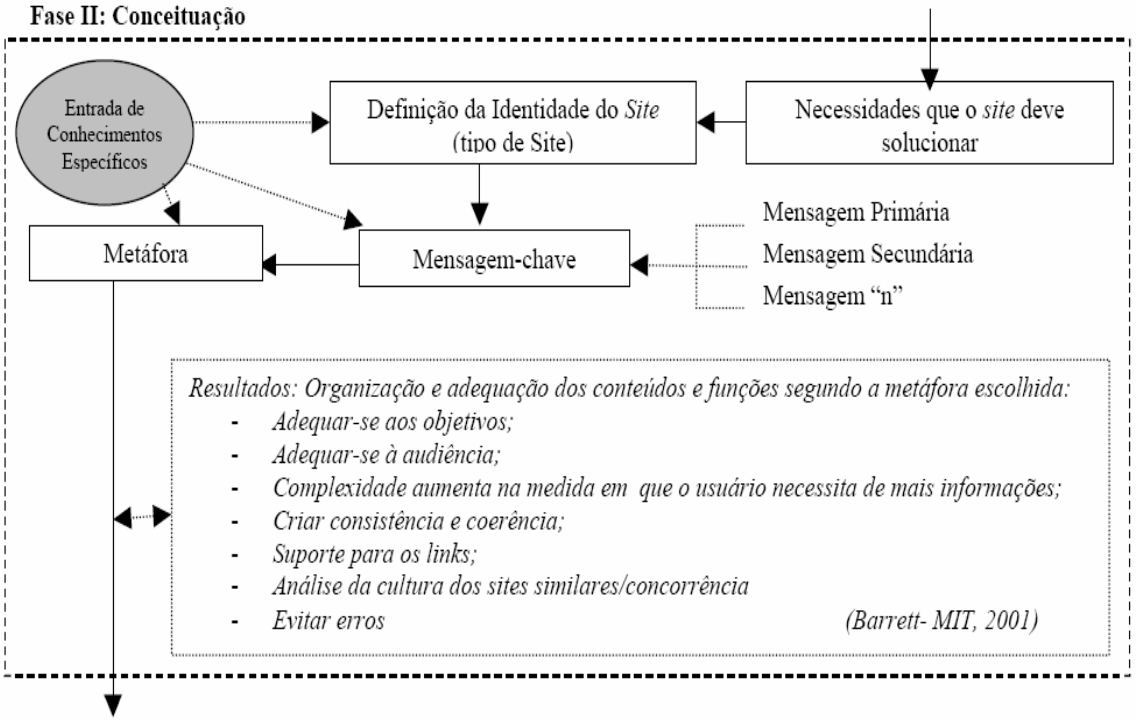


Figura 3.13. Metodologia de Conceituação (Barret, 2001).

3.4.2.1. Mensagem Chave da Interface

Dando continuidade ao processo metodológico, é necessário que o *site* seja rapidamente identificado pelo usuário assim que o abrir em seu computador. Logo, há necessidade de determinar uma mensagem-chave que permita ao usuário perceber que está acessando o *site* correto para atender suas necessidades.

Seguindo a metodologia de Barret (2001), para a seguinte interface, de acordo com a sua complexidade e dependendo também dos usuários, muitas vezes a mensagem-chave precisa se apoiar em um conjunto de mensagens secundárias. No caso da interface para o código do LTCM, precisa-se especificar que, estando na página referente ao solver – figura 3.14, há necessidade de uma mensagem-chave indicadora do local para acessar as informações do solver – figura 3.15.

Figura 3.14 – Mensagem-chave indicando o *site*.

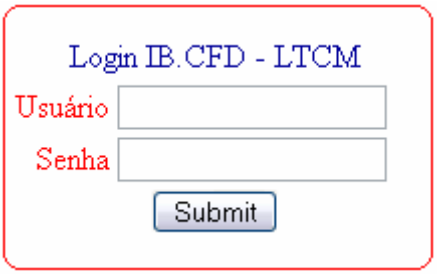


Figura 3.15 – Mensagem-chave indicando acesso às informações do solver.

Com estas informações bem delimitadas, de forma que a mensagem-chave seja clara e objetiva, dá-se inicio a conceituação do *site* partindo pela identidade do mesmo.

3.4.2.2. Metáfora da Interface.

Segundo Deborah Levinson, designer do Art Technology Group, Inc/USA,

(profissional convidada para palestra no *MIT* durante as aulas de Comunicação no Ciberespaço): "*Em um projeto de comunicação, a metáfora deve ser pensada como uma associação, reação emocional, experiência, relacionamento com a audiência".*

De acordo com a mensagem-chave, escolheu-se a metáfora do mundo da Dinâmica dos Fluidos Computacional. Foi utilizado, como referência, o "suporte" que leva à mecânica dos fluidos até as mãos do público alvo, através da conceituação de simulação computacional. É a imagem que possibilita reconhecimento imediato de CFD, sendo o elemento da metáfora escolhido. Figura 3.16.

Figura 3.16. Mensagem-chave indicando a metáfora da interface.

3.4.3. Fase III: Organização da Interface

Nesta fase os conteúdos do *site*, conforme as definições anteriores, são organizados e alocados em um planejamento visual coerente, através de textos, imagens, grafismos, entre outros. O *site* começa a tomar forma visual no momento em que os requisitos técnicos, específicos de mecânica dos fluidos, contribuam acentuadamente como orientadores do seu processo de criação e desenvolvimento. Para orientar a escolha dos elementos presentes na interface e a organização dos conteúdos de acordo com o conceito escolhido, público alvo e cliente, permitindo um planejamento de qualidade estético-formal coerente à proposta do *site*, deve-se abordar alguns critérios importantes na implementação da relação homem-software, como, informação e comunicação, ergonomia em conjunto com usabilidade e considerações técnicas de layout.

Para “fidelizar os usuários”, a página inicial deve permitir, ao mesmo tempo, que seja conhecida qual informação pode-se obter com a utilização da interface para o solver do LTCM, bem como do conhecimento sobre o uso da metodologia de fronteira imersa para solução de escoamento sobre geometrias complexas, além de citar a importância da geração da malha computacional. Deve ser uma premissa básica que as informações contidas na página, mostrem que o solver trabalha com técnicas avançadas para solucionar problemas para o qual foi proposto, com ilustrações de simulações para caracterizá-lo.

Como o *site* tem objetivo acadêmico, o único tipo de marketing utilizado para divulgá-lo, é um cabeçalho com o logotipo do LTCM e as mensagens-chave identificando que o *site* acolhe conteúdo de dinâmica dos fluidos, juntamente com o solver vinculado a interface.

No que se refere ao layout, não serão utilizadas cores que identifiquem um público específico, uma vez que este público é muito variado e eclético. Será proposto utilizar as cores do logotipo e variações das mesmas. O mesmo cabeçalho permanecerá em todas as páginas, com intuito de tornar-se uma imagem pregnante, originando um padrão estético para o *site* como um todo.

Referindo-se a ergonomia e usabilidade, serão destacados os *links* para configuração de casos a serem simulados. A organização dos *links* foi elaborada de forma a manter uma ordem funcional na configuração da simulação dos casos. Além disso, os *links* são dispostos em forma de ficheiro, de forma que quando o link ativo (destacado em verde – figura 3.17) estiver selecionado, as demais abas do ficheiro não mostrem as informações contidas nelas.

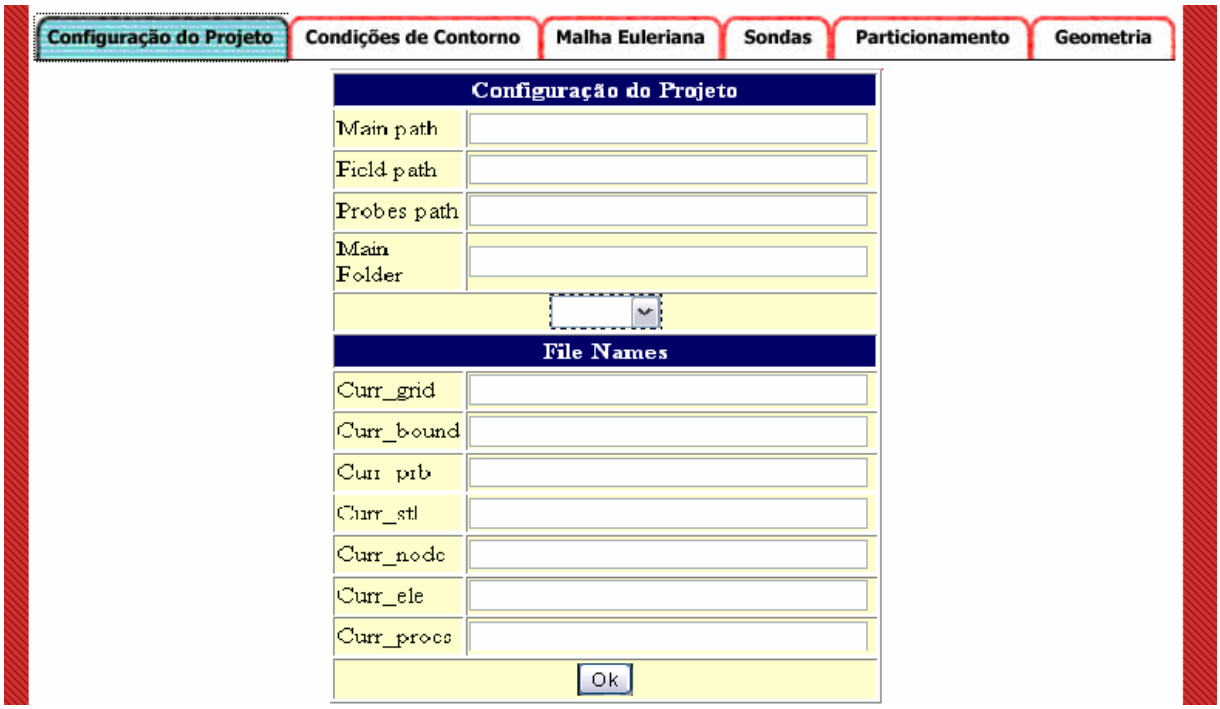


Figura 3.17. Destaque dos *links* em relação à ergonomia e usabilidade.

Com a interface disposta dessa forma, propõe-se que o usuário inicie sua simulação, preenchendo informações em cinco formulários seqüenciais, sendo cada um destes relacionado a um arquivo que será vinculado ao solver do laboratório.

Com a organização do *site* definida segundo os conteúdos específicos necessários, pode-se elaborar o mapa do mesmo, Figura 3.18.

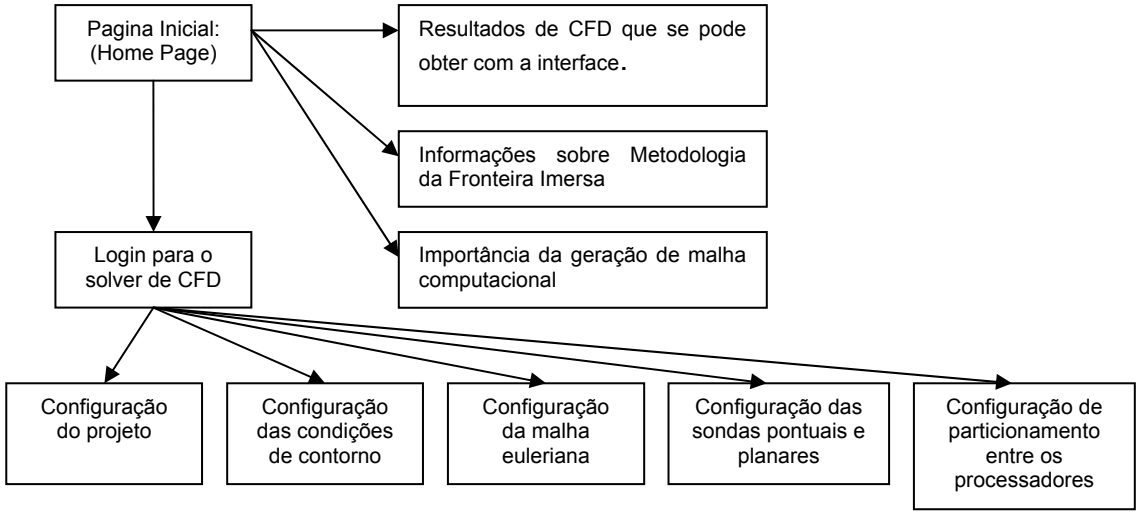


Figura 3.18. Mapa do *site* IB-CFD.

Definida a organização do *site* e elaborado o seu mapa, pode-se roteirizar cada página com seu conteúdo, apresentado na tabela 3.1 abaixo. Nesta fase de roteirização, são abordados os conhecimentos de mecânica dos fluidos que estão vinculados ao solver do LTCM.

Página	Roteiro
Página Inicial (Home Page)	- Conteúdo explicativo sobre: • Resultados de CFD que se pode obter com a interface vinculada ao solver do LTCM; • Breve introdução à Metodologia de Fronteira Imersa; • Importância da fase de geração de malha computacional utilizada nos casos a serem simulados usando-se o solver. Área de acesso ao solver, com restrição de acesso por usuário e senha.

Página de Configuração do Projeto	Configuram-se as características da simulação a ser executada pelo solver, dentre elas estão: - Configuração do projeto: • Main Path: indica o caminho onde o programa será rodado; • Field Path: indica o caminho onde o programa gravará os campos; • Probes Path: indica o caminho onde o programa gravará as sondas; - Nome dos arquivos: • Curr_grid: nome usado para o arquivo malha euleriana • Curr_bound: nome usado para o arquivo de condições de contorno e condições iniciais; • Curr_prb: nome usado para o arquivo de posicionamento das sondas; • Curr_stl: nome usado para o arquivo STL com características da geometria; • Curr_node: nome usado para o arquivo de nós com características da geometria; • Curr_ele: nome usado para o arquivo de conectividade com características da geometria; • Curr_procs: nome usado para o arquivo de particionamento do domínio de cálculo. Botão de validação das informações preenchidas no formulário.
Página de Condições de contorno	Configuram-se as condições de contorno e condições iniciais, editando os seguintes tópicos do formulário: • u_in: velocidade inicial em x; • v_in: velocidade inicial em y; • w_in: velocidade inicial e z; • t_in: temperatura inicial; • dte: passo de tempo; • final: período total a ser simulado;

	• div_max: máximo divergente; • steplot: intervalo de tempo para salvar os campos; • rho_cte: massa específica; • mi_cte: viscosidade dinâmica; • k_cte: condutividade; • cp: calor específico; • beta: coeficiente de expansão; • grav: aceleração da gravidade; • upwall: caixa de seleção caso haja ou não parede em Ymax (último volume na direção y); • dowall: caixa de seleção caso haja ou não parede em Ymín (primeiro volume na direção y); • topwall: caixa de seleção caso haja ou não parede em Zmax (último volume na direção z); • bowall: caixa de seleção caso haja ou não parede em Zmín (primeiro volume na direção z); • upisol: caixa de seleção caso a parede seja adiabática ou não; • t_up: temperatura em y=ytotal; • doisol: caixa de seleção caso a parede seja adiabática ou não • t_do: temperatura em y=1 • topisol: caixa de seleção caso a parede seja adiabática ou não; • t_top: temperatura em z=ztotal • boisol: caixa de seleção caso a parede seja adiabática ou não; • t_bo: temperatura em z=1; • les: caixa para seleção da metodologia de modelagem de turbulência LES;

	• lesvd: caixa para seleção da função amortização para LES; • as: caixa para seleção do modelo de turbulência SA; • Des: caixa para seleção do modelo de turbulência DES; • Cs: constante de smagorinsky; • start_up: caso haja interrupção na simulação, selecioná-lo e indicar o valor de iterações para que seja reinicializada a simulação após aquele valor; • termic: caixa para seleção caso use a equação de energia ou não; • period : caixa para seleção caso seja periódico em y ou não; • lag: caixa para seleção caso use fronteira imersa ou não; • arquivo stl: caixa de seleção caso use Usa arquivos stl para fornteira imersa; • arquivo nos: caixa de seleção caso use arquivos provenientes do Ansys para fornteira imersa; • alpha do Sip: informar o fator de relaxamanto do SIP; - Botão de validação das informações preenchidas no formulário.
Página da malha euleriana	Para o arquivo de Malha euleriana deverá se especificar as seguintes características: - Tamanho da Malha euleriana nas posições x, y e z; Numero de Zonas em x, y e z, sendo que para cada zona, dever-se-á especificar as seguintes características: • Fim da zona; • Número de malhas; • Taxa de expansão ou relaxamento;

	• Sinal para indicar contração “+”, ou expansão “-”. - Botão de validação das informações preenchidas no formulário.
Página de Sondas	Indica a configuração das sondas pontuais e planares, caso existam seguirá as seguintes características: - Número de sondas pontuais, sendo que cada sonda tem as seguintes especificações: • Inicio da sonda em i, j, k; • Posição da sonda em x, y, z; • Caixa de seleção da variável, podendo ser: u, v, w, p, t; • Freqüência. - Número de sondas planares sendo que cada sonda contém as seguintes especificações: • plano onde está a sonda: sendo x = 1, y = 2 e z = 3; • posição na direção escolhida; • passo de tempo para salvar as informações deste plano. - Botão de validação das informações preenchidas no formulário.
Página de particionamento	Configura o arquivo de particionamento dos domínios, escolhendo a quantidade de processadores usados no cluster para simular o caso. O arquivo conterá: - Número de processadores usados, sendo que para cada processador deverá ser especificada a seguinte informação: • Inicio do domínio em x, y e z; • Fim do domínio em x, y e z; • Número de volumes em x, y e z; - Valores para transladar a fronteira imersa para a

	posição nos domínios x, y e z; - Botão de validação das informações preenchidas no formulário.

Tabela 3.1. Roteirização do *site* para interface.

Referente ao roteiro com as informações das páginas do site, acima citado, cabe observar que as Condições de Contorno que definem a interface fluido-sólido nos métodos convencionais são tratadas virtualmente pela metodologia de fornteira imersa. Em outras palavras, os campos de força acrescentados às Equações de Navier Stokes modelam a interface fluido-sólido. As condições de contorno na geometria (malha lagrangiana) inserida neste domínio podem ser impostas através do modelo físico virtual (Lima e Silva, 2002), tal caractetrística é uma opção na forma de implementação do solver em questão, devido a metodologia adotada para o mesmo.

Depois de finalizado o roteiro, segue-se aliando as considerações provenientes dos estudos até agora efetivados, e assim parte-se para a idealização do layout da interface. Sendo a página inicial composta pela figura 3.19. e as demais páginas semelhantes a figura 3.20, alterando somente as informações referente a cada aba, citada na tabela 3.1 acima.

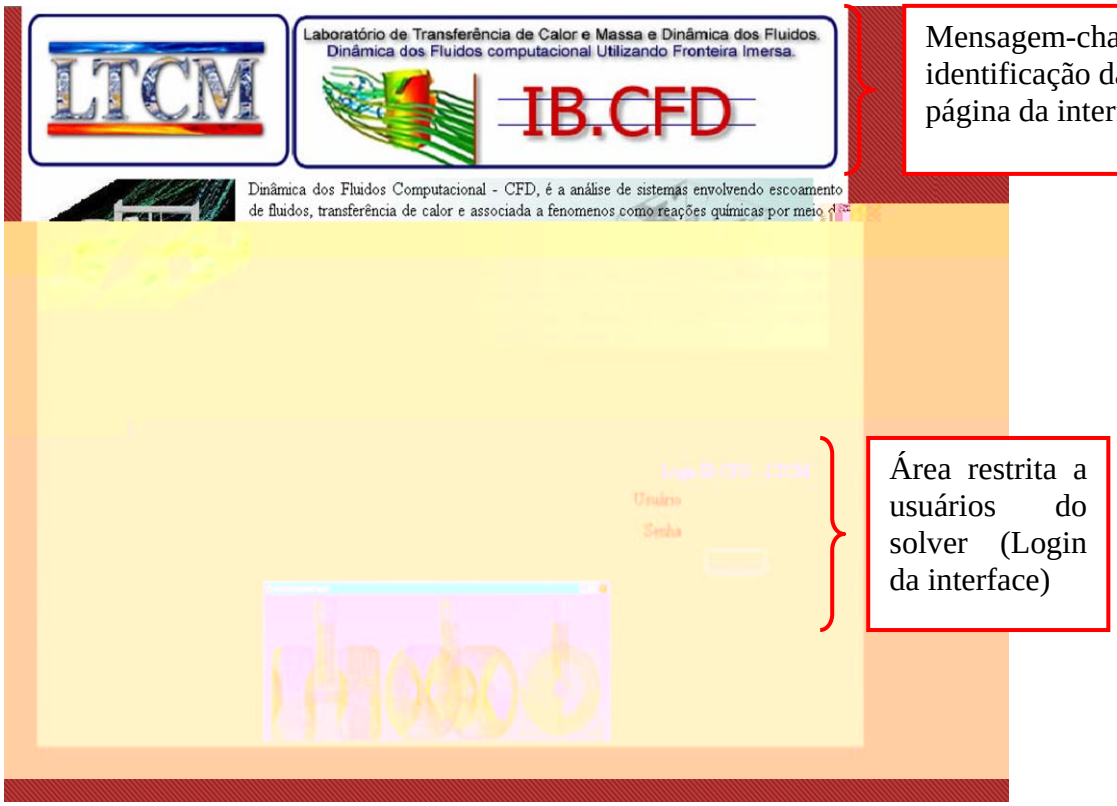


Fig 3.19. Página Inicial.

Fig 3.20. Página da interface de configuração do solver.

3.4.4.Fase IV: Desenvolvimento e Implementação

Nesta fase da metodologia, será abordada a programação a ser vinculada à interface, utilizando-se de tecnologias disponíveis, de acordo com a qualificação das mesmas, feitas no capítulo anterior. Apesar de alterar os códigos computacionais da interface, acrescendo algumas páginas de programas em PHP, a interface não sofre alterações em função da programação.

Uma das páginas de código acrescentada a interface é a validação do *login* de usuários mostrada na figura 3.21 abaixo. Nesta página, as funções mais importantes usadas são: método POST e consulta SQL. No método POST, as informações digitadas na área de *login* (usuário e senha) são passadas ao código para posterior comparação. A consulta em SQL é utilizada para buscar no banco de dados se existem as informações passadas pelo método POST, com esta informação é feita uma comparação para validar a existência do usuário, mediante tal validação, o usuário tem acesso à interface do solver.

```
<?php
        $user = $_POST["usr"];
        $passwd = $_POST["passwd"];
        $conexao = mysql_connect ("localhost", "root", "") or    // conecta ao B.D
                                    die('Could not connect: ' . mysql_error());
        mysql_select_db("login");                                  // seleciona a tabela
        $resultado = mysql_query("select * from usuario where usr='$usr'
                                    and passwd='$passwd'");       //consulta existência
                                                                    // de usuário
        $linhas = mysql_num_rows($resultado);    //  verifica número de resultados
                if($linhas != 0)                  // se resultado diferente de zero
                { header("location: menu.htm");}  // chama página do solver
                        else{include "user_incorreto.html";} // chama página de erro
                        mysql_close($conexao);                // fecha conexão c/ BD.

?>
```

Figura 3.21. Página de Código para validação de usuário.

Além das funções acima citadas, o PHP nos oferece diversas funções relacionadas com o *filesystem* do sistema operacional. Pode-se então utilizá-las para realizar operações de gravação e leitura de dados em arquivos-texto (txt). As mais utilizadas são *fopen* (abertura), *fclose* (fechamento), *fwrite* (escrita), *fread* e *fgets* (leitura). Utilizando estas funções, as informações digitadas nos formulários da interface do solver serão arquivadas para posterior uso pelo código. Abaixo (figura 3.22) segue uma ilustração de como são usadas as funções acima citadas, uma vez que o código da interface possui cinco páginas de formulários e cada uma com várias informações a serem escritas em arquivos, um exemplo simplificado será mostrado.

```
<?php
        $arquivo = “cond_contorno.txt”;
        $write = “Condições de contorno”;
        $fd = fopen ($arquivo, “w”);  //abre o arquivo
                fwrite ($fd,$write);       //escreve no arquivo
                fclose ($fd);                //fecha o arquivo
?>
```

Figura 3.22. Uso simplificado das funções de PHP.

3.4.5 – Fase V: Caixa Preta

Como citado anteriormente, a caixa preta denomina uma atividade transformativa entre o que entra de um lado da caixa, o processo obscuro que ocorre dentro desta, e o

resultado final que sai pelo outro lado. Os resultados são apenas respostas do processo de simulação computacional, no qual sempre que se haja necessidade de efetuar uma nova simulação, novos dados poderão entrar e ser considerados, alterando o resultado final.

Como o objetivo dessa dissertação não é validar o código computacional existente, esta etapa serve somente para citar que existe um código que será vinculado a interface, de forma a facilitar o uso do solver.

3.4.6 – Fase VI: Manutenção e Observação

Como a interface foi desenvolvida no LTCM, as devidas sugestões e modificações a serem executadas foram realizadas no decorrer da implementação do *site*, uma vez que houve participação conjunta da equipe do laboratório para compor o design da interface. No entanto, alterações posteriores são sempre possíveis para atualização e melhorias da interface.

3.5. Metodologia de Geração de Geometrias

A metodologia adotada para geração de geometrias e por conseqüência das malhas, segue conceitos adotados pelo GMSH, por ser esta a ferramenta adotada para gerar as formas a serem virtualizadas computacionalmente. Vide Anexo nº I - Tutorial do GMSH.

No GMSH, geometrias são criadas por um fluxo sucessível de clique para definir os pontos das mesmas, definir suas linhas orientadas (segmentos de retas, círculos, elipses, splines, etc.), superfícies orientadas (superfícies planas, superfícies curvas, etc.) e volumes. Grupos compostos por entidades geométricas podem ser definidos, baseados nestas entidades geométricas elementares.

A malha de elementos finitos gerada com base na geometria criada a priore, é a tecelagem de um dado subconjunto de espaços tridimensionais de elementos geométricos elementares de varias formas (Ex: linhas, triângulos, quadriláteros, tetraedros, prismas, hexaedros, e pirâmides), arranjados de forma que se dois deles se interceptam, eles formam uma face, uma borda ou um nó. Portanto a geração da malha é executada na mesma seqüência de criação da geometria, partindo das estruturas mais simples até chegar às mais complexas.

As geometrias simples são desenhadas com conceitos simples. Para criar, por exemplo, uma esfera, basta definir as dimensões da mesma, especificar os pontos extremos

da mesma, ligar estes pontos com arcos de circunferência e definir que cada face da esfera será uma superfície curva, Figura 3.23.

No caso de geometrias complexas, a idéia principal é partir da composição de geometrias simples e conseqüentemente gerar formas complexas compostas. Porém, para as geometrias complexas desejadas para esta dissertação, sendo algumas delas automóveis e aviões, a simples composição através de união de formas simples torna-se inviável. Sendo assim, existe a necessidade de certa destreza para desenho, o que dificulta mais ainda a geração dessas geometrias, uma vez que no GMSH os pontos devem ser editados um a um.

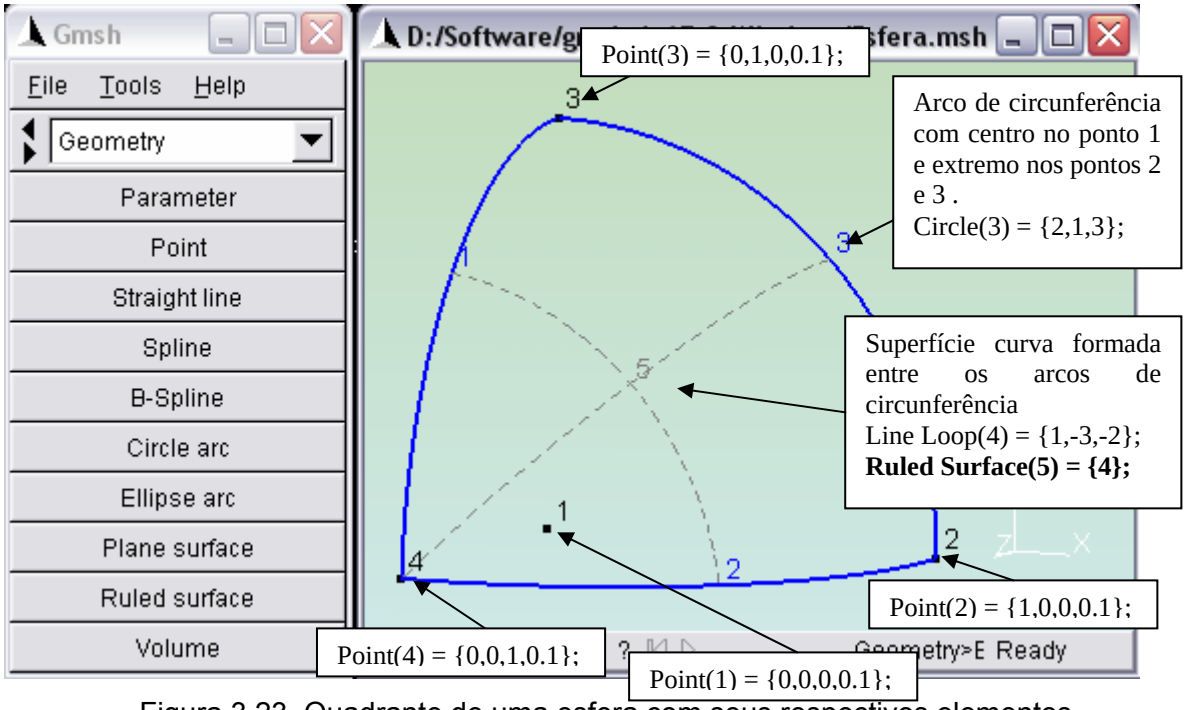


Figura 3.23. Quadrante de uma esfera com seus respectivos elementos.

Para os exemplos de geometria complexas acima citados, informações sobre os modelos originais em tamanho real foram pesquisadas no *site* dos fabricantes dos mesmos, sendo eles:

Modelo:	Lamborghini Gallardo	Legacy 600
Comprimento:	4,3m	26,33m
Largura:	1,9m	21,17m
Altura:	1,165m	6,76 m

Tabela 3.2. Dimensões dos modelos originais.

Após verificar as dimensões dos modelos a serem virtualizados

computacionalmente, há uma necessidade de adquirir fotos dos mesmos, com a finalidade de visualizar detalhes que não podem ser vistos com imagens 2D. No caso do Legacy 600, algumas fotos adicionais foram consultadas, pois o comprimento acima citado faz referência à envergadura das asas, necessitando ainda de informações sobre altura e comprimento do corpo do avião, Figura 3.24. Lamborghini Gallardo e Figura 3.25. Legacy 600.

Verificado as dimensões, a única necessidade antes de virtualizar o modelo é fazer um desenho em unidades de medidas menores, para facilitar o desempenho computacional. Na figura 3.26 mostra-se uma Lamborghini Gallardo desenhada em escala. Na figura 3.27, um legacy 600 visualizado em escala.

Figura 3.24. Lamborghini Gallardo.

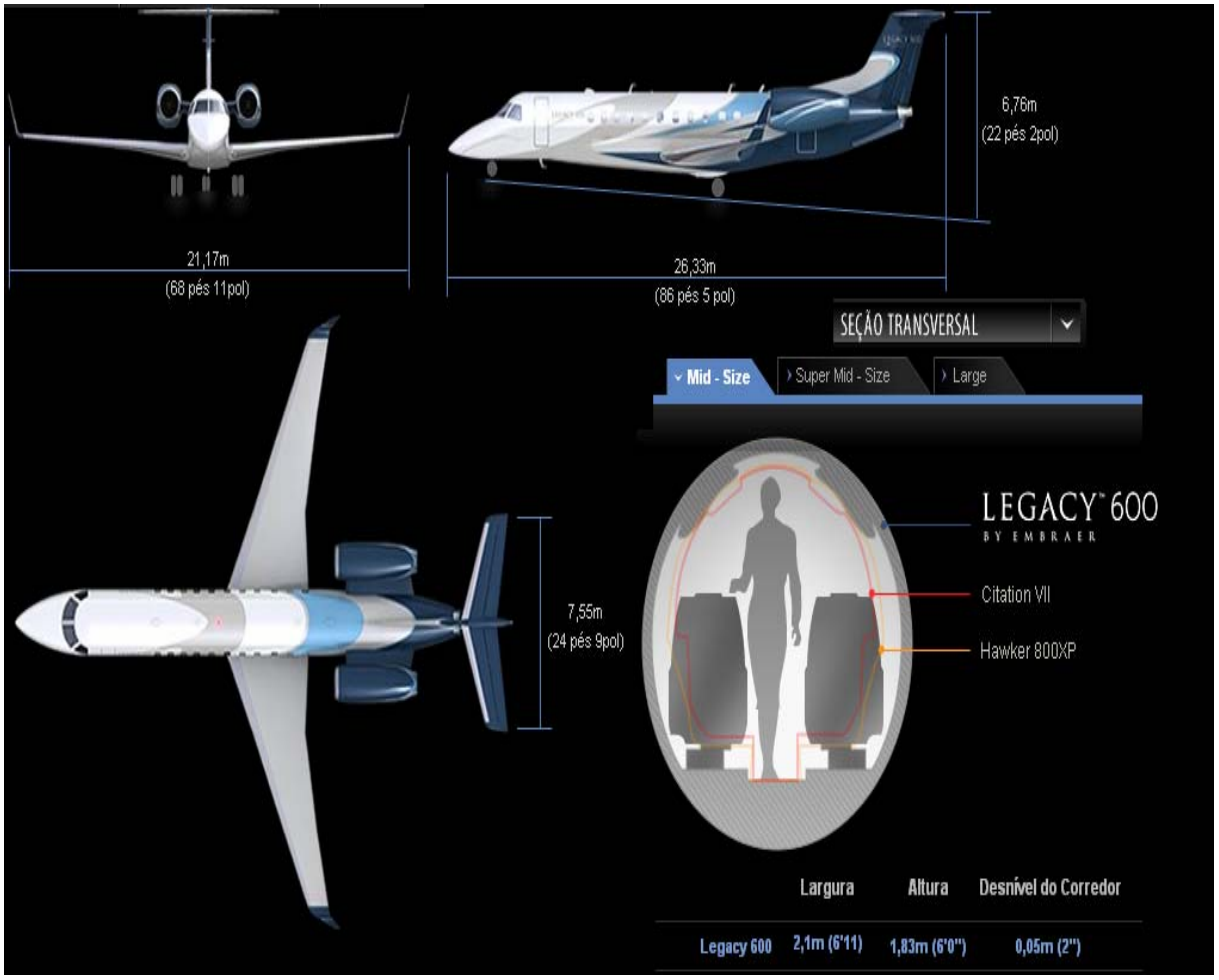


Figura 3.25. Legacy 600 com especificação das dimensões.

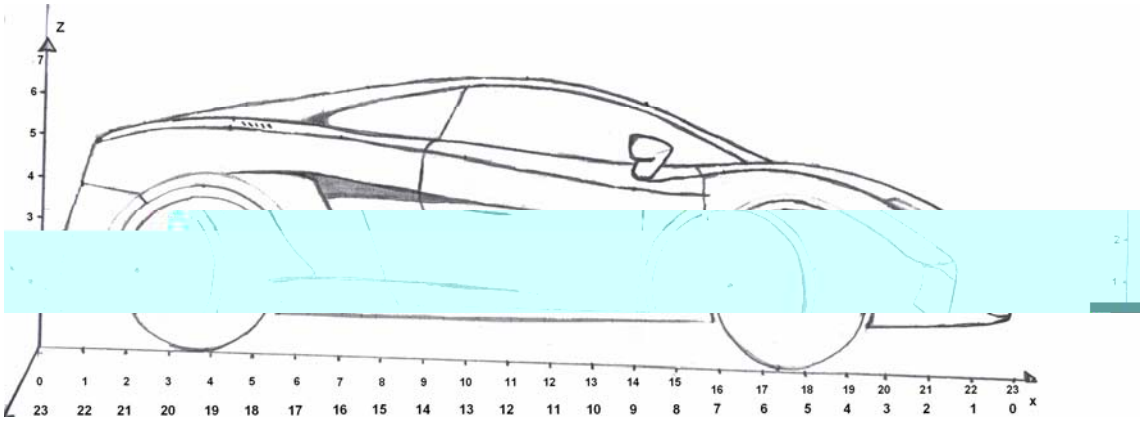


Figura 3.26. Croqui da Lamborghini com escala em centímetros.

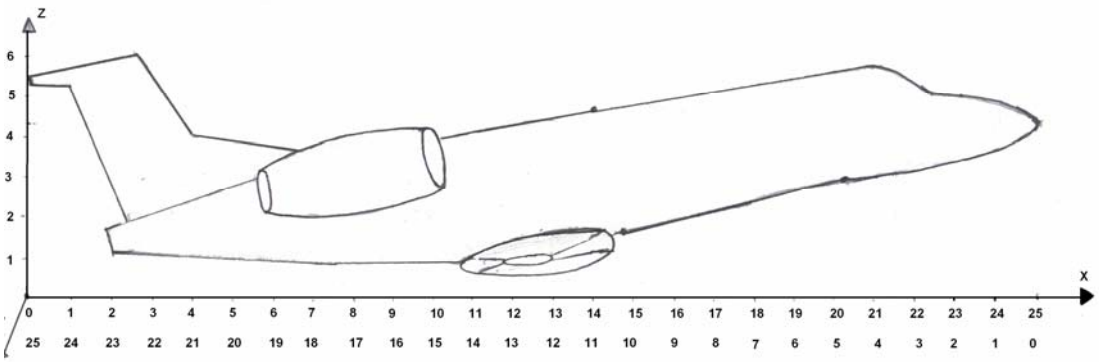


Figura 3.27. Croqui do Legacy 600 com escala em centímetros.

Como se pode notar, os esboços das geometrias a serem geradas no GMSH são bem simplificados, de forma a orientar na criação do perfil dos mesmos. Para se desenhar o modelo antes de virtualizá-lo, algumas definições são necessárias. A geometria a ser desenhada deve estar apontando para o início do eixo “x”. Por isso existem duas escalas neste eixo, uma com a orientação crescente e outra invertida, para que na hora da virtualização se utilize a escala invertida. Essa preocupação com a orientação da geometria parte da predefinição que o domínio onde a geometria será inserida no eixo x, portanto para simular o escoamento contra a geometria ela deve estar apontando para o início deste eixo. Além desta especificação, a altura da geometria deve ser definida no eixo z, ficando o comprimento em y.

Mesmo sendo um processo que exige destreza e paciência, os resultados obtidos na geração de geometrias complexas são satisfatórios. Os resultados serão mostrados no capítulo seguinte, juntamente com outras geometrias simples.

CAPITULO 4

Resultados

Neste capitulo, serão apresentados os resultados referentes à interface gráfica desenvolvida para o código CFD Fluids-3D, mostrando suas principais características.

De acordo com Vedovoto (2007), as mudanças realizadas no pré-processamento do solver disponível no LTCM viabilizaram uma maior facilidade de *set-up* físico das simulações e facilitou também a importação de geometrias, possibilitando assim, o estudo de escoamentos sobre diversas geometrias.

Portanto, serão apresentados resultados obtidos com a utilização do solver do LTCM, através dos quais torna-se possível verificar a validade das malhas geradas utilizando o GMSH, bem como resultados de aplicações de CFD a escoamentos sobre geometrias complexas. Vale ressaltar que como o objetivo desse projeto era elaborar uma interface para pré-processamento, as imagens de pós-processamento foram elaboradas com pacotes comerciais e geradas por Vedovoto (2007).

Assim, serão ilustradas as duas contribuições do presente trabalho: o desenvolvimento da interface gráfica que possibilita o acesso remoto ao solver, facilitando no seu manuseio, e também no estabelecimento do procedimento de geração de malha utilizando-se o GMSH.

4.1. Idealização do Layout da Interface

Após a idealização da interface, utilizando as definições metodológicas citadas no capítulo anterior, parte-se para idealização da interface a ser relacionada com o solver do

LTCM. Nesta idealização, serão apresentadas as páginas da interface, juntamente com seus respectivos comentários técnicos, conforme mostrado na figura 4.1.

4.1.1 Fatores Humanos no projeto da interface.

Por ser uma ferramenta de diálogo entre o computador e o ser humano e para que isto ocorra de forma harmoniosa, deve-se levar em conta os fatores humanos, ou seja, deve-se entender o usuário e seu comportamento. As tarefas que o sistema executa para o usuário e principalmente as tarefas que são exigidas do usuário nesta interação devem ser levadas em consideração.

Para produzir uma interface, deve-se levar em conta o elemento primordial na interação com o software: o usuário. Respondendo a perguntas do tipo: quem são eles; como eles interpretam as informações produzidas pelo sistema; como ele aprende a interagir com o sistema; podendo-se assim, projetar a interface da melhor forma possível.

Quando é necessária uma interface ser humano-computador (HCI - *Human-Computer Interface*), deve-se lembrar que para o ser humano é predominante o uso dos sentidos visual, tátil e auditivo, através da qual o usuário de um sistema possa abstrair melhor da informação que a ferramenta possibilita fornecer, armazená-las na memória e processá-las usando raciocínio indutivo.

Para um raciocínio indutivo no desenvolvimento da interface, utilizou-se de meios visuais como, por exemplo: relatórios, formulários, gráficos ou qualquer atrativo visual. Os estilos de integração ser humano-computador percorrem uma grande quantidade de opções que estão relacionadas às tendências HCI, porém como foi visto na introdução do presente trabalho, a ferramenta disponível no LTCM possui um modelo de HCI denominado de interface de comando e consulta, justificando assim, a utilização das tendências supra citadas. A interface que era utilizada no solver do LTCM funcionava através de comunicação puramente textual e impulsionada mediante os comandos e respostas a consultas geradas pelo sistema, na qual o usuário especifica um comando no modo texto de interação com o software.

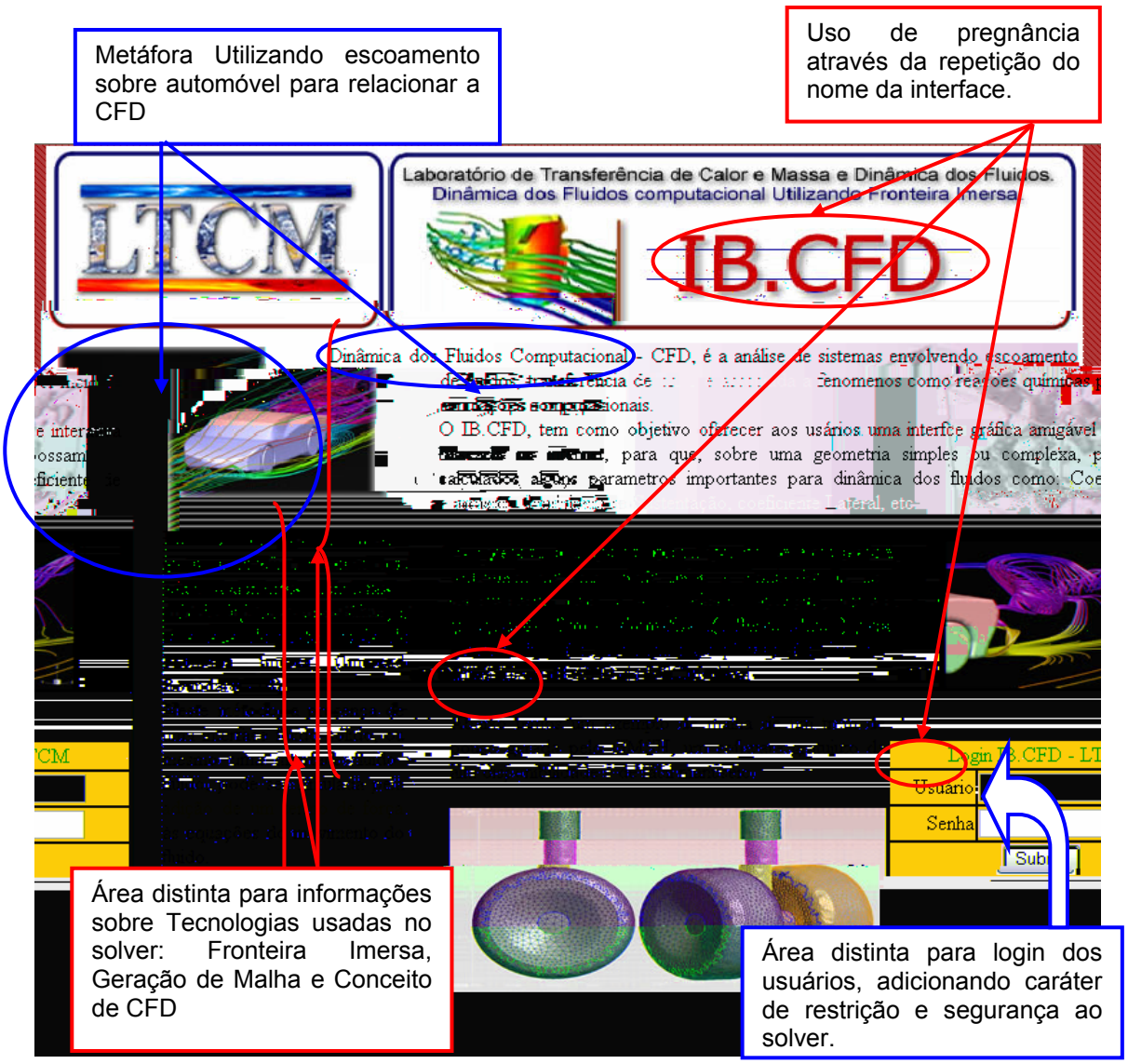


Figura 4.1. Página inicial do solver do LTCM com comentários técnicos.

Como mostrado na figura 4.1 acima, a página inicial segue a organização do roteiro, abordando o conteúdo específico de mecânica dos Fluidos. Por ser uma página informativa, a página inicial não possui menus aparentes, evitando a iteração de usuários com a interface do solver, sem autorização previa. Para a autorização do usuário, existe uma área distinta para *login*, atribuindo caráter de restrição e segurança para o solver do LTCM.

Como foi citado na metodologia da atual dissertação, foram utilizadas técnicas de desenvolvimento de interface gráfica. Portanto, desenvolver a interface simplificada, tem uma justificativa simples, pois ao facilitar o manuseio da interface, o usuário obterá um ritmo natural de interação, podendo até se esquecer de que está se comunicando com um software.

A característica de facilidade na iteração usuário-interface é visível na página inicial,

caracterizada por ter pouco conteúdo, valendo-se de mensagens chaves explicitas, de forma a identificar facilmente a funcionalidade da interface.

Com o aprofundamento no aprendizado sobre fatores humanos e suas influências sobre o projeto de interfaces, concluiu-se que a melhor forma de oferecer benefícios ao usuário é por meio de uma interface moderna baseada em janelas[1] como mostrado abaixo, figura 4.2.

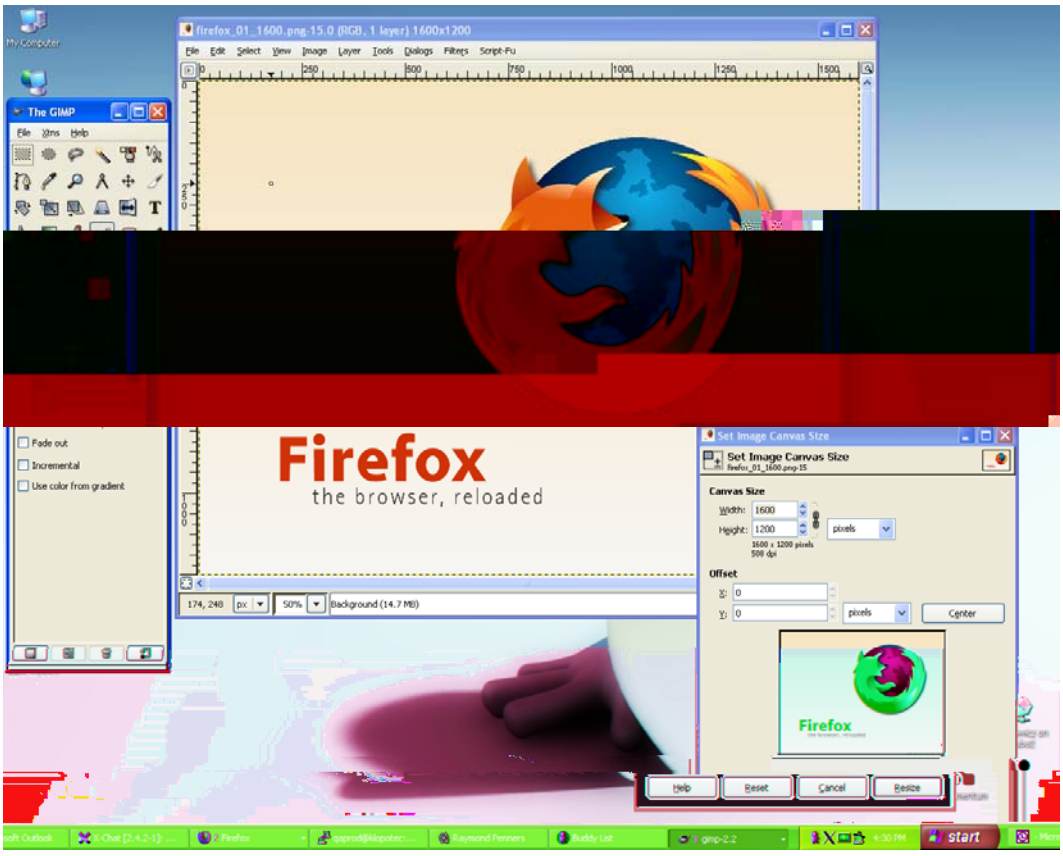


Figura 4.2 – Janela, ícones, menus e dispositivos de indicação.

Como objetivo do presente trabalho é oferecer ao usuário do código do LTCM uma interface que lhe possibilite exibir diferentes tipos de informações, de modo que o usuário possa interagir entre formulários no solver do LTCM, tendo disponível em uma tela a visualização de uma página com todas as abas (em formato de identificador de ficheiro) necessárias para configurar uma simulação computacional, Figura 4.3.

[1] Estas interfaces são citadas como interfaces WIMP (windows, icons, menus and point device) ou seja, janelas, ícones, menus e dispositivos de indicação.

Figura 4.3. Área do solver restrita ao usuário, com abas dos respectivos itens a serem configurados.

Como visto na figura acima, a aba para configuração do caso a ser simulado tem destaque em cor diferente das demais, e estando ela selecionada destacada na cor verde, os conteúdos dispostos nas abas seguintes não podem ser visualizados. Além disto, a organização dos *links* está elaborada de forma a manter uma ordem funcional na configuração da simulação dos casos.

Com a finalidade de firmar a forma de iteração do usuário com o solver, elementos característicos são utilizados em ambas as telas, para tornar sistêmico o uso da interface. Dentre estes elementos temos o uso de um botão de validação das informações preenchidas nos formulários de cada página, e também, um botão de interrogação, simbolizando a metáfora conhecida nas interfaces como ajuda ou *help*.

4.1.2 Modelos de Projeto de Interfaces

O modelo de usuário descreve o perfil dos usuários finais do sistema. Para construir uma interface efetiva com o usuário, o projeto deve-se iniciar com o entendimento do usuário a que se destina, podendo este ser qualificado como:

- Principiante: sem conhecimento sintático do sistema ou pouco conhecimento semântico da aplicação ou uso do computador em geral. Neste perfil podemos encaixar os alunos de graduação que estão adquirindo conhecimentos sobre Dinâmica dos Fluidos Computacional. Portanto os mesmos terão dificuldade em interagir com a ferramenta, dado o fato que muitas das nomenclaturas utilizadas pelo software depender de conhecimento prévio em tal área.
- Usuários instruídos e intermitentes: possuem razoável conhecimento semântico da aplicação, mas relativamente pouca lembrança de informações sintáticas para usar a interface. A este perfil podemos classificar os usuários com conhecimento em dinâmica dos fluidos computacional, porém, que não sabem como aplicá-la usando a ferramenta em questão.
- Usuário instruído e freqüente: possuem bom conhecimento semântico e sintático, ou seja, o usuário que tem o conhecimento sobre dinâmica dos fluidos computacional e sabe como aplicá-la.

Esses modelos descritos acima são abstrações de como o sistema pode ser visualizado pelo usuário, e a partir destes modelos produziu-se uma interface que satisfaça o elemento principal em um projeto de interfaces: conhecer o usuário e suas tarefas.

4.1.3. Análise e Modelagem da Tarefa

A análise de tarefas pode ser aplicada de duas maneiras. Para entender as tarefas que devem ser executadas para que possa atingir a meta da atividade, um engenheiro de software deve entender as tarefas que os usuários realizam correntemente quando usam uma abordagem manual e depois elaborar um conjunto semelhante de tarefas implementadas no contexto da HCI.

Outro modo é estudar uma especificação existente de uma solução baseada em computador e derivar um conjunto de tarefas do usuário que acomodem o modelo usuário, o modelo do projeto e a percepção do sistema.

Levando em conta estas duas abordagens para análise e modelagem de tarefa da

interface para o solver do LTCM, a fim de facilitar a modelagem, segue-se a segunda abordagem, visto que o laboratório já possui uma ferramenta, que não atende aos moldes de HCI.

Partindo deste princípio, foi classificada a principal tarefa envolvida durante a elaboração de um caso com a ferramenta do LTCM, para simulação.

4.1.4. Configuração do Projeto

Nesta seção, o usuário deverá ter informações sobre as configurações do cluster para que se possa indicar o local no qual o programa será executado (caminho principal), tanto esse "*path*" como os demais caminhos citados deverão ser exatos, tais como existentes na máquina de destino no cluster, figura 4.4. Especificar como o caminho de execução do programa, devendo indicar também o caminho onde o programa gravará os campos e as sondas. Para a parte inicial da configuração de uma simulação, ainda deve-se informar qual o nome do arquivo que indica a malha euleriana, o nome do arquivo com condições iniciais e de contorno, o nome do arquivo com o posicionamento das sondas

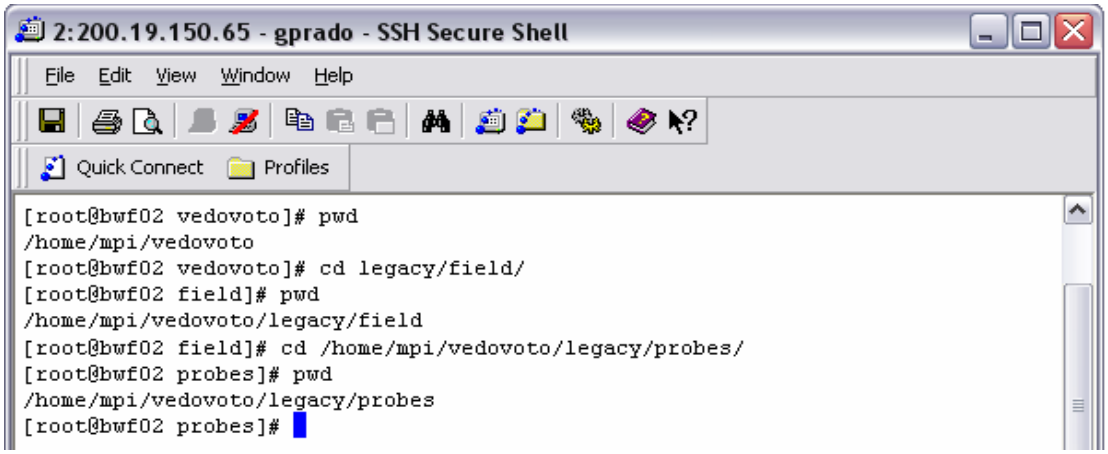


Figura 4.4. Caminho específico dos diretórios com os locais para gravar informações no cluster.

Neste momento, o solver ainda se encontrará em estado de espera, pois nenhuma informação ainda foi inserida no código. A meta é que o caso a ser executado altere o estado do sistema assim que todas as etapas estejam configuradas, podendo assim ser executado sem nenhuma interrupção.

Continuando a configuração da simulação, a próxima etapa é tão importante quanto as demais, lembrando que nenhuma delas pode ser pulada, uma vez que a interface foi elaborada da forma mais simplificada possível.

Na configuração do arquivo de condições de contorno e condições iniciais são setadas as seguintes informações:

- Variáveis com as velocidades iniciais em u, v e w;
- Temperatura inical é especificada em “t_in”;
- Passo de tempo é especificado em “dte”, sendo que nas primeiras iterações, usa-se um valor menor para intensificar a força lagrangiana quando fizer uso de Fronteira Imersa (Campregher, 2005);
- A variável “final” armazena o período de tempo a ser simulado;
- Na variável “div_max”, será armazenado o máximo divergente, utilizado para manter a conservação da massa;
- “Steplot” deve conter o valor do intervalo de tempo entre o qual serão salvos os campos;
- rho_cte indica a densidade;
- mi_cte indica a viscosidade dinâmica;
- k_cte indica a condutividade;
- cp indica o calor específico;

- beta indica o coeficiente de expansão e grav indica aceleração da gravidade, sendo que estes valores serão especificados somente se a variável “termic”, que será citada posteriormente, estiver setada para a simulação;
- upwall indica caixa de seleção caso haja ou não parede em Ymax (último volume na direção Y);
- dowall indica caixa de seleção caso haja ou não parede em Ymín (primeiro volume na direção Y);
- topwall indica caixa de seleção caso haja ou não parede em Zmax (último volume na direção Z);
- bowall indica caixa de seleção caso haja ou não parede em Zmín (primeiro volume na direção Z);
- upisol indica caixa de seleção caso a parede seja adiabática ou não;
- t_up indica temperatura em y=ytotal;
- doisol indica caixa de seleção caso a parede seja adiabática ou não
- t_do indica temperatura em y=1
- topisol indica caixa de seleção caso a parede seja adiabática ou não;
- t_top indica temperatura em z=ztotal
- boisol indica caixa de seleção caso a parede seja adiabática ou não;
- t_bo indica temperatura em z=1;
- les indica caixa para seleção do modelo de turbulência LES;
- lesvd indica o uso da função de amortização para LES, portanto, esta caixa de seleção é dependente da seleção de LES;
- “as” indica caixa para seleção do modelo de turbulência SA;
- Des indica caixa para seleção do modelo de turbulência DES;
- Cs indica constante de smagorinsky, este valor será setado para o caso, somente se tiverem sido selecionadas as variáveis LES e LESVD;
- CDes indica a constante em Des, portanto, este valor será setado somente se DES estiver selecionado;
- start_up indica que, caso haja interrupção na simulação, deve-se selecioná-lo e indicar o valor de iterações para que seja reinicializada a simulação após aquele valor;
- termic indica seleção caso se use a equação de energia ou não, ou seja, caso seja isotérmico ou não;
- period indica caixa para seleção caso seja periódico em y ou não;
- lag indica seleção caso use fronteira imersa ou não;

- arquivo stl indica caixa de seleção caso se use arquivos stl para fornteira imersa;
- arquivo nos indica caixa de seleção caso use arquivos provenientes do Ansys para fornteira imersa;
- alpha do Sip indica o fator de relaxamanto do SIP.

No preenchimento deste formulário deve-se estar atento à interdependência entre algumas variáveis, pois como especificado na explicação de cada uma, a configuração da seqüência de informações interfere na configuração geral do caso. Abaixo aprasenta-se a pagina de configuração das condições de contorno, visualizada na figura 4.6.

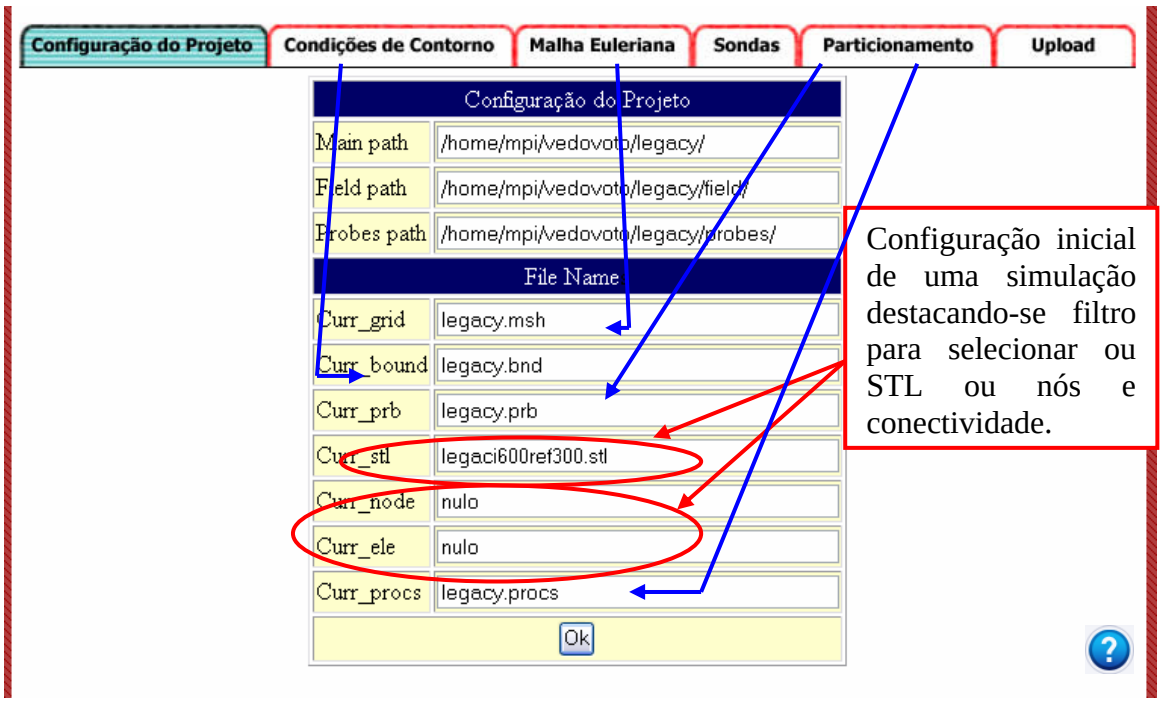


Figura 4.5. Primeira página na configuração de simulação de um escoamento: gera arquivo com as configurações principais do projeto.

Alguns detalhes são característicos de todos os formulários, como por exemplo, ao se passar o mouse sobre um determinado campo com o nome da informação a ser preenchida, uma caixa de texto explicativa aparecerá. Outra característica valida para cada formulário é que, ao se enviar os dados do fromulário preenchido, os mesmos são visualizados numa tela semelhante, porém com a opção para marcar algum campo que se deseje corrigir, e no final do formulário a opção para corrigir caso haja informações erradas, ou a opção para gerar o arquivo desejado, que resultará em uma mensagem de êxito.

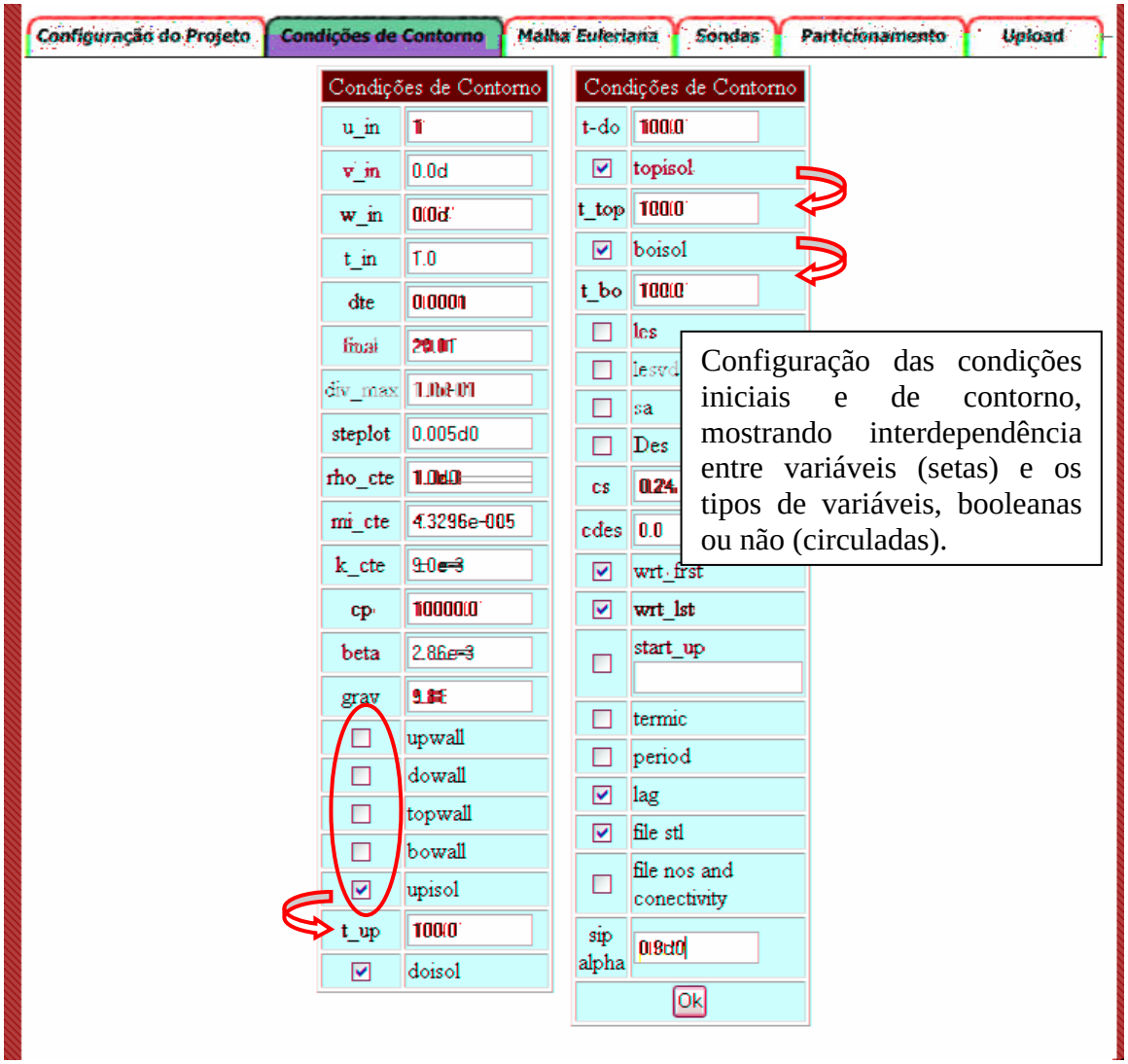


Figura 4.6. Segunda página na configuração de simulação de um escoamento: gera arquivo com as condições de iniciais e de contorno.

Assumiu-se que “zona” caracteriza uma região da malha do domínio que possui características de tamanhos distintos para cada região determinada. Para o arquivo de malha euleriana dever-se-á especificar as seguintes características:

- Tamanho da Malha euleriana nas posições x, y e z;

- Numero de Zonas em x, y e z, sendo que para cada zona, deverá se especificar as seguintes características:

  - Fim da zona;
  - Número de malhas;
  - Taxa de expansão ou relaxamento;
  - Sinal para indicar contração “+”, ou expansão “-”.

Visualizando o formulário para preenchimento das características da malha

euleriana, figura 4.8, vê-se que não existe um campo a ser preenchido para determinar o número de zonas existentes em x, y e z. O valor do número de zonas para cada variável especificada será determinado pelo número de vezes que o usuário enviar as informações configuradas naquela zona. Portanto, se o usuário preencher e enviar as informações das zonas quatro vezes em x, este será o número de zonas existente para x, e da mesma forma para as outras.

Uma consideração importante a ser feita é que o usuário preencha os formulários na ordem em que eles estão dispostos, e que cada formulário seja enviado da mesma forma. Assim, no caso do formulário da malha euleriana, primeiramente envia-se as informações com o tamanho da malha, depois quantas zonas em x forem necessárias, assim como em y e z.

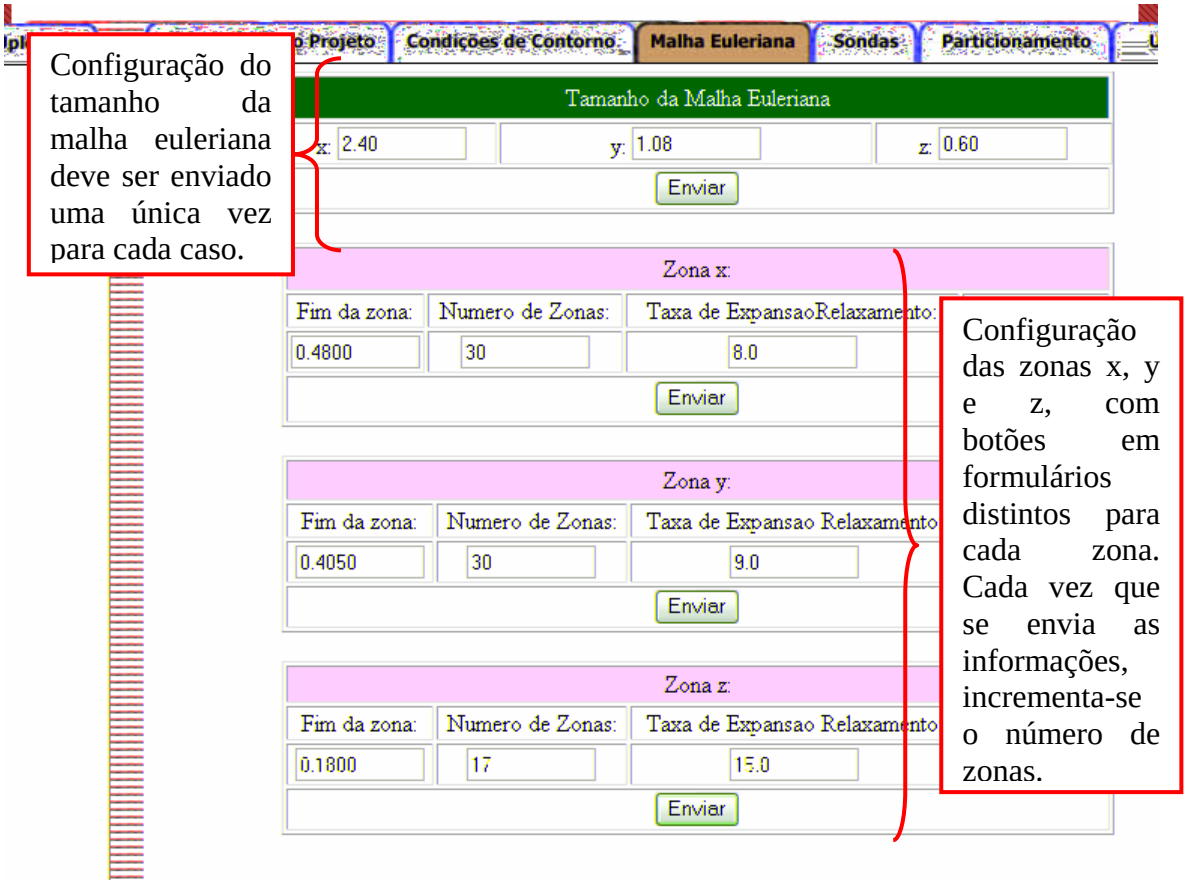


Figura 4.7. Terceira página na configuração de simulação de um escoamento: gera arquivo com as características da malha euleriana.

A próxima etapa na simulação é a configuração do arquivo de sondas, onde se indica a configuração das sondas pontuais e planares, na figura 4.8, que caso existam, seguirão as seguintes características:

- Número de sondas pontuais, sendo que cada sonda tem as seguintes especificações:

- Início da sonda em i, j, k;
- Posição da sonda em x, y, z;
- Caixa de seleção da variável podendo ser: u, v, w, p, T, ρ;
- Freqüência.

- Número de sondas planares sendo que cada sonda contém as seguintes especificações:

- plano onde está a sonda, sendo que quando gerado o arquivo, as variáveis assumirão os seguintes valores: x = 1, y = 2 e z = 3;
- posição na direção escolhida;
- passo de tempo para salvar as informações deste plano.

O preenchimento deste formulário segue a mesma metodologia de envio das informações do formulário que foi especificada na página da malha euleriana. Sendo assim, quantas vezes forem enviadas as informações das sondas pontuais, este será o número de sondas existentes, e da mesma forma para as sondas planares.

Para as sondas planares, deve-se seguir a seqüência de preenchimento para cada plano, ou seja, primeiro deve-se enviar quantas sondas desejadas perpendicular ao plano x, depois para y e da mesma forma para z.

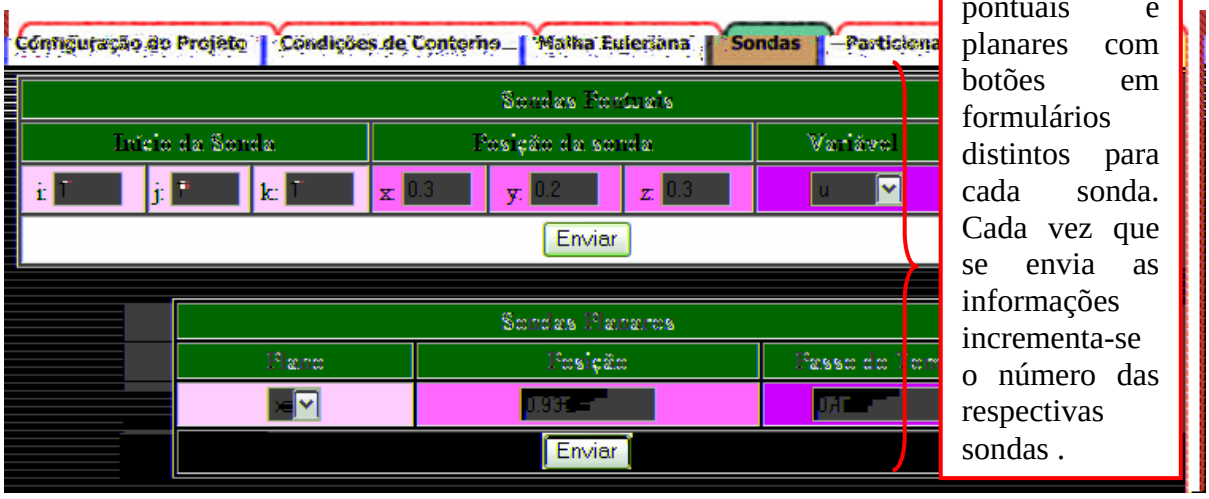


Figura 4.8. Quarta página na configuração de simulação de um escoamento: gera arquivo com as características das sondas (Pontuais e Planares).

Na etapa que segue, configura-se o arquivo de particionamento dos domínios, escolhendo a quantidade de processadores usados no cluster para simular o caso, figura 4.9. O arquivo conterá:

- Número de processadores usados, que é incrementado à medida que se envia as informações de particionamento para o processador, sendo que para cada processador

dever-se-á especificar a seguinte informação:

- Inicio do domínio em x, y e z;
- Fim do domínio em x, y e z;
- Número de volumes em x, y e z;

- Valores para transladar a fronteira imersa para a posição nos domínios x, y e z.

A última consideração a ser feita referente ao particionamento, é que o posicionamento da geometria em relação à fronteira imersa deve ser enviado somente uma vez.

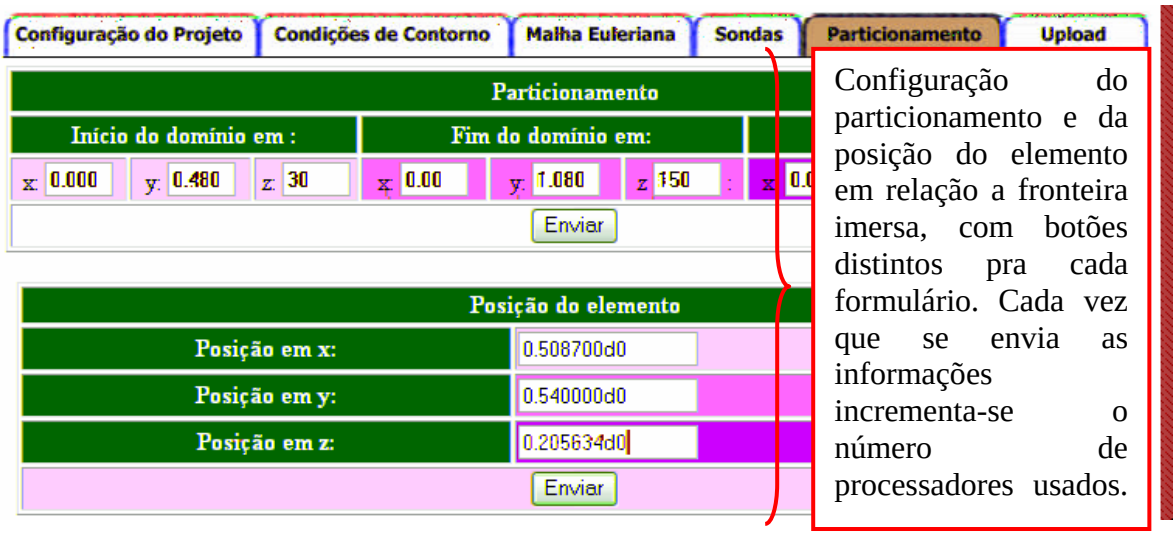


Figura 4.9. Quinta página na configuração de simulação de um escoamento: gera arquivo com as características do particionamento entre os processadores.

Para finalizar a configuração para posteriormente poder executar a simulação em CFD, alguns arquivos deverão ser enviados (*uploaded*) para o servidor, para tanto, o usuário deverá possuí-los a fim de agilizar a configuração do caso. Entre os arquivos, deverão ser: ou o arquivo que compõe a malha lagrangiana representando a geometria, ou os arquivos com as características da geometria porém sendo eles, os arquivos de nós e os de conectividades, conforme ilustrado na figura 4.10.

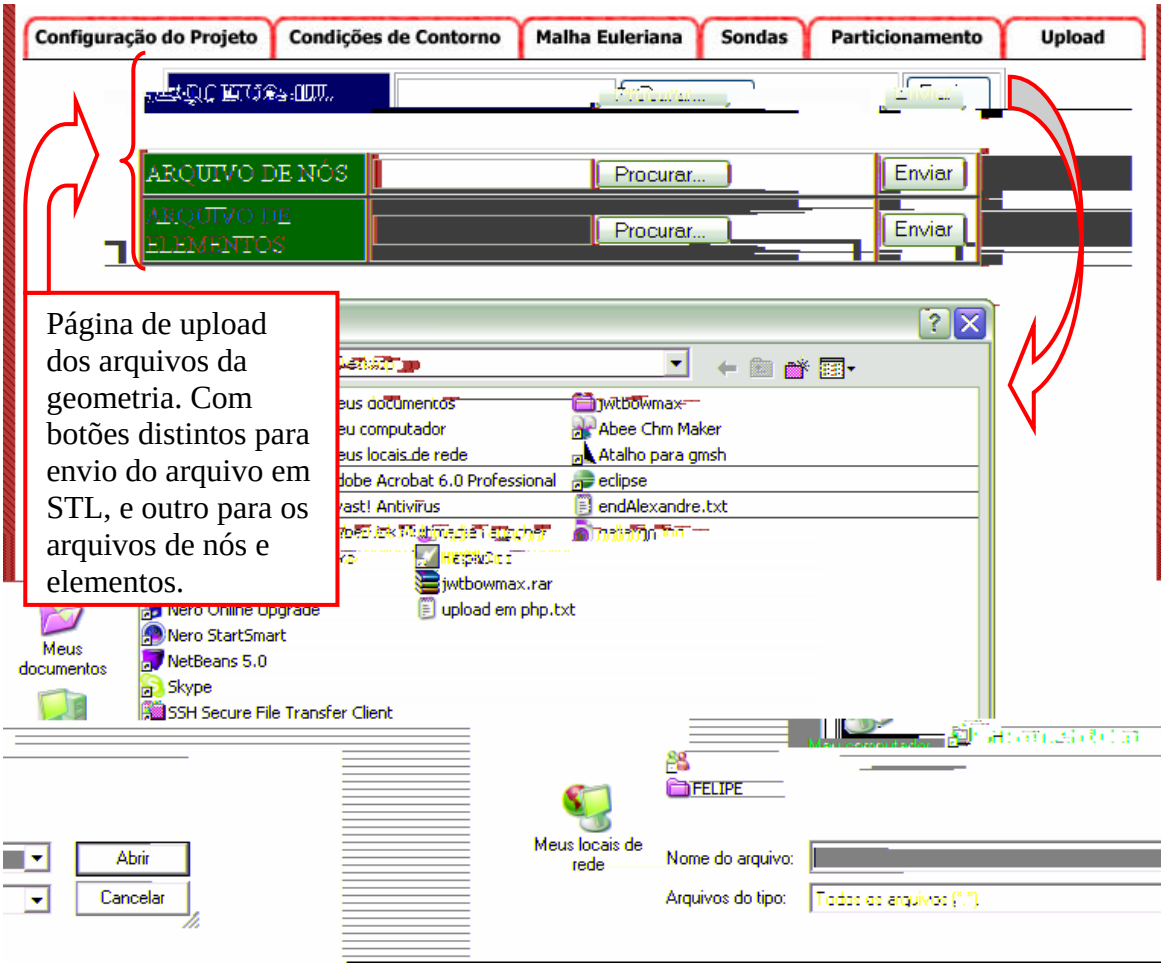


Figura 4.10. Última página na configuração de simulação de um escoamento: envia arquivos de malhas para o servidor.

Em relação a página de *upload* de arquivos, deve-se fazer a mesma consideração assumida na página de condições de configuração do projeto, ou seja, caso se o utilize o arquivo no formato STL, os arquivos de nós e elementos não deverão ser enviados.

4.2. Validando as malhas geradas pelo GMSH

Para analisar a topologia de escoamentos sobre geometrias simples, serão utilizados alguns parâmetros, efetuando comparações dos resultados ao utilizar o GMSH para gerar malhas, comparadas com malhas utilizando pacotes comerciais para a mesma geometria.

Analisando as características dos elementos triangulares da malha, tomando por base uma esfera gerada por um software comercial, figura 4.11, observa-se que esta esfera é descrita por triângulos, em sua maioria eqüiláteros, de forma a fornecer uma malha o mais homogênea possível.

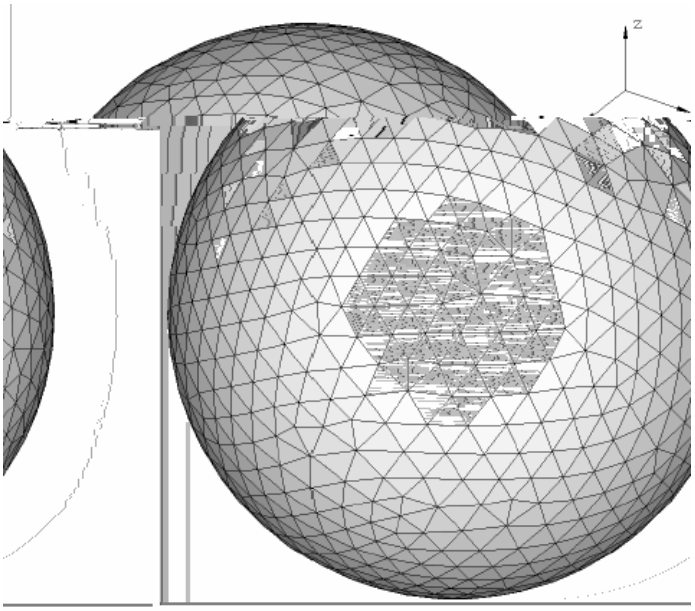


Figura 4.11. Malha de esfera gerada por software comercial; raio = 0,02m.

Nos resultados aqui apresentados, verificar-se-á que, mesmo sendo a malha gerada pelo GMSH, figura 4.12, tem-se que a malha superficial é composta por elementos triangulares que possuem diferença de tamanho nos seus elementos, essas diferenças não afetam significativamente os resultados alcançados com a simulação do escoamento, visualizados mais adiante.

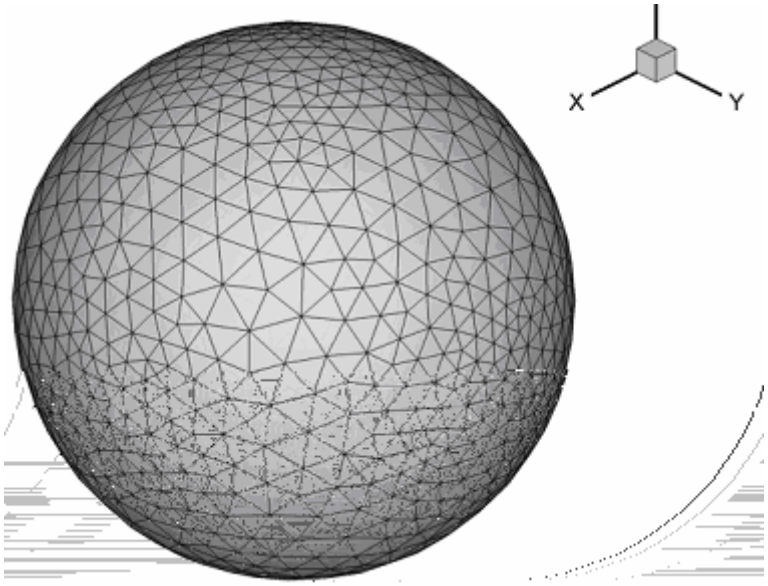


Figura 4.12. Malha de esfera gerada no GMSH; raio = 0,02m.

A Justificativa para esse tipo de deformidade encontrada na malha gerada pelo GMSH é devido à geometria ser estruturada por composição de faces, como mostrado no capitulo anterior. Além desse fato, tem-se que o arquivo STL é gerado sobre a composição da malha. Portanto o refinamento da malha adotado na geração da geometria, como

mostrado na figura 4.13 abaixo, também influencia na não uniformidade da malha. Um exemplo bem visível é mostrado nas figuras 4.13 “c” e “d”, onde são visualizados o encontro de quadrantes da esfera, sendo estes locais onde há maior deformação da malha, justificando a formação de superfícies com variação do elemento triangular da malha.

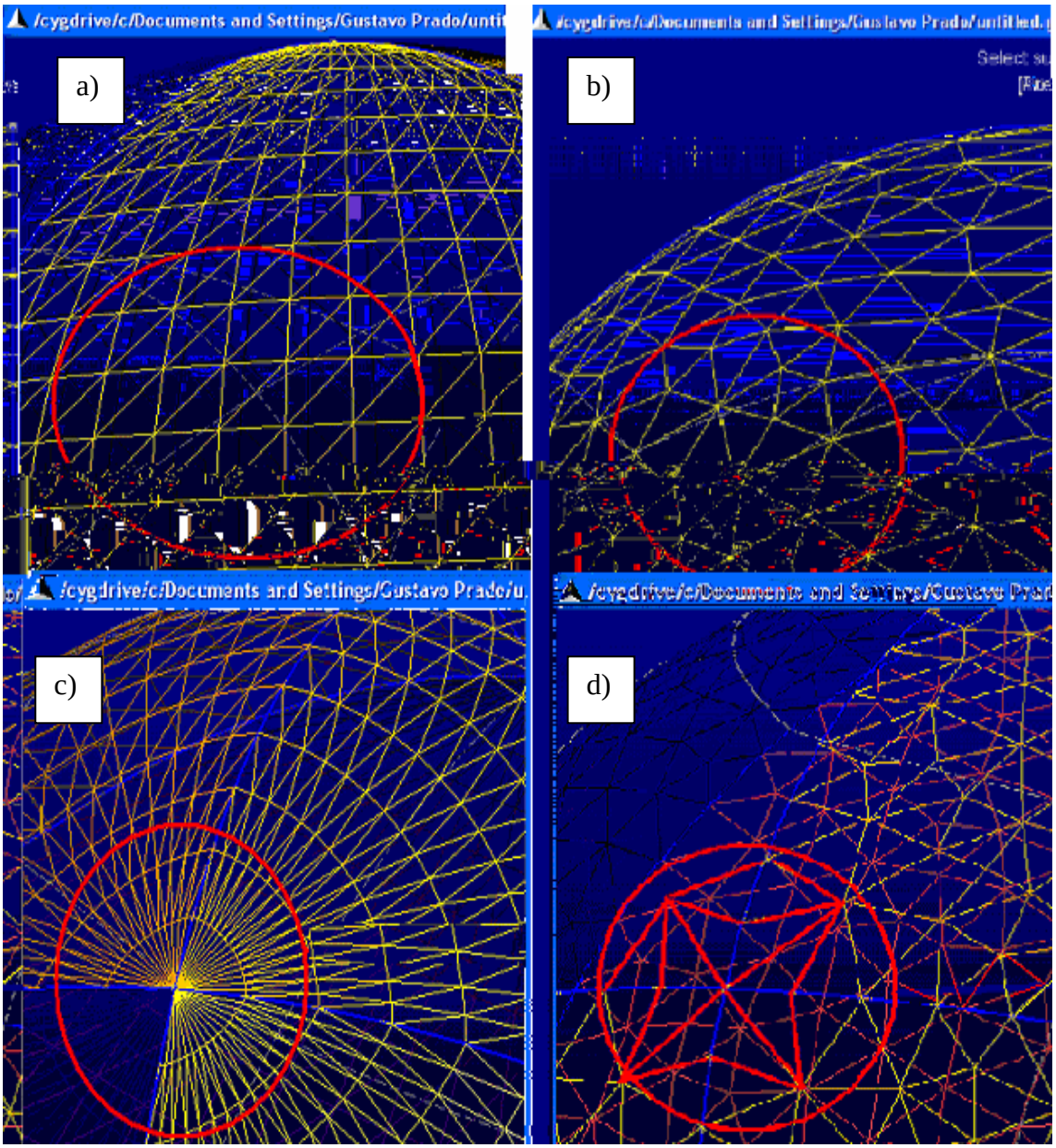


Figura 4.13. Não uniformidades causadas por composição de superfícies: a) centro da malha com refinamento; b). centro da malha sem refinamento; c) extremidades da malha com refinamento; d) extremidades da malha sem refinamento.

Após ter sido avaliado a qualidade das malhas geradas, deve-se averiguar a atuação da simulação computacional sobre as mesmas, verificando como se formam as estruturas

trubilhonares do escoamento. Para tal avaliação, um importante parâmetro que pode ser empregado para visualizar o escoamento é o critério Q, Eq. 4.1 (Jeong e Hussain, 1995), definido da forma:

$$Q = \frac{1}{2}\left[\left|\Omega\right|^2 - \left|S\right|^2\right] > 0 \ , \tag{4.1}$$

onde *S* e Ω, são as componentes simétrica e assimétrica do gradiente de *u,* portanto Q representa o balanço local entre a taxa de deformação e a vorticidade, ou seja, a norma Euclidiana para a qual o tensor rotação supera a taxa de deformação (Haller, 2005), definidos respectivamente como:

$$\Omega = \frac{1}{2}\left[\nabla v - (\nabla v)^T\right], \tag{4.2}$$

$$S = \frac{1}{2}\left[\nabla v + (\nabla v)^T\right]. \tag{4.3}$$

Sendo assim, o critério Q é uma das várias ferramentas usadas para analisar as propriedades das estruturas turbilhonares geradas no escoamento. Sua principal vantagem é mostrar as estruturas que adquirem mais peso devido ao efeito rotacional, permitindo visualizar as regiões do escoamento com fortes deformações do fluido.

O parâmetro adimensional número de Reynolds (Re), Eq. 4.4, foi empregado para caracterizar o regime de escoamento do fluido sobre a esfera. O *Re* foi definido baseado no diâmetro D da esfera na forma:

$$\mathrm{Re}_D = \frac{\rho U_\infty D}{\mu}, \tag{4.4}$$

onde U_∞ é a velocidade da corrente livre do escoamento, ρ e μ são a massa específica e a viscosidade dinâmica do fluido, respectivamente.

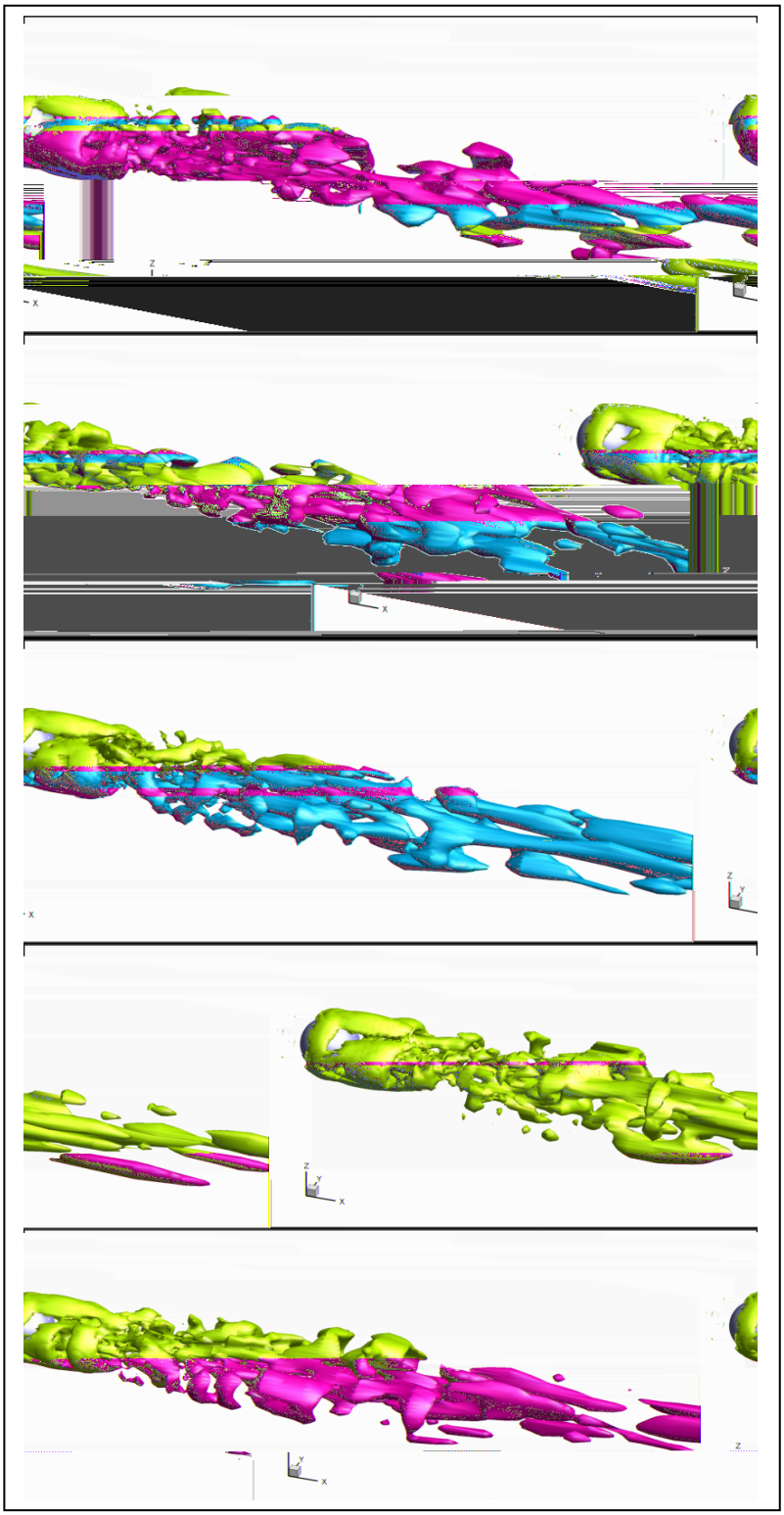


Figura 4.14. Detalhe dos vórtices sobre a esfera a Re = 1.000 evidenciado pelo critério Q (Vedovoto, 2007); malha gerada com pacote comercial.

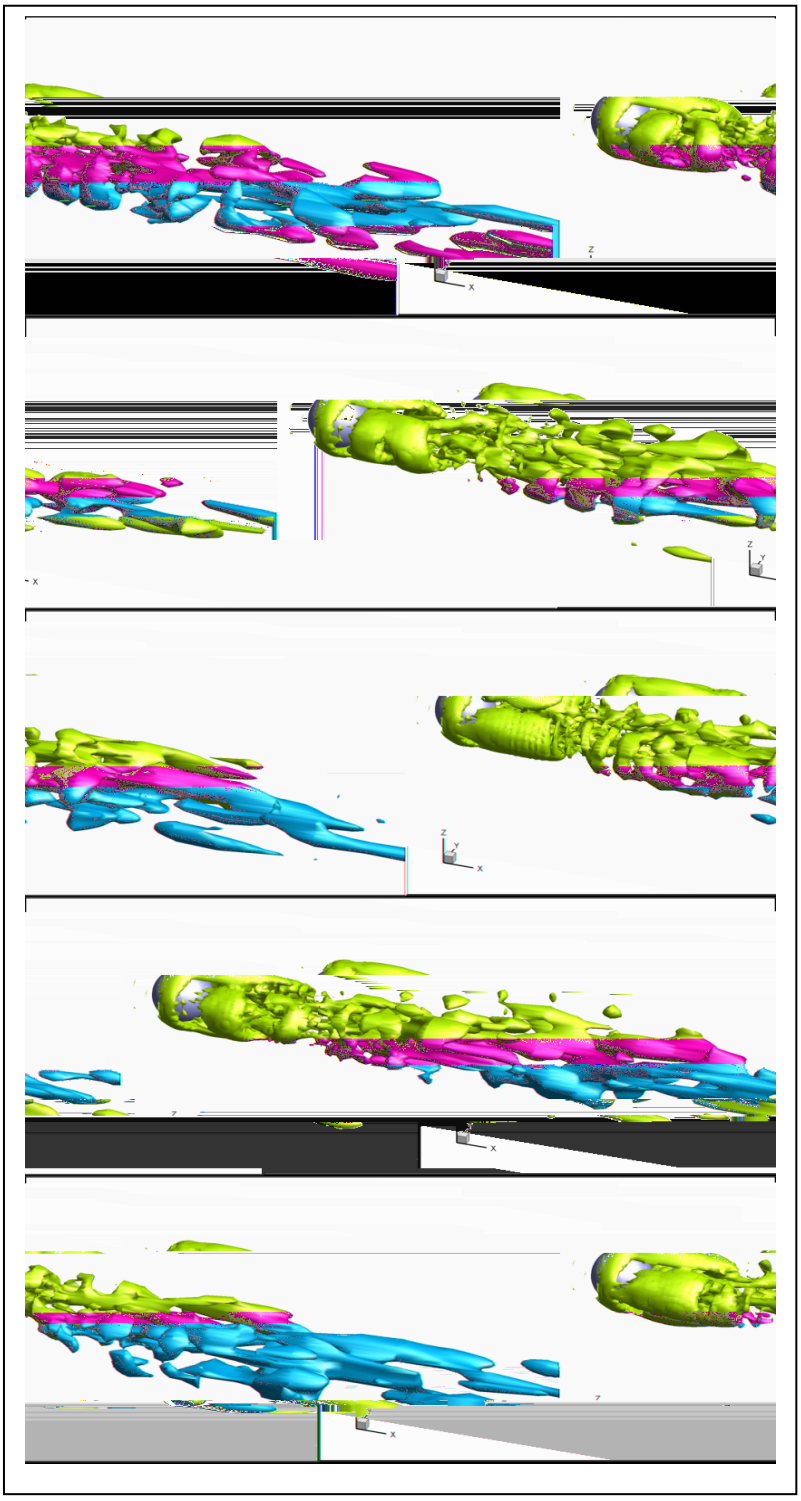


Figura 4.15. Detalhe dos vórtices sobre a esfera a Re = 1.000 evidenciado pelo critério Q (Vedovoto, 2007), malha gerada com GMSH.

As figuras acima citadas e a figura 4.16 mostram isosuperfícies de Q = 10, para o escoamento a jusante de uma esfera, simulado com malhas gerada com o GMSH e com software comercial. A visualização efetuada nas figuras, foram elaboradas utilizando-se TecPlot. Observa-se a figura 4.16 ilustrando semelhança entre os resultados baseados nas isoQ (Vedovoto, 2007).

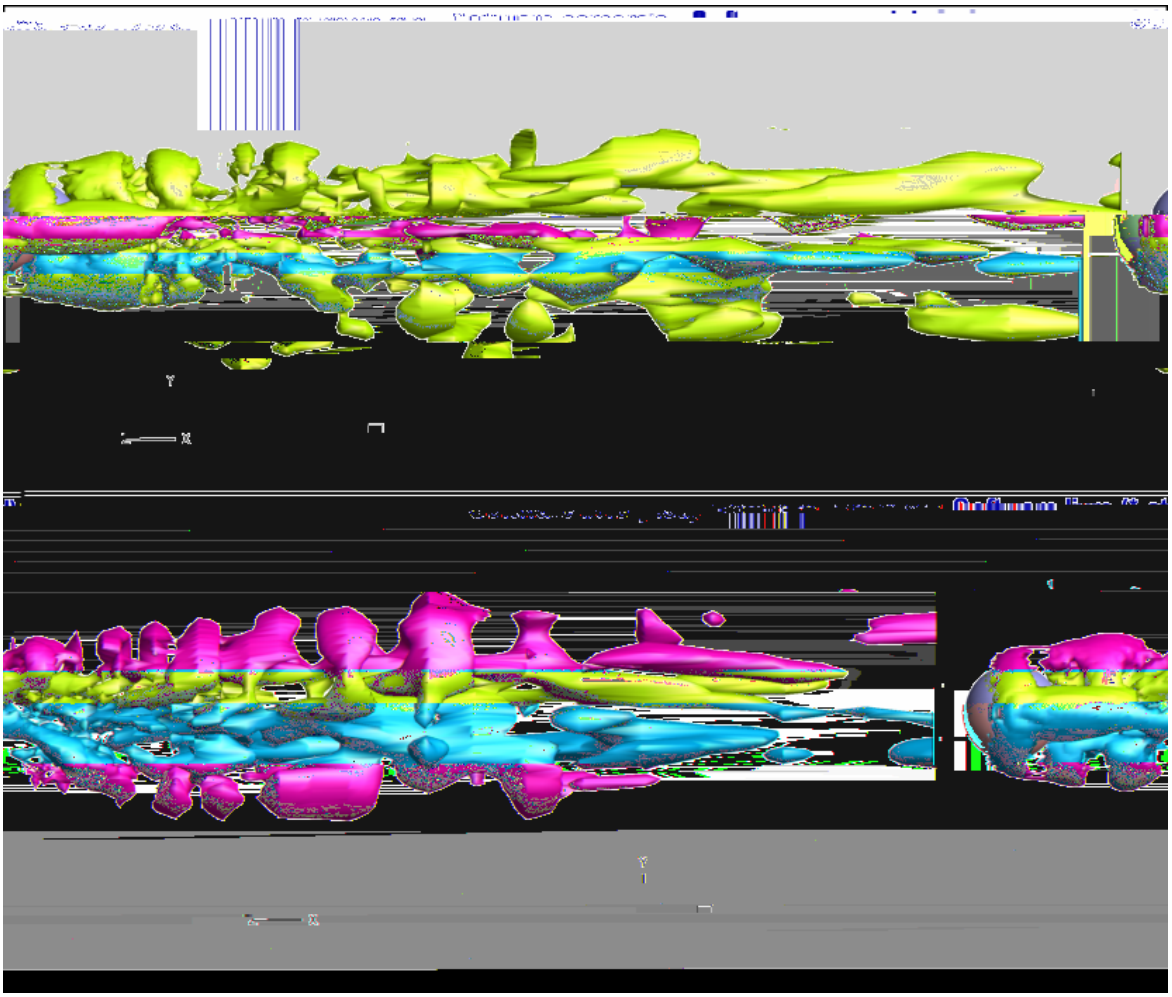


Figura 4.16. Estruturas turbilhonares geradas a jusante de uma esfera, vistas no plano XY, em t = 7,0s (Vedovoto, 2007).

O coeficiente de arrasto é definido abaixo, Eq. 4.5:

$$C_D = \frac{F_X}{\left(\frac{1}{2}\right)\rho U_\infty{}^2 \left(\frac{\pi D^2}{4}\right)}, \qquad (4.5)$$

sendo Fx a força de arrasto, agindo sobre o corpo imerso (no caso, a esfera), obtida no

presente trabalho a partir do Modelo Físico Virtual (Campregher, 2005). É interessante salientar que seu valor é uma resposta direta da metodologia de fronteira imersa, não sendo necessário nenhum procedimento adicional para a avaliação da força de arrasto. Esta é uma forma totalmente diferente de se avaliar a força de arrasto (ou de sustentação) em um corpo, pois, nas metodologias tradicionais, é necessário obtê-la de forma indireta como, por exemplo, a partir da distribuição de pressão e de tensões cisalhantes na superfície do corpo.

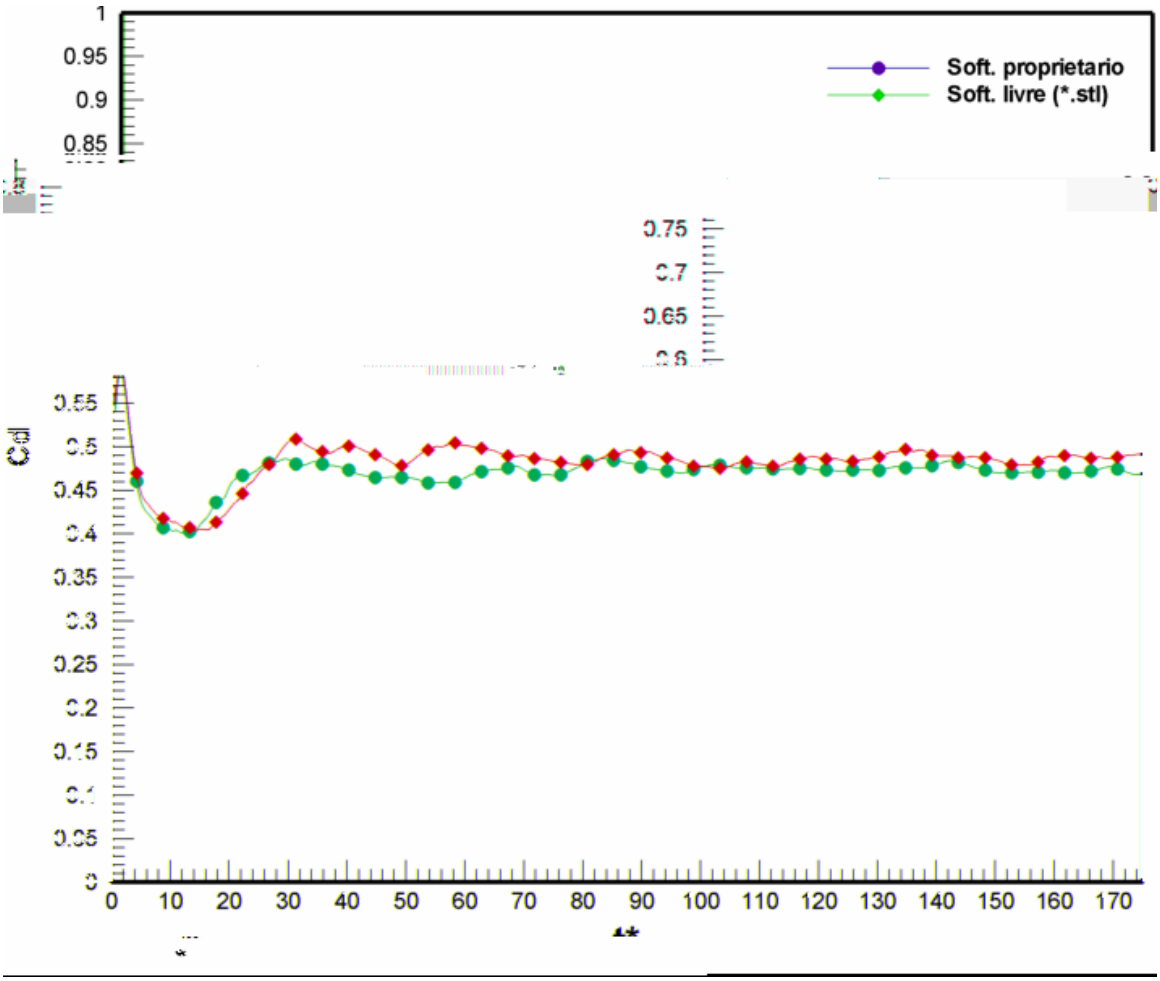


Figura 4.17.Coeficiente de arrasto C_D a Re = 1.000, em função de t*,(Vedovoto 2007).

O tempo adimensional t^* é dado por:

$$t^* = \frac{t \cdot U_\infty}{D},$$

onde t é o tempo físico e D é o diâmetro da esfera.

A partir da Figura 4.17 acima, pôde-se perceber que os resultados para o coeficiente de arrasto são bastante satisfatórios para o número de Reynolds simulado, tanto para a

malha usada no pacote comercial quanto para o GMSH (Oliveira et al., 2006). Como dito no capitulo anterior, a interface utilizada não se vale de pós-processamento, sendo assim, os gráficos citados neste capitulo foram produzidos com software proprietário (TecPlot) após se ter extraídos todos os dados calculados pelo solver, realizando um download destas informações, e passando para o software para realizar o pós-processamento.

Outros parâmetros igualmente importantes, obtidos de maneira semelhante ao coeficiente de arrasto, são o coeficiente de sustentação (*lift coefficient* – C_L) e o coeficiente lateral (*side coefficient* – C_S) avaliados, respectivamente, com a Eq. 4.6 e com a Eq. 4.7:

$$C_L = \frac{F_Z}{\left(\frac{1}{2}\right)\rho U_\infty{}^2 \left(\frac{\pi D^2}{4}\right)}, \tag{4.6}$$

$$C_S = \frac{F_Y}{\left(\frac{1}{2}\right)\rho U_\infty{}^2 \left(\frac{\pi D^2}{4}\right)}, \tag{4.7}$$

onde *Fy* e *Fz* são a somatório das forças que atuam sobre a esfera nas direções y e z, respectivamente.

Nas Figuras 4.18 e 4.19, respectivamente, pode-se visualizar os resultados de CL e CS para Re = 1.000, indicando o comportamento aleatório da emissão de estruturas turbilhonares à jusante da esfera. Neste gráfico especificam-se as regiões do escoamento onde se encontram fortes áreas de deformação do fluido.

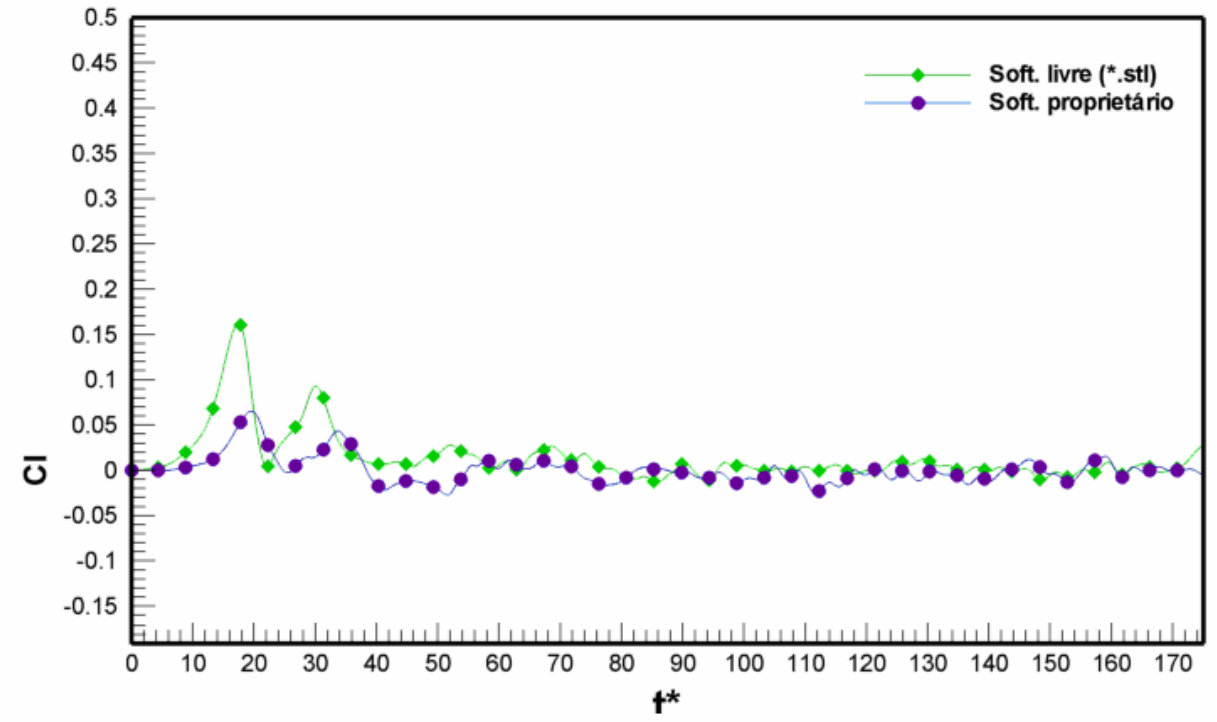


Figura 4.18. Coeficiente de sustentação (C_L) para Re = 1.000, (Vedovoto, 2007).

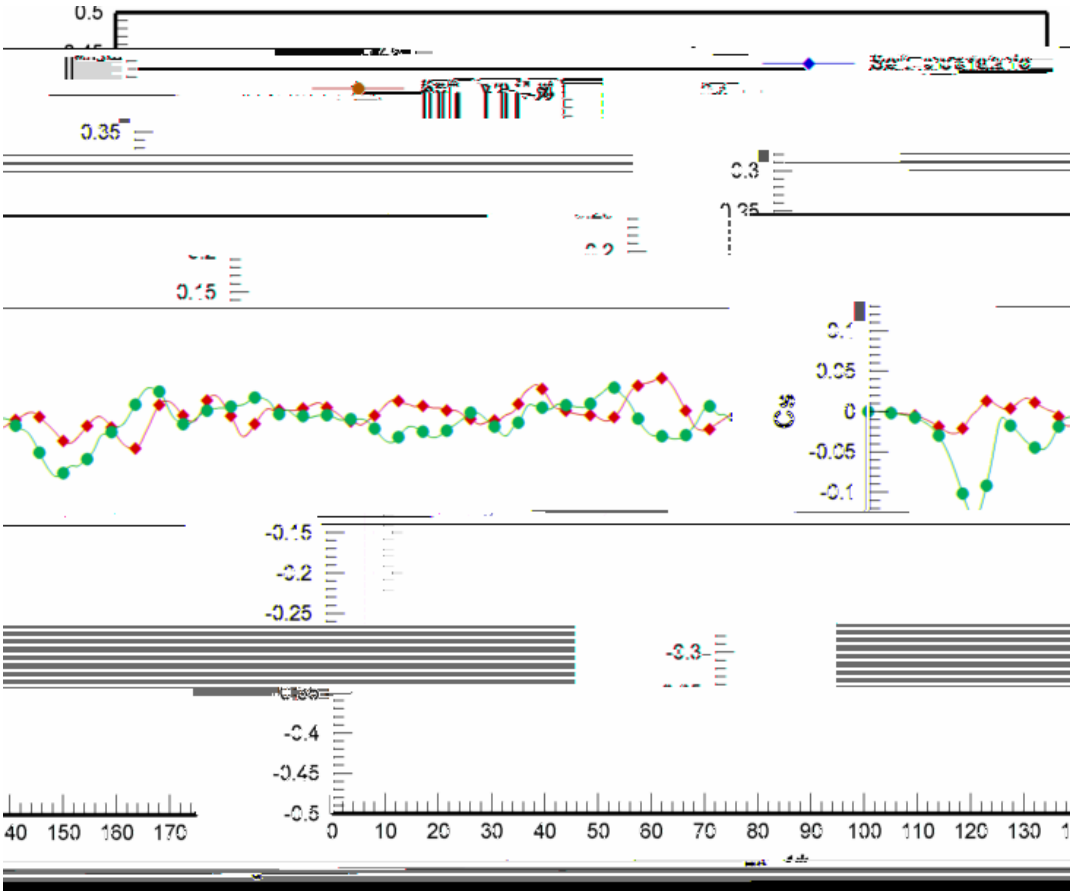


Figura 4.19 - Coeficiente lateral (C_S) para Re = 1.000 (Vedovoto, 2007).

Avaliando estatisticamente, os valores para o coeficiente de sustentação gerados a partir de dois softwares geradores de malha, nota-se que são em média compatíveis, porém para o coeficiente lateral podemos notar maiores variações devido ao mesmo ser mais sensível ao tipo de malha gerada, uma vez que o GMSH não consegue gerar uma malha feita com elementos completamente iguais (triângulos eqüiláteros), havendo nesta malha variações de tamanho dos triângulos. Contudo, pode-se verificar que nos cálculos realizados, os valores médios tendem a convergir para um valor semelhante, validando o resultado para esfera gerada pelo GMSH (Oliveira et al., 2006).

Um outro parâmetro importante a ser calculado sobre a geometria é o L_2, que é definido aqui como a raiz quadrada da norma da diferença entre a velocidade do fluido sobre a interface fluido/sólido e da velocidade desta interface (que no caso da esfera estacionária tem valor nulo). Este parâmetro tem por finalidade medir se há condição de não deslizamento sobre a interface está sendo bem modelada, através do campo de força calculado. Como esperado, L_2 deve tender a zero à medida que o tempo evolui, visto que a velocidade do fluido sobre a interface tende ao valor da velocidade da interface.

Na Figura 4.20, visualiza-se a evolução temporal do parâmetro L_2, para o

escoamento sobre a esfera onde o número de Reynolds é igual a 1.000.

A pequena diferença existente entre o L_2 produzido pela esfera feita com o pacote comercial, e o L_2 produzido pela esfera feita no GMSH, é devido a não uniformidade existente na malha produzida pelo GMSH, apesar de que esta última esfera possui aproximadamente duzentos pontos a mais que a do pacote comercial, representando assim uma área superficial um pouco maior que a malha da esfera do software comercial (Oliveira et al., 2006).

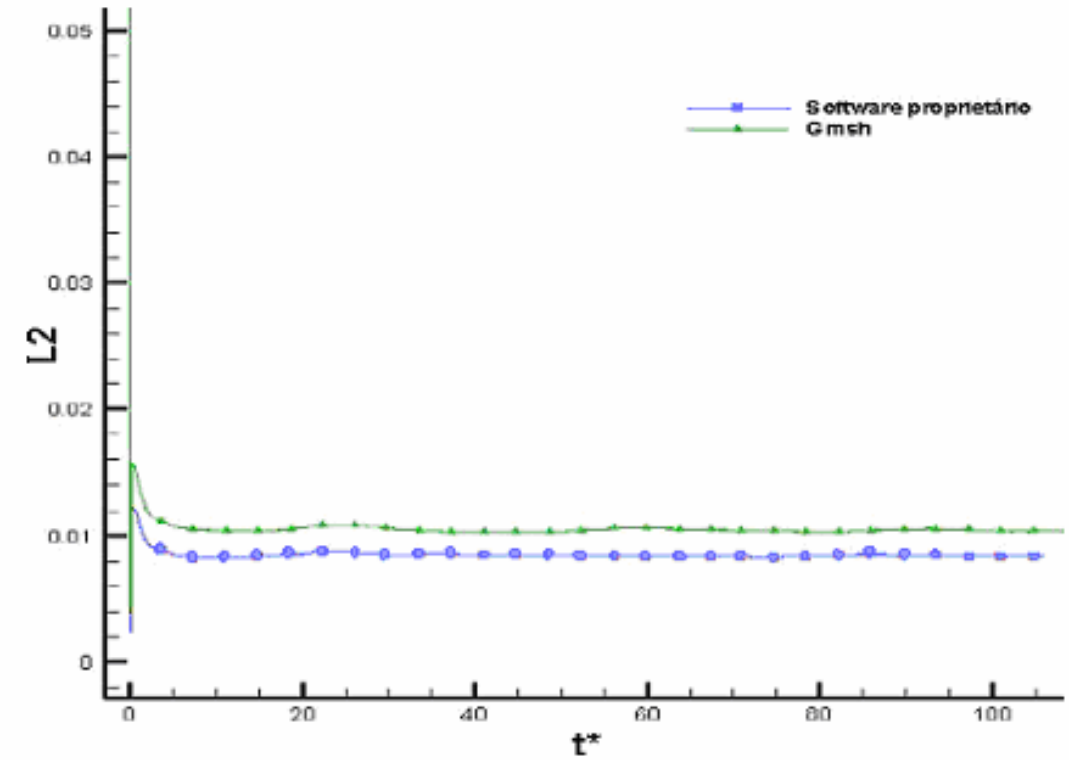


Figura 4.20 - Evolução temporal do parâmetro L_2 para Re = 1.000.

4.3 .Qualidade das Malhas

Como se pôde notar no tópico anterior, os resultados quantitativos calculados usando a malha de cada geometria, depende da qualidade que esta malha oferece ao código computacional.

Um fator desfavorável que o GMSH demonstra na geração das geometrias é gerar elementos geométricos por composição de superfícies, fragmentando assim a geometria em faces, como mostrado no item anterior, com as figuras 4.13 “c” e “d” Esta fragmentação gera bordas com deformações na malha, diminuindo a qualidade das mesmas. Mesmo diminuindo a qualidade das malhas, esta fragmentação em faces é um fator relativo à

geometria complexa que se queira produzir, pois se a geometria for compostas por faces quadradas, nas malhas geradas com o GMSH, essas irregularidades não aparecerão. Figura 4.21

Analisando as características dos elementos triangulares de malha, tomando por base um cubo gerado por um software comercial, figura 4.21, observa-se que este cubo é composto por triângulos na sua maioria eqüiláteros. Sendo assim, pode-se admitir ser uma malha de qualidade para obter resultados quantitativos.

Como mostrado anteriormente, há de se esperar que, a malha gerada no GMSH apresente algumas deformações nos encontros entre cada face da geometria. Porém, na análise da figura 4.22, pode-se verificar que o esperado não acontece, uma vez que a forma geométrica sobre a qual deve ser gerada a malha é favorável à triangularização.

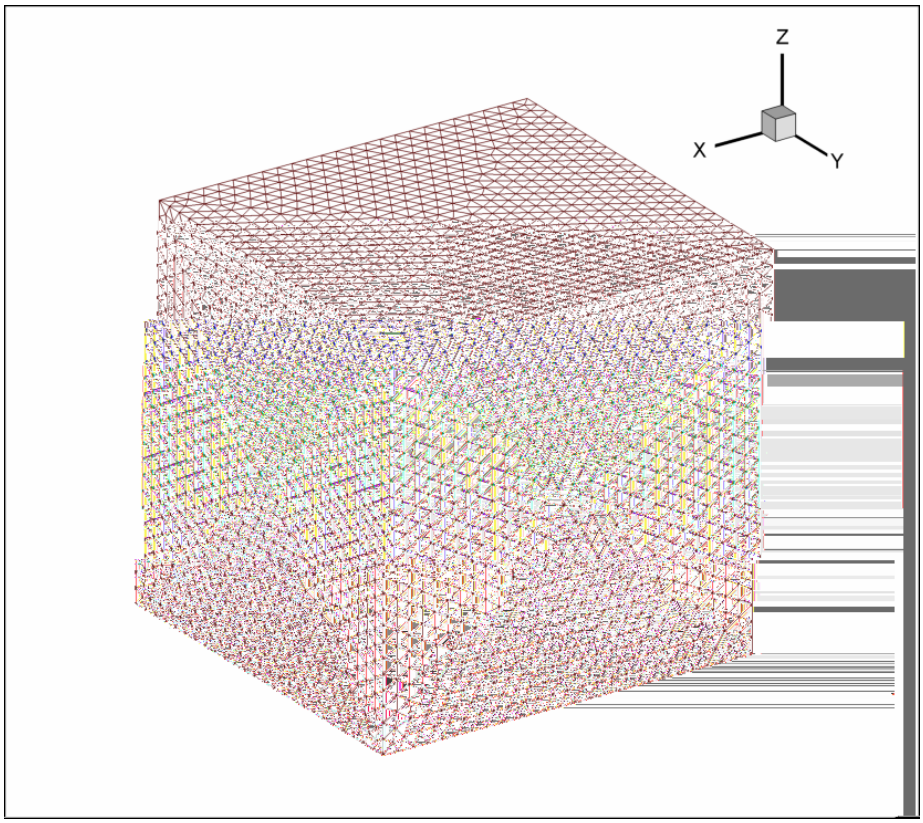


Figura 4.21. Malha de elementos triangulares representando um cubo (Vedovoto, 2007).

Portanto, ao se comparar as duas figuras, pode-se concluir que a malha gerada pelo GMSH é tão recomendada para a simulação computacional, quanto a malha gerada pelo software comercial.

Como vê-se abaixo, a figura é favorável a triangularização, e por conseqüência poderia ter sua face representada apenas por dois triângulos em cada face. Isso não ocorre pois na metodologia da fronteira imersa, para se reconhecer uma geometria através de um

campo de força, ela deve apresentar uma quantidade de pontos suficientes, de forma que cada ponto da malha lagrangiana se enquadre em um elemento da malha do domínio contíguo (malha euleriana).

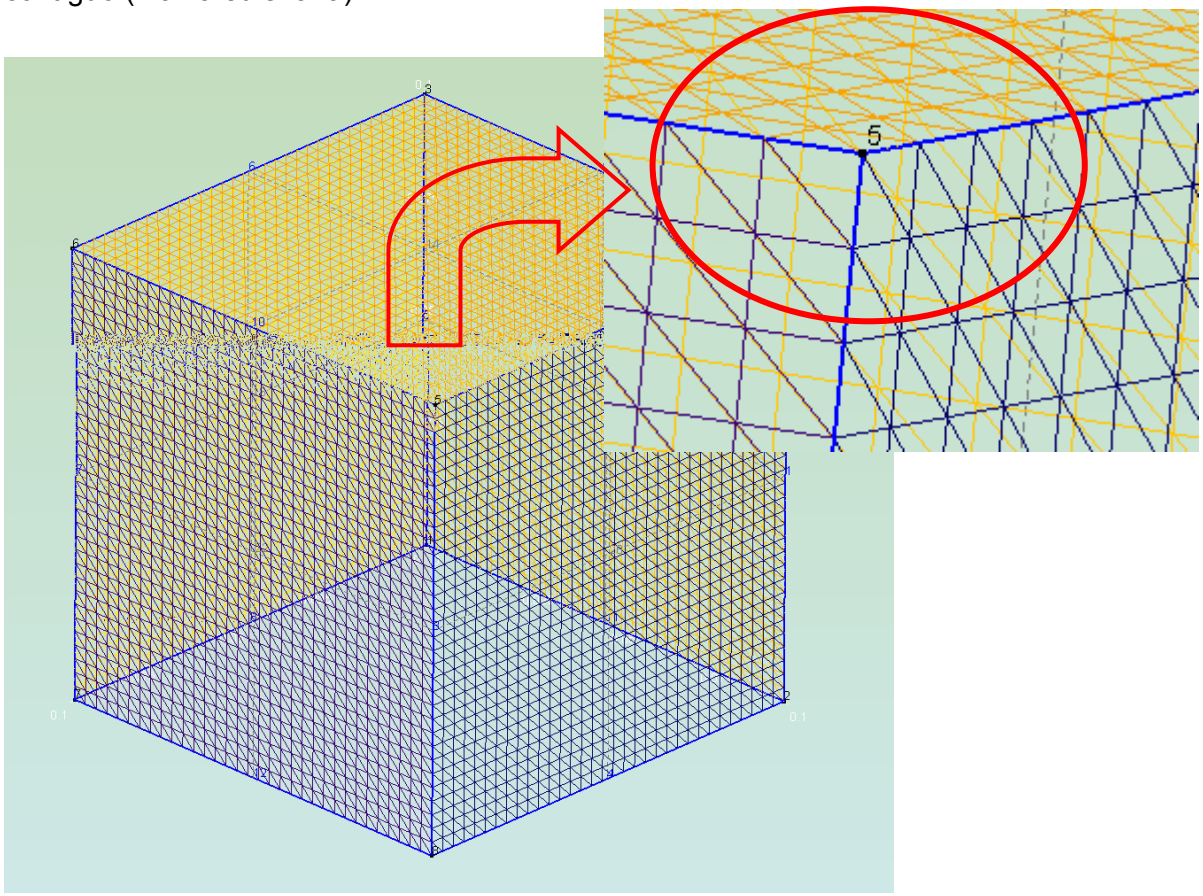


Figura 4.22. Malha gerada pelo GMSH, sem deformações nas faces e nas bordas.

4.3. Geração de geometrias complexas e validação das malhas

Nesta seção serão apresentadas as malhas de geometrias complexas, sendo uma delas a de um protótipo de automóvel e a outra, a de um protótipo de um avião. Essas geometrias servirão para mostrar o nível de complexidade que se pode alcançar, mesmo usando uma ferramenta gratuita geradora de malhas.

4.3.1. Protótipo de um automóvel

A geometria do protótipo em questão foi baseada em um automóvel esportivo, o Lamborghini Gallardo. Como foi explicado anteriormente, para se construir uma geometria desta complexidade, foi realizada uma consulta no *site* www.lamborghini.com, onde foram encontradas especificações referente ao comprimento, largura e altura do automóvel.

Maiores detalhes foram retirados através da visualização de fotografias do automóvel, de vários ângulos. A escolha deste carro como modelo, foi feita pelo fato dele possuir uma geometria aerodinâmica e ter uma altura em relação ao solo pouco significante, de forma que ao simular um escoamento sobre o mesmo, este poderia ser comparado com escoamento sobre geometrias simples que possuem resultados validados, por exemplo, o escoamento sobre um cubo apoiado no solo.

A malha de elementos triangulares que representa o protótipo, mostrada na figura 4.23, é composta por 11.486 nós e 22.953 elementos, que é uma malha bastante representativa para um objeto com escala em centímetros.

Apesar de ser um modelo simplificado de automóvel, cuidados foram tomados para que o protótipo tivesse características próximas da real, como a criação de rodas independentes, como mostrado na figura 4.24.

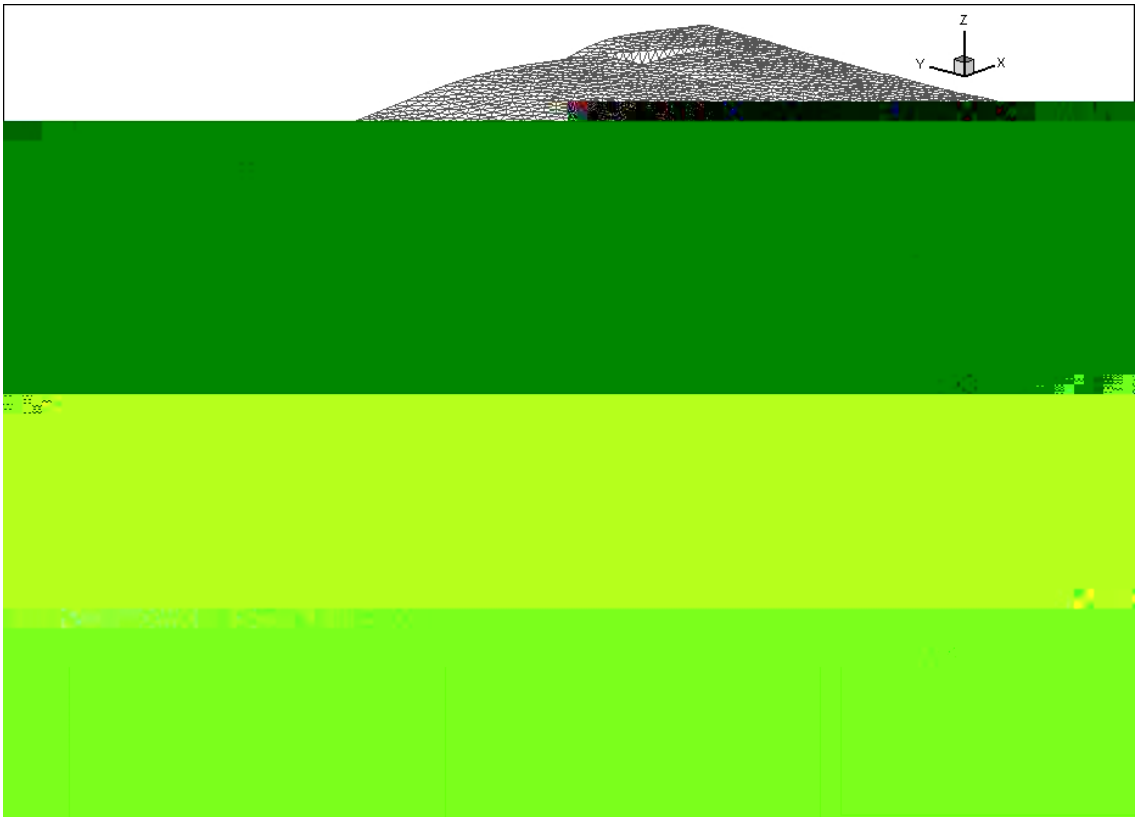


Figura 4.23. Malha de elementos triangulares representando um protótipo de automóvel.

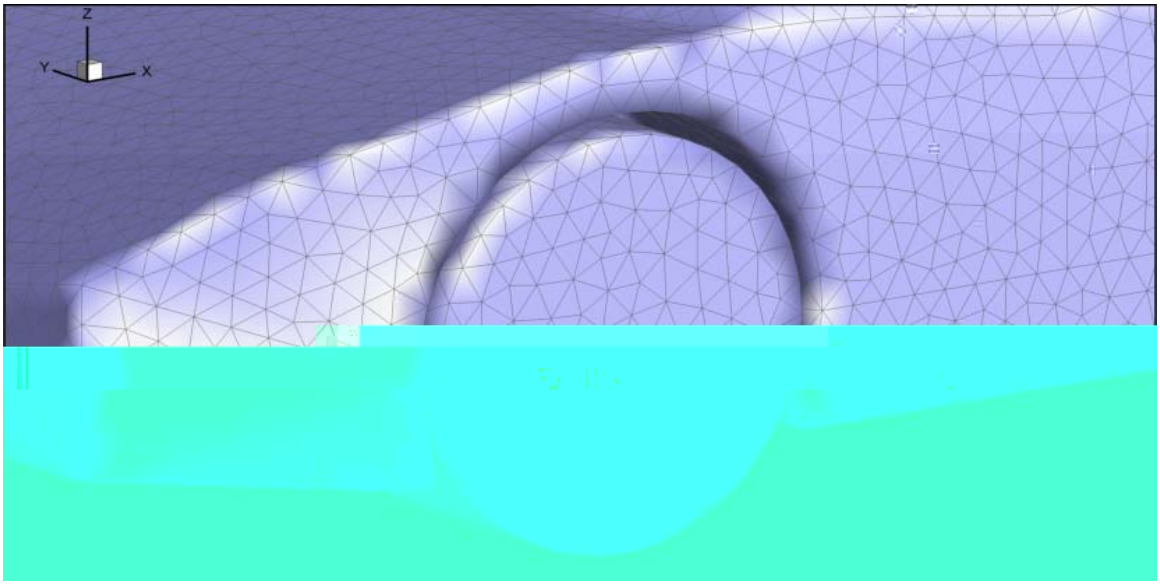


Figura 4.24. Detalhe da roda dianteira do protótipo do automóvel.

A composição da geometria do automóvel, como citado anteriormente, foi realizada através da comparação com imagens do modelo. A principio, foi traçado o perfil que assumiu a posição em um dos eixos do plano cartesiano, como mostrado na figura 4.25. Com exceção do bordo de ataque dianteiro (figura 4.27), toda a lateral do carro é uma geometria plana, compostas por linhas retas ou *splines* (Exemplos de splines: linhas em vermelho).

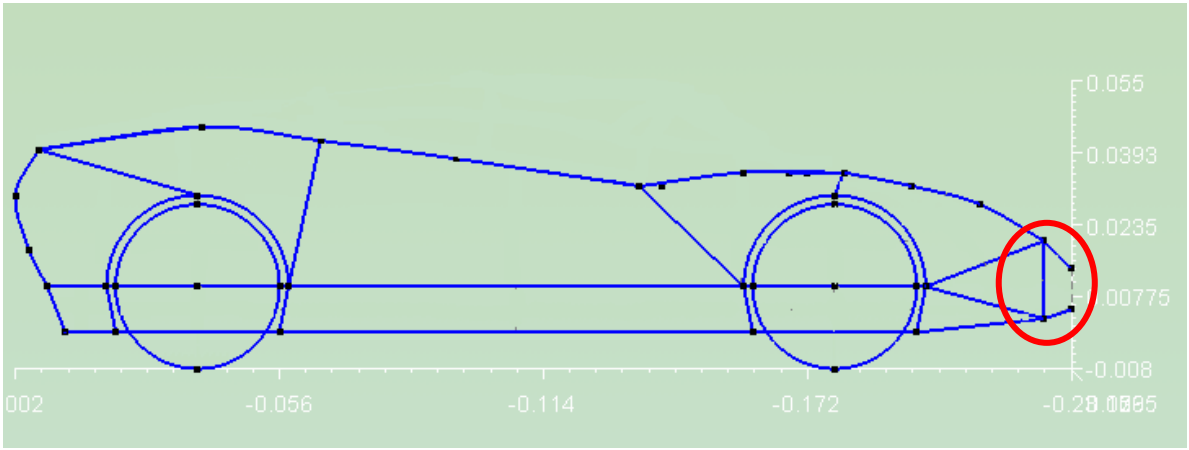


Figura 4.25. Perfil sobre o eixo z, parte plana do carro.

Após ter sido traçado o perfil do carro, a forma mais simples de construir o perfil do lado oposto, é transladando os pontos do perfil acima visualizado, em direção oposta ao eixo. Assim, obtém-se a outra face do protótipo, visualizado na figura 4.26.

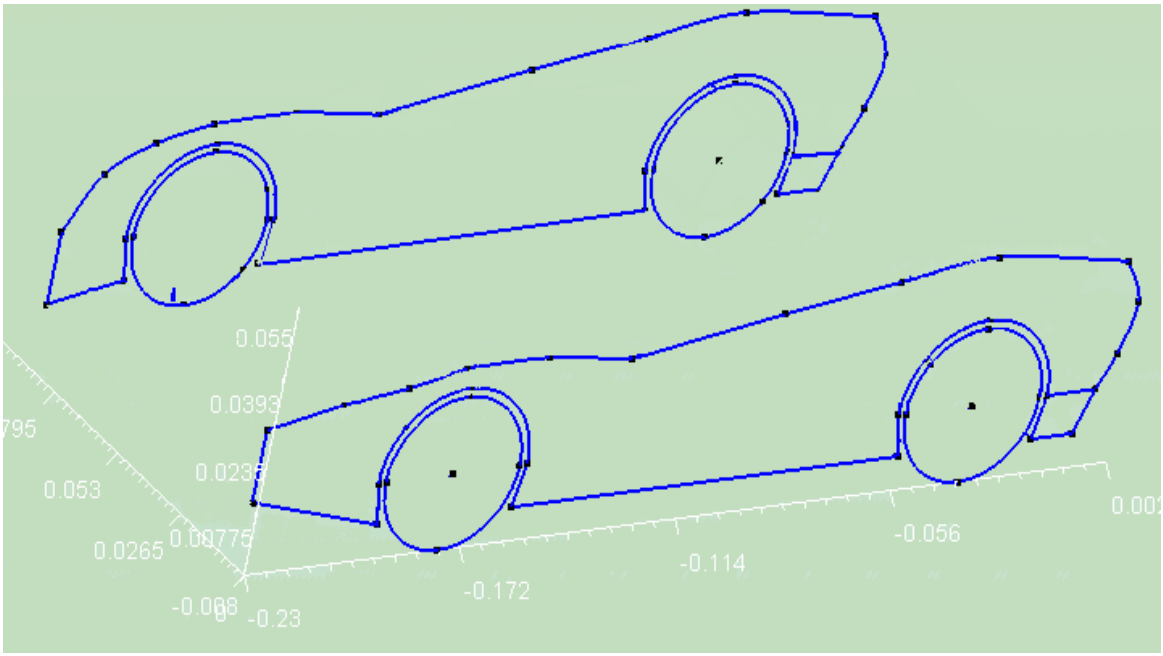


Figura. 4.26. Perfil do carro transladado ao longo do eixo z

O próximo passo para compor a geometria é unir as duas faces, utilizando pontos específicos que delimitam a geometria, como mostrado na figura 4.27. Nesta figura também está transladado o perfil tanto da roda quanto da caixa que a comporta.

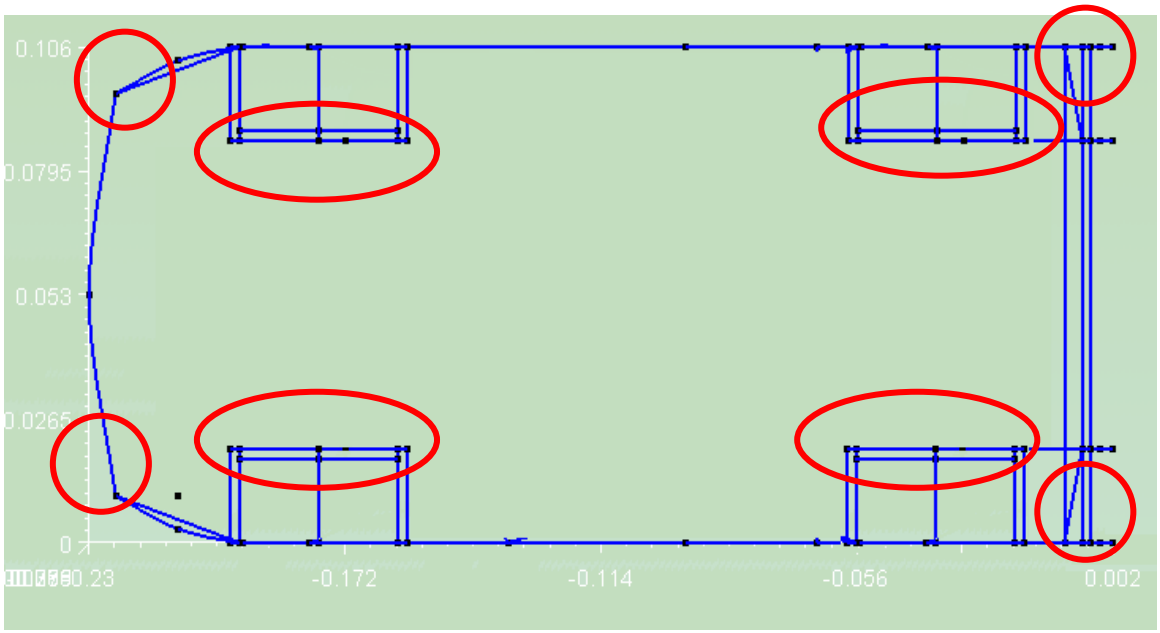


Figura. 4.27. Vista superior do chassi do carro, com a união das faces.

Finalizadas as partes consideradas mais simples na geração da geometria, segue-se para a vista superior do carro, onde se tem maior detalhe em superfícies curvas. Esta parte

requer maior atenção, despendendo de maior tempo na sua produção. A forma como foi produzida, segue o mesmo padrão da lateral do carro. São traçados pontos para delimitar o perfil mostrado na figura 4.28, em relação a altura do modelo original, figura 4.29. Maior cuidado deve ser tomado, pois esses pontos variam tanto em x quanto em z, obtendo como resultado a figura 4.30, em comparação a vista superior da imagem do original, na figura 4.31.

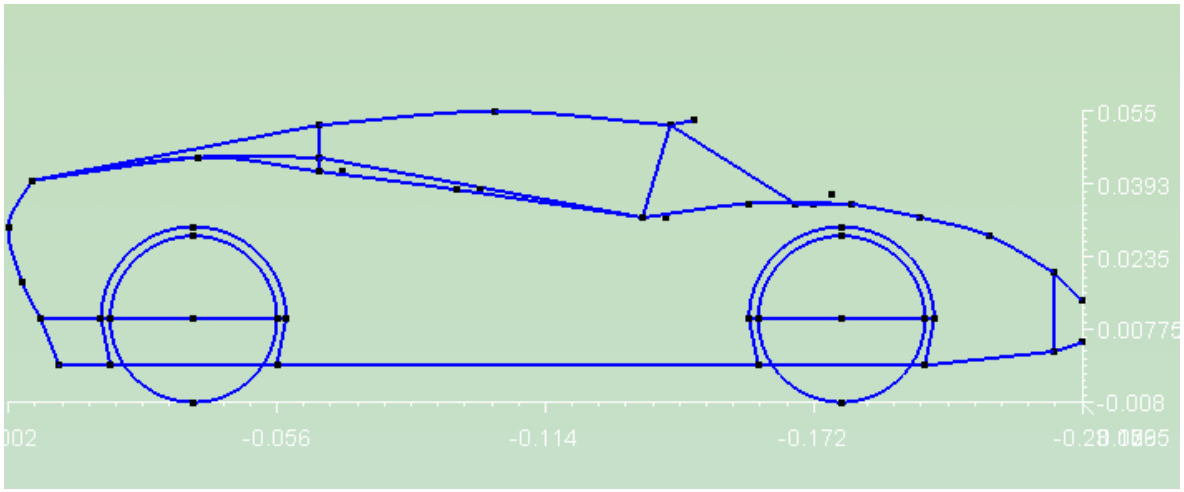


Figura 4.28. Delimitação do perfil da capota do carro.

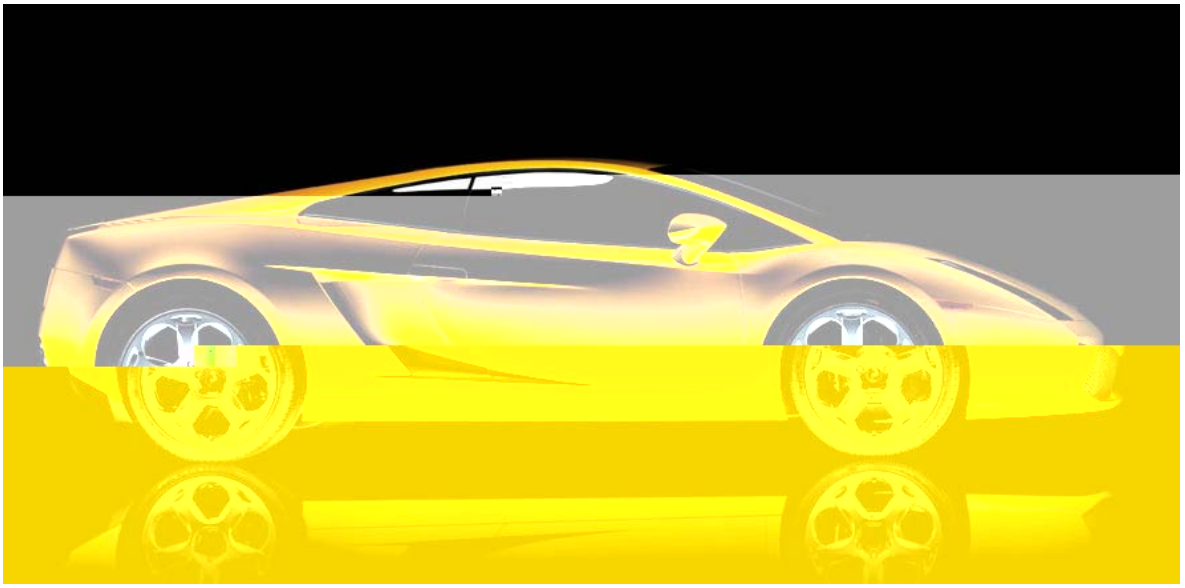


Figura 4.29. Foto do perfil do modelo.

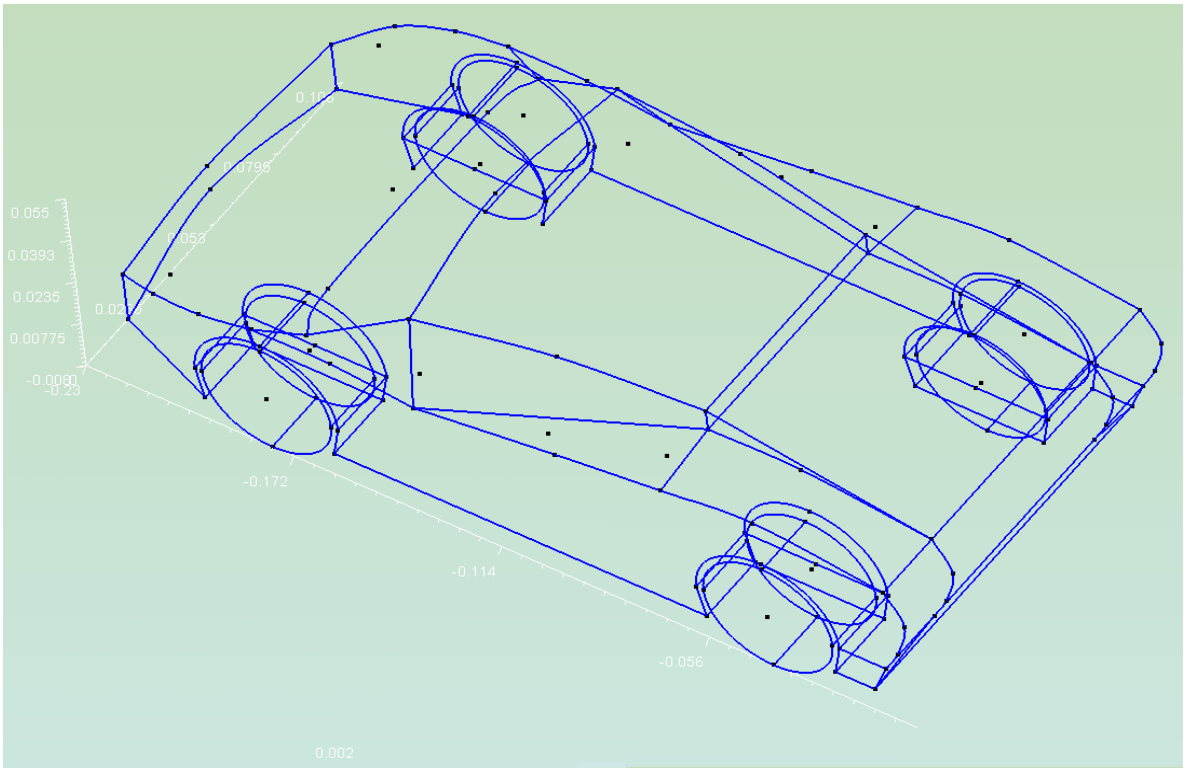


Figura. 4.30. Vista da capota do protótipo após a união de cada perfil da mesma.

Figura 4.31. Foto com detalhes da capota do modelo.

Após ser realizado todo o trabalho de delimitação da geometria, pode-se então, gerar o arquivo de malha, figura 4.32, para posterior geração do arquivo no formato STL mostrada na figura 4.33, que representa a superfície (casca) da geometria e é o formato desejado para a simulação computacional. Na figura de formato STL, pode-se visualizar linhas em azul, que são consideradas como cantos vivos. Estes cantos vivos não influenciam nos resultados de pós-processamento a não ser que a geometria esteja deformada, pois na analise da geometria, a mesma é considerada como um todo, mas caso necessário pode-se refinar a malha ou otimizar a geometria de forma a eliminar os cantos vivos.

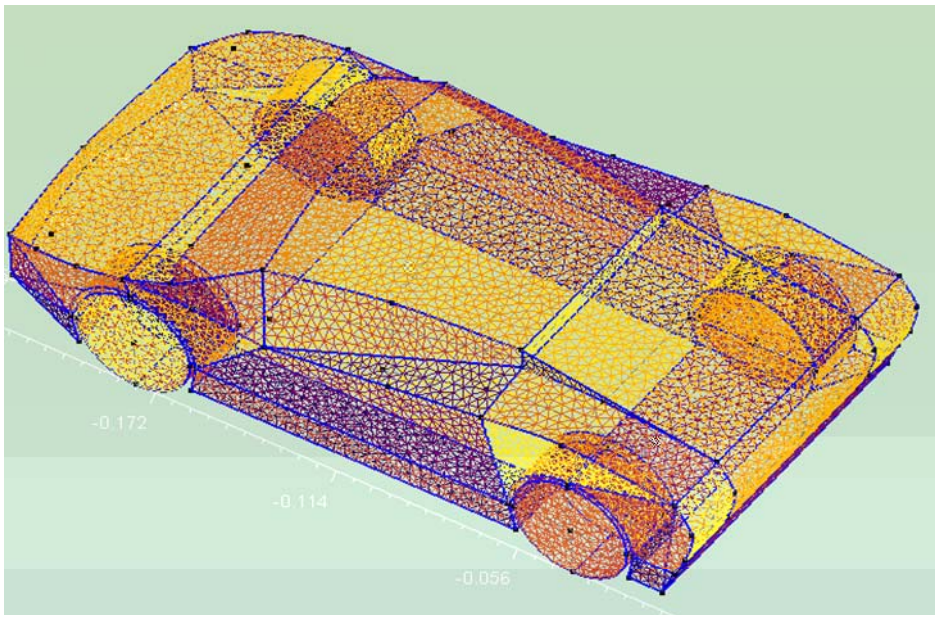


Figura 4.32. Malha do protótipo de automóvel.

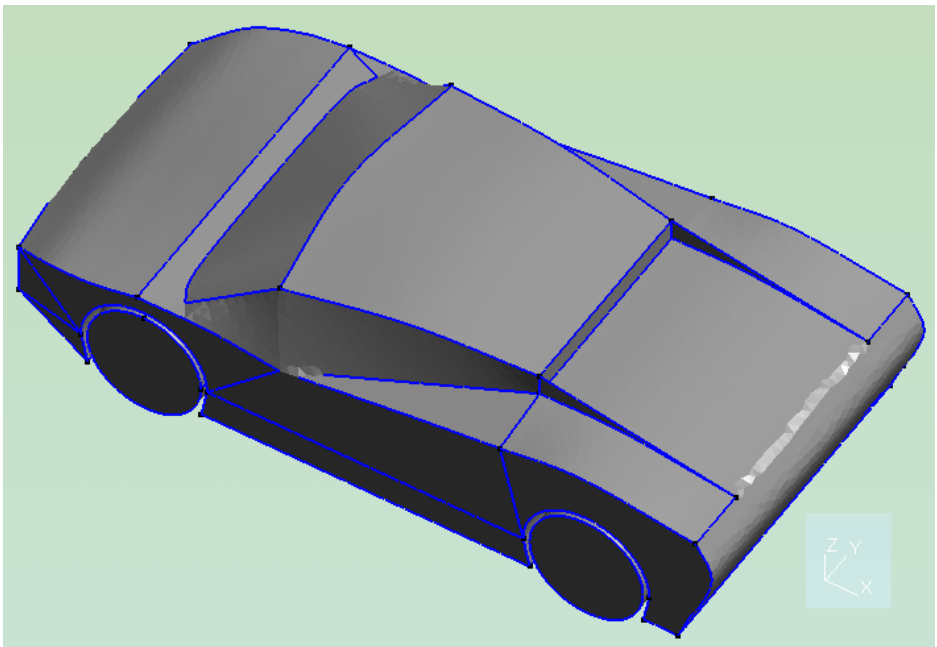
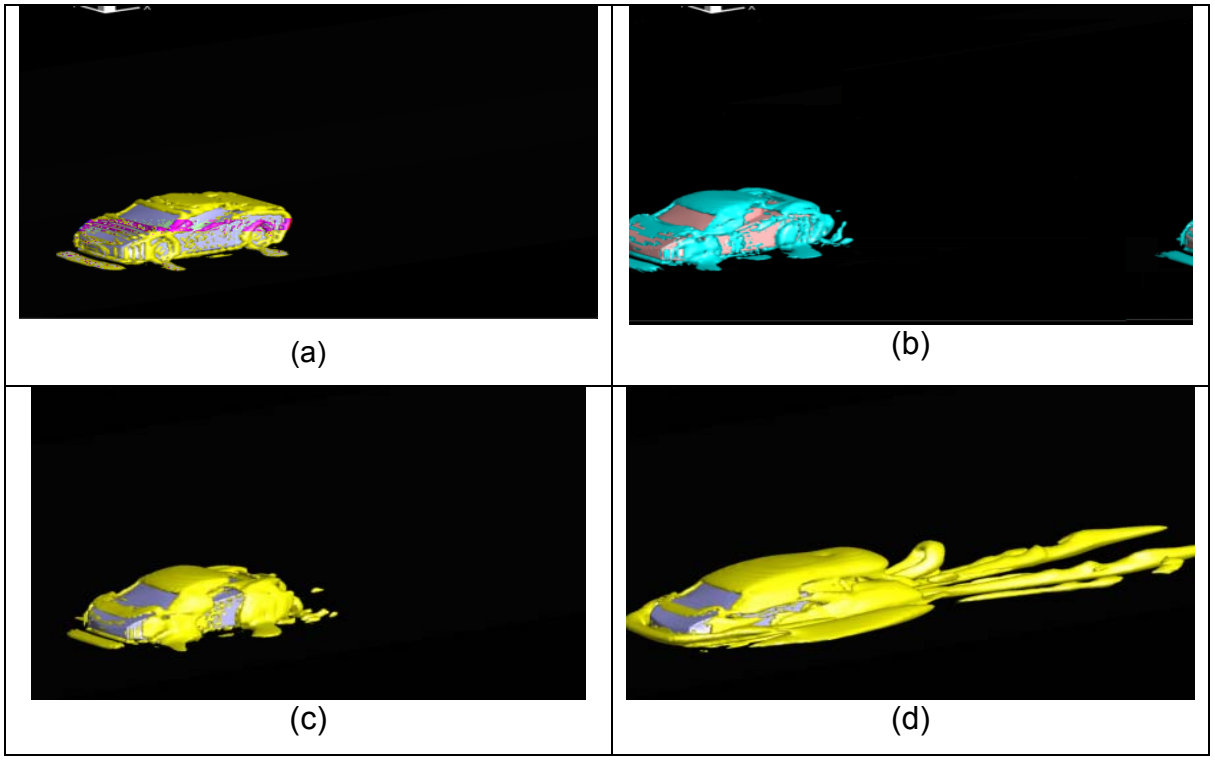
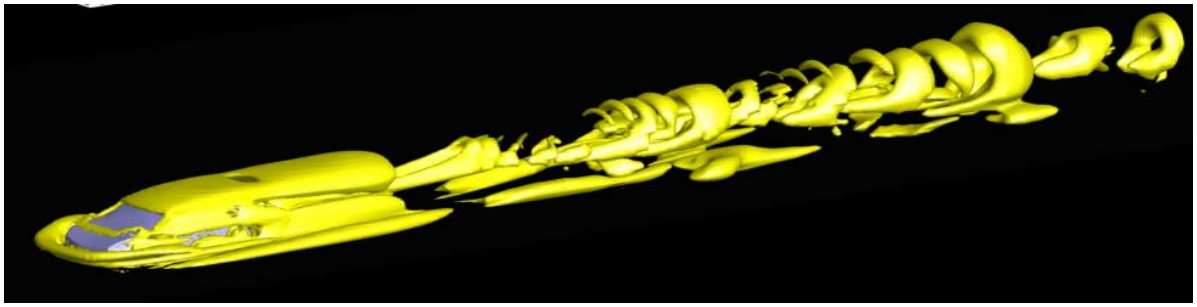


Figura 4.33. Superfície (casca) do protótipo do automóvel, formato STL.

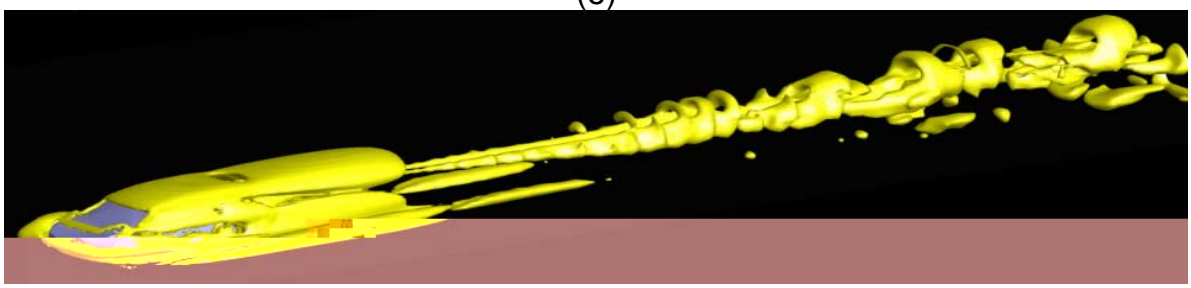
Utilizando a malha importada para o solver do LTCM, por Vedovoto (2007), pode-se visualizar na figura 4.34 (a)-(f), que mostra a evolução de isosuperfícies de Q = 5 para o escoamento sobre o protótipo de automóvel. É possível notar estruturas turbilhonares coerentes como os grampos de cabelo a jusante. A montante, estruturas do tipo ferradura de cavalo, são observadas. É importante dizer que essas estruturas apareceram, uma vez que a condição de contorno para Z = 0 foi utilizada com velocidade nula. Se a parede estivesse com uma velocidade U_∞, estas estruturas turbilhonares não seriam notadas.

A figura 4.35 mostra, em três vistas, detalhes das isosuperfícies de Q = 10 para t = 0,6s. Nota-se claramente os vórtices laterais, a estrutura tipo ferradura de cavalo a montante, e uma grande estrutura tipo grampo de cabelo surgindo a jusante.





(e)



(f)

Figura 4.34. Evolução temporal de isosuperfícies de Q = 5 para os tempos físicos: (a) t = 0,05s, (b) t = 0,045s, (c) t= 0,09s, (d) t = 0,8s, (e) t = 2,7s e (f) t = 4,7s (Vedovoto, 2007).

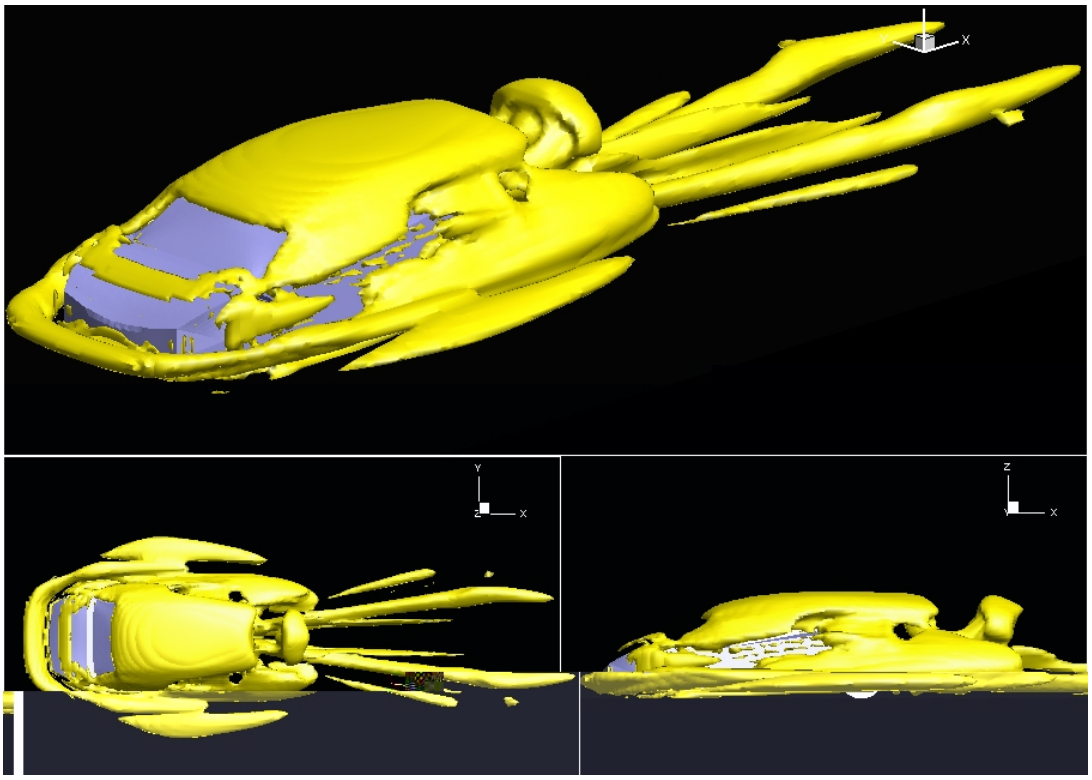


figura 4.35. Detalhes das isosuperfícies de Q = 10 para t = 0,6s (Vedovoto, 2007).

Realizando uma análise comparativa entre o escoamento obtido com a malha do lamborghini (Vedovoto, 2007), em relação ao escoamento sobre um cubo, pode-se notar a semelhança das estruturas turbilhonares, mesmo com condições de simulação diferentes.

Hwang e Yang, 2004, estudaram o escoamento sobre um cubo solidário a uma parede em Z = 0. A figura 4.36 mostra isosuperfícies de Q obtidos por Hwang e Yang (2004).

Tanto na figura 4.35 quanto na figura 4.36, é possível visualizar claramente, os vórtices em formato de ferradura, vórtices laterais e vórtices em formato de grampo de cabelo. Com esse tipo de comparação é possível confirmar a viabilidade das malhas geradas pelo GMSH, mesmo sendo as malhas geradas por composição de faces, resultando em irregularidades na união das mesmas.

É válido ressaltar que, a boa qualidade das malhas complexas, depende da habilidade ao desenhar as geometrias. Sendo assim, para que os resultados quantitativos possam ser validados, eles dependerão da virtualização do protótipo, e da riqueza de detalhes do mesmo.

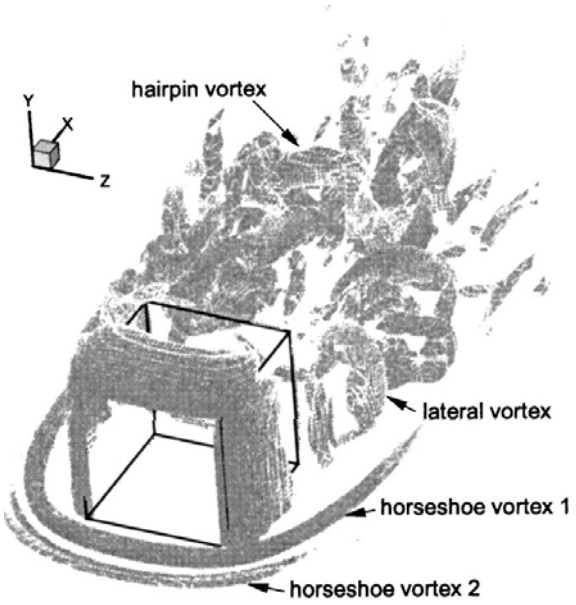


Figura 4.36. Isosuperfícies de Q obtidos por Hwang e Yang (2004), Re = 1.000.

4.3.2. Protótipo de uma aeronave

A geometria do protótipo em questão foi baseada em uma aeronave comercial (Legacy 600), de fabricação da EMBRAER. Apesar de ser composta por geometrias circulares, a complexidade na virtualização de seu protótipo é bem mais elevada que a do lambroghini Gallardo, pois o lamborghini teve sua estrutura criada em paralelo ao eixo x. Já o Legacy, como mostrado na figura 4.37, possui um ângulo de inclinação em relação ao eixo x.

Caso o avião fosse projetado em paralelo ao eixo x, todas as circunferências que compõe sua estrutura sofreriam modificações apenas na coordenada ao longo do eixo x.

Porém, com a inclinação acima citada, cada ponto das circunferências têm que ser recalculados à medida que se variam as posições em x, como mostrado na figura 4.38 abaixo.

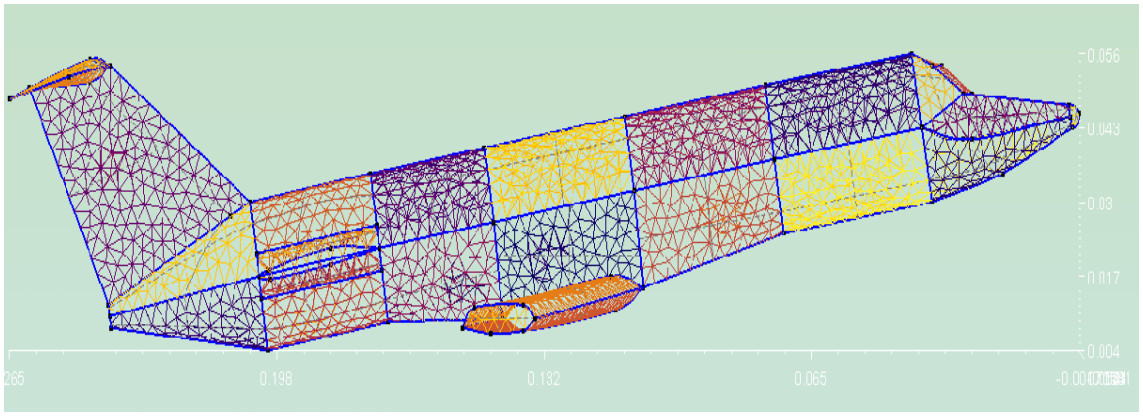


Figura 4.37. Malha do Protótipo do Legacy 600 com 10º de inclinação em relação ao eixo x.

Deve-se lembrar que a metodologia utilizada na virtualização deste protótipo, seguiu o mesmo padrão utilizado na criação do lamborghini, partindo da criação do perfil da aeronave. Como o modelo possui estrutura tubular, o perfil do mesmo é referencial da divisão da aeronave ao meio exato, como mostrado na figura 4.38, através do qual se cria a simetria entre as partes para produção da geometria completa.

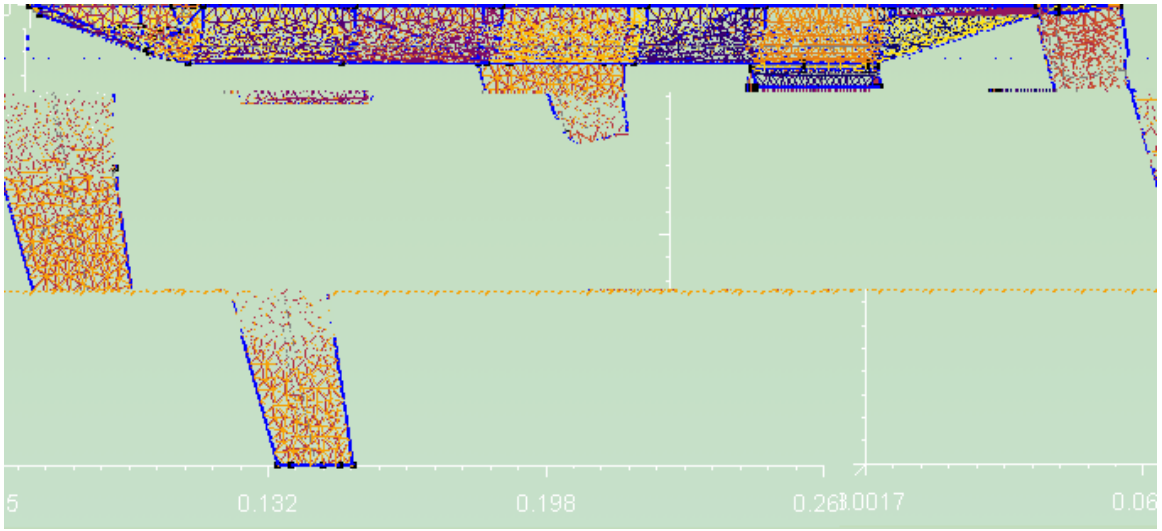


Figura 4.38. Malha do lado simétrico do protótipo.

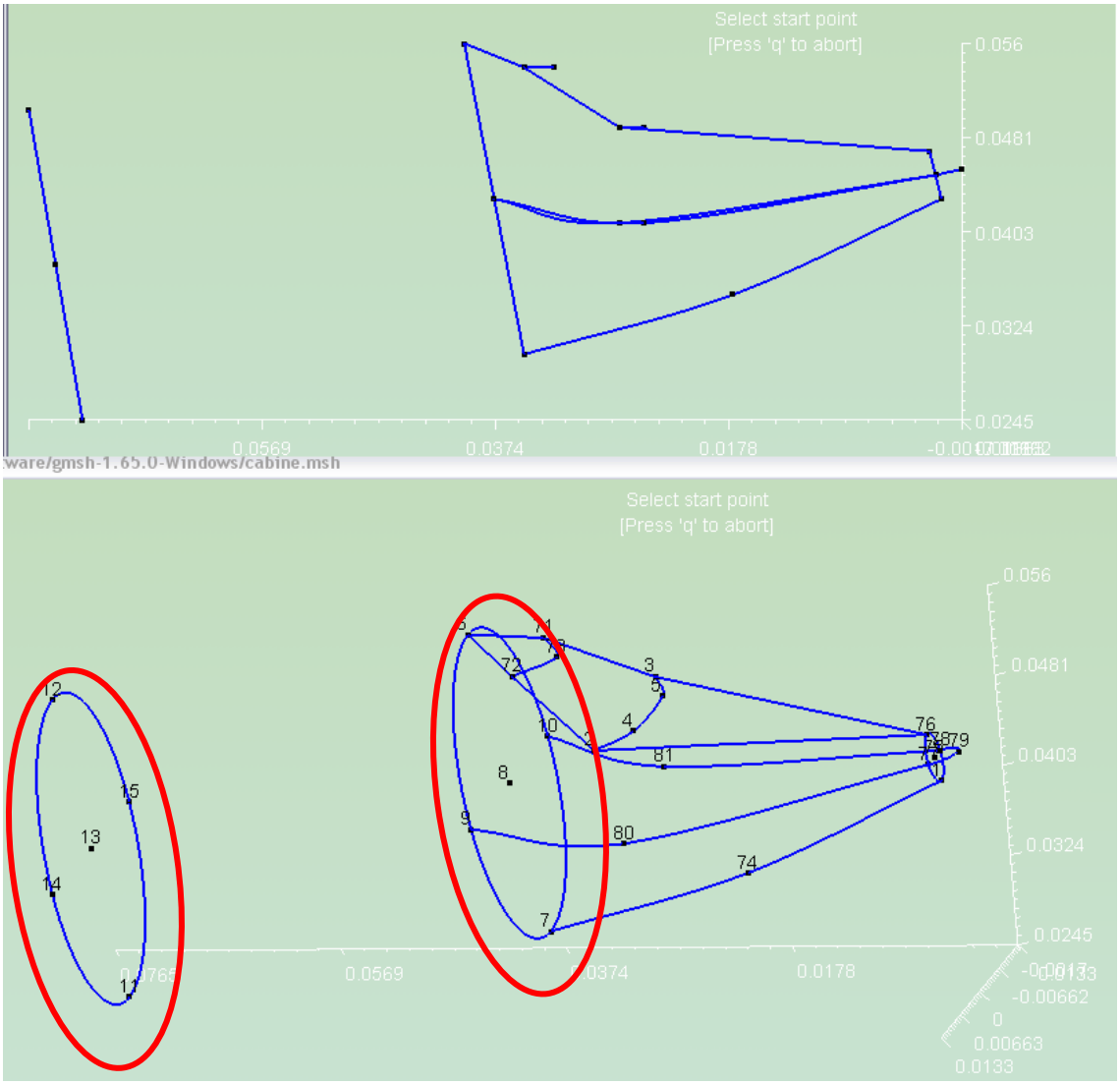


Figura 4.39. Detalhe dos pontos da geometria da cabine do avião.

Com essa inclinação (figura 4.39), pode-se notar a mudança em todos os pontos de cada circunferência, dando destaque para as que estão circuladas em vermelho, sendo os pontos mostrados na figura 4.40 abaixo, e essas mudanças persistem até o fim da virtualização do projeto.

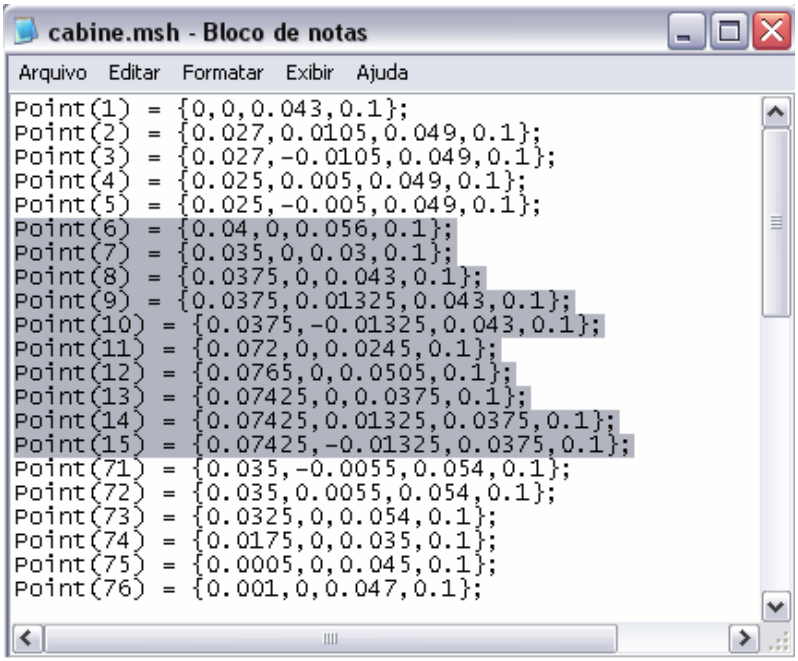


Figura 4.40. Variação das coordenadas dos pontos da geometria.

A malha de elementos triangulares que representam o protótipo pode ser visualizada com mais detalhes na figura 4.41 abaixo.

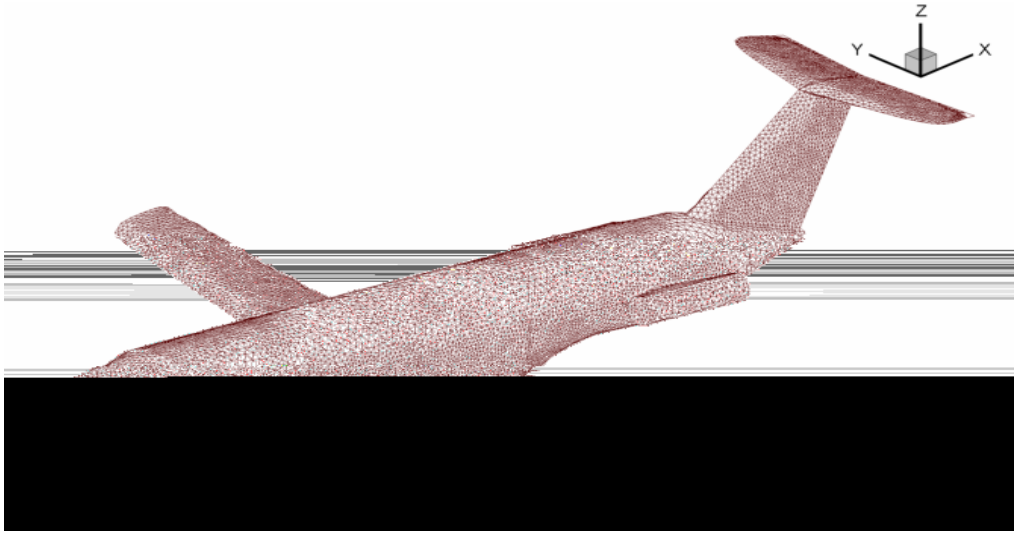


Figura 4.41. Malha de elementos triangulares do protótipo de aeronave.

Este protótipo virtual pode ser melhor detalhado com um processo de virtualização mais rigoroso, de forma que seja rico em detalhes. Um exemplo de detalhe a ser acrescentado para dar riqueza ao projeto, é mostrado na figura 4.42 abaixo, com a virtualização do trem de pouso do avião.

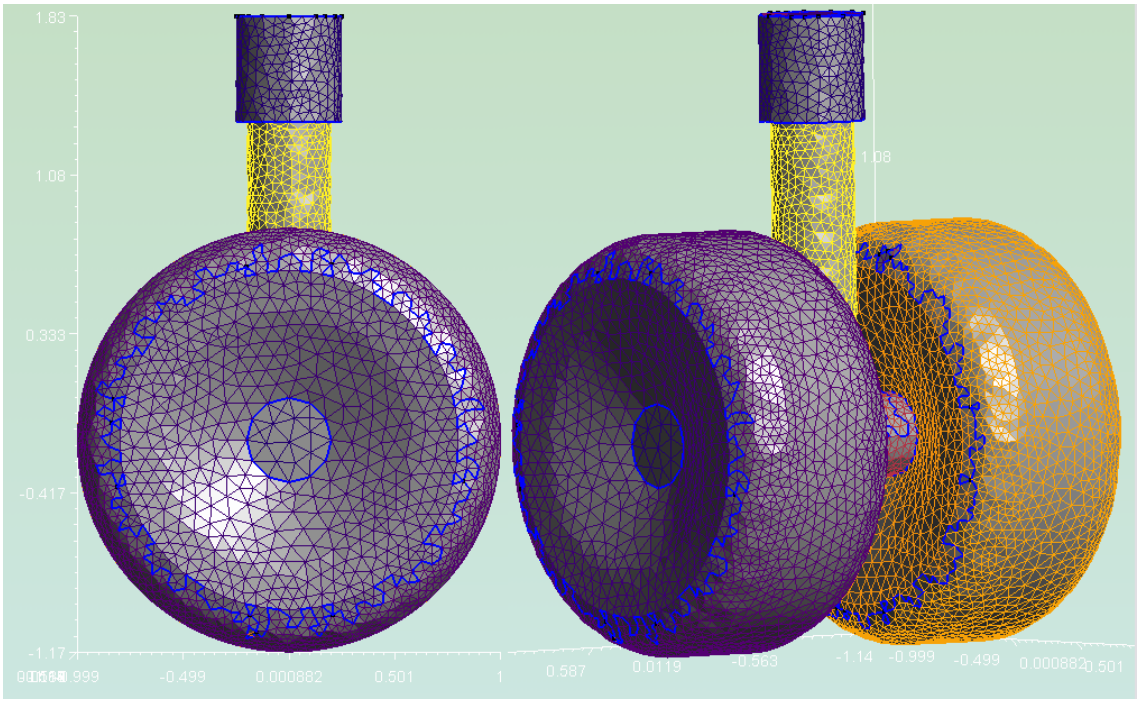


Figura 4.42. Detalhes de um trem de pouso de avião.

De acordo com a visualização gerada por Vedovoto (2007), a figura 4.43 abaixo, mostra algumas linhas de corrente posicionadas sobre o protótipo de aeronave, gerada neste projeto. Uma vez que um dos objetivos deste trabalho é verificar a capaciade de importação (upload) de geometrias complexas, não houve ainda uma preocupação de se simular o protótipo da aeronave a altos números de Reynolds, ou realizar a simulação em domínios com razões de aspecto muito grandes.

Nota-se que apesar do domínio escolhido não ser totalmente adequado (o tamanho característico da malha euleriana na região da fronteira imersa é duas vezes maior que o tamanho característico da malha lagrangiana), o código convergiu e mostrou resultados fisicamente consistentes. Portanto, o que se pode observar mediante as malhas geradas pelo GMSH, que é possível obter resultados quantitativos de forma a realizar, por exemplo, otimização de estruturas, através da virtualização de geometrias e da simulação numérica do escoamento.

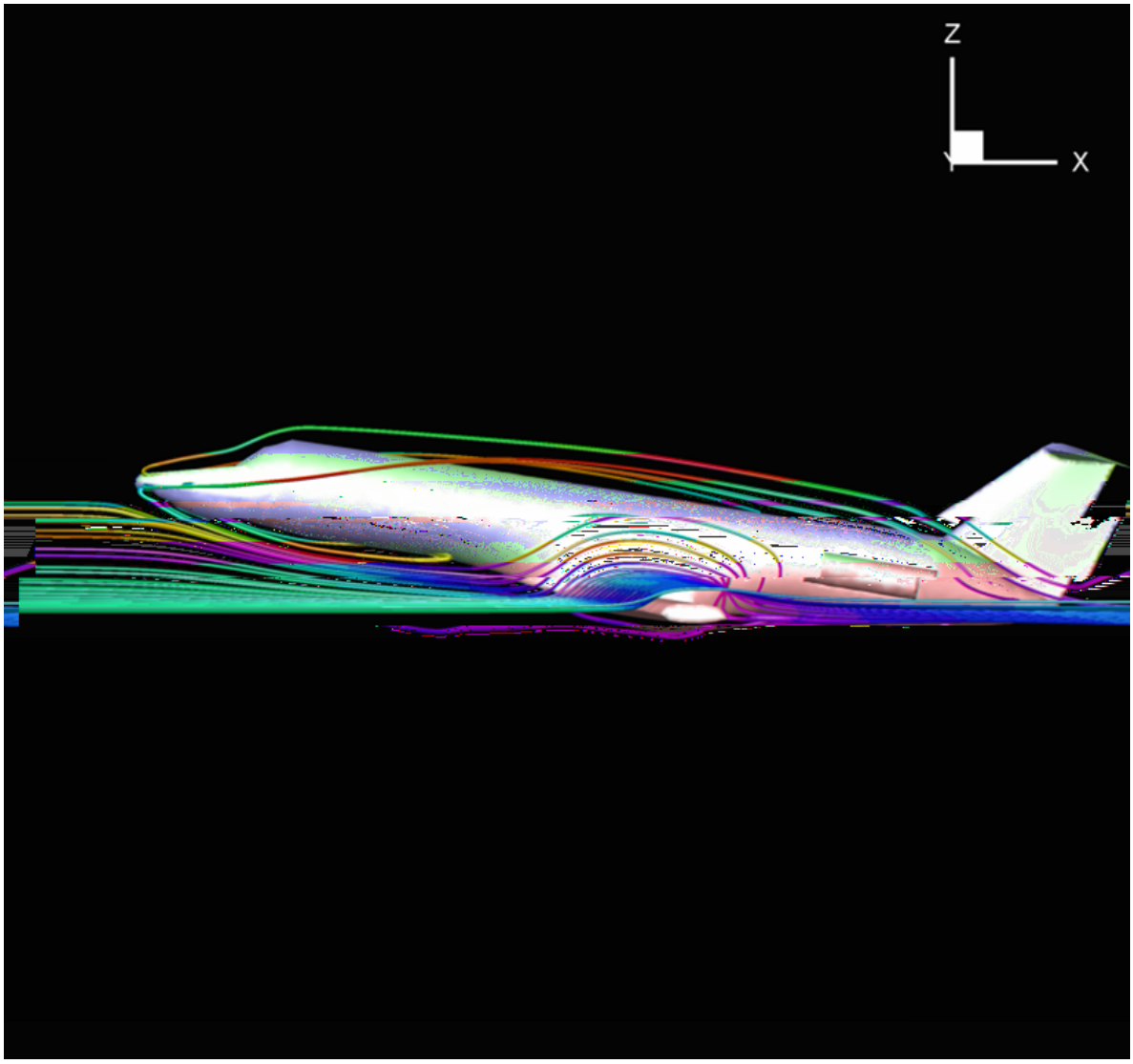


Figura 4.43. Linhas de corrente indicando o padrão de escoamento sobre o protótipo da aeronave (Vedovoto, 2007).

CAPITULO 5

Conclusão e Trabalhos Futuros

A aplicação da proposta metodológica apresentada permitiu auxiliar no processo de desenvolvimento da interface para o solver do LTCM, em forma de *site*, em um procedimento de virtualização de protótipos e no processo de geração de malhas superficiais.

As fases da metodologia tornaram este trabalho possível, pois na fase inicial resumiu-se a entrada de dados, através da coleta de informações sobre o código existente no LTCM e informações obtidas com os usuários do código, orientando o processo de delimitação dos elementos visuais presentes na interface segundo os objetivos de um *site* eficiente.

Embora a complexidade do projeto esteja diretamente ligada ao tipo de *site* a ser criado (portal, pessoal, institucional, comercial, educacional, entre outros), este fato não foi o fator delimitador deste projeto.

No presente trabalho propôs-se elaborar uma interface gráfica que pudesse ser aplicada ao solver de CFD desenvolvido no LTCM, com a finalidade de atrair usuários ao uso do solver, oferecendo facilidade de interação homem-software. É importante salientar que uma interface em forma de *site,* produzido com a finalidade de viabilizar a comunicação entre usuários e o solver, não é mais importante que uma interface gráfica de instalação local.

Com a implementação da interface, percebe-se que mesmo sendo a iteração homes/software via Web ou de instalação local, ambas têm objetivos comuns. Cada uma possui suas qualidades e facilidades no manuseio, porém o processo metodológico para criação de um site, amplia o acesso às informações, pois as informações são manipuladas e viabilizadas de qualquer local com acesso a internet.

Percebeu-se que trabalhar com conhecimentos específicos, tornando-os requisitos técnicos do processo projetual foi uma tarefa enriquecedora. No que diz respeito a vincular as informações específicas de mecânica dos fluidos de forma prática e objetiva, oferecendo os conteúdos mais básicos e necessários para a interface, constatou-se que, pelo fato do código computacional já estar parcialmente validado, a maior preocupação foi evitar a redundância de informações a serem enviadas para o solver do LTCM. Verificando-se esta preocupação, viabilizou-se uma solução, ainda que simples, sobre o conceito de interface gráfica, porém funcional e versátil.

O presente trabalho alcançou os objetivos propostos, limitando-se ao enfoque do tema sob a ótica da comunicação entre homem/software, o que reforça a não presença de orientação e suporte de conteúdos sobre alteração ou desenvolvimento do código computacional existente. Com isso, pode-se visualizar que a metodologia de implementação para a interface do Fluids-3D possui caráter generalista, deixando possibilidades de melhoras na interface, tais como módulos adicionais, à medida que os mesmos sejam acrescentados ao solver.

Prevê-se que futuramente a WEB esteja viável a um grande grupo de pessoas através da televisão, o que já se constata, ainda que de forma incipiente, através de TVs via cabo ou satélite. Por esse motivo, supõe-se que a comunicação continuará sendo o objetivo dos *site*s, porém as inovações tecnológicas irão delimitar novas possibilidades que poderão afetar os conhecimentos básicos necessários ao profissional. Isso nos leva a crer que, o conhecimento, seja ele geral ou especifico como o de mecânica dos fluidos, poderá ser disponibilizado de maneira rápida e funcional.

Além dos resultados alcançados com a produção da interface gráfica acima citada, no presente trabalho, dedicou-se a busca por tecnologias gratuitas na área de geração de malhas de geometrias complexas.

O uso das ferramentas para criar a interface e viabilizar a comunicação da interface com o solver, permitiu selecionar uma ferramenta gratuita de geração de malha, chamada GMSH, a qual possui uma funcionalidade satisfatória para as necessidades do projeto.

É importante salientar que, o fato de valer-se de ferramentas gratuitas na composição de trabalhos científicos, exige uma necessidade maior de se buscar resultados comparativos, para validar as informações computadas.

Mesmo obtendo bons resultados com a ferramenta geradora de malha escolhida, foi notório que softwares livres possuem suas limitações, como dificuldades em se encontrar documentações, existência de erros (bugs), etc; problemas esses que não foram impecinhos no desenvolvimento do projeto.

Como trabalhos futuros, deixa-se as seguintes recomendações que podem melhorar

ainda mais a interface desenvolvida:

Implementar o gerenciamento de usuários da interface, através de filtros que possam restringir o número de usuários e o número de máquinas que os mesmos possam acessar no cluster.

Acrescentar à interface, módulos de pós-processamento, tal qual usado pelo Ganglia, por exemplo, que obtém informações quantitativas e as transformam em gráficos, através do uso da linguagem PHP com a ferramenta RRDtool.

Disponibilizar uma ferramenta geradora de malha através da Internet, vinculada à interface, de forma a diminuir uma etapa na simulação de casos de mecânica dos fluidos; além de possibilitar ao solver a capacidade de importação de diferentes listas de geometrias;

Incluir um módulo de visualização de arquivos no formato STL, o qual pode ser feito em linguagem Java e inseridos no código PHP da interface, dando maior caráter gráfico a interface em questão.

CAPITULO 6

Referências Bibliográficas

ARAÚJO, A. C. Framework de Análise e Projeto Baseado no RUP para o desenvolviemento de aplicações web. Dissertação de mestrado publicado na UFPE, Recife, 2001

BARRETT, E. MIT guide to teaching web *site* design. Cambridge: Mit Press 2001

BARRETT, E. The Society of text: hypertext, hypermedia, and the social construction of information. Series: Information systems. Cambridge: Mit Press, 1989.

BLACK, R. *Web Sites that work.* San Jose:Adobe Press, 1997.

BITTNAR, Z.; RYPL, D. ``*Direct Triangulation of 3D Surfaces Using the Advancing Front Technique''*, In: Numerical Methods in Engineering '96 (ECCOMAS '96), Wiley & Sons, 86-99, 1996.

BONSIEPE, G. *Design – do material ao digital.* Florianópolis, FIES/IEL, 1997.

BÜRDEK, B. E. *Diseño – História , teoria y prática del diseño industrial.* Barcelona: Gustavo Gili, 2ª ed., 1999.

CAMPREGHER, R. “Modelagem Matemática Tridimensional Para Problemas de Iteração Fluido-Estrutura”. Tese de Doutorado. Universidade Federal de Uberlândia, 2005.

DALHEIMER, M., “Programação em Qt”, Editora Moderna, 1999;

DYN, N.; GREGORY, J.A.; LEVIN, A. ``*A Four-Point Interpolatory Subdivision Scheme for Curve Design''*, Computer Aided Geometric Design, Vol. 4, 257-268, 1987.

GATTASS, M.; LEVY., C. H. New Techniques to Aid Interactive Graphical Finite Element Analysis, Anais do Workshop Internacional de Aplicação de Mecânica Computacional em Geotecnia, Rio de Janeiro, 29-31 de junho de 1991.

GILMANOV, A.; SOTIROPOULOS, F; BALARAS, E. “A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartesian grids”, Journal of Computational Physics 191, 660-669, 2003.

GLENN, B. T.; CHIGNELL, M.H.; Hypermedia: Design for Browsing. In: Hartson, HR Hix, D. Advances in Human-Computer Interaction. Nerwood, New Jersy: Ablex Publishing Corporation, v.3, 1992

HALLER, G., 2005, “An objective definition of a vortex”, Journal of Fluid Mechanics 525, 1-26.

HALSTEAD, M.; KASS, M.; DEROSE, T. ``*Efficient, Fair Interpolation using Catmull-Clark Surfaces''*, In: Computer Graphics Proceedings (SIGGRAPH '93), 35-44, 1993.

HARTSON, H. R.; HIX, D. Human-Computer Interface Development: Concepts and Systems for its Management, ACM Computing Surveys (21), No. 1, pp. 9-93, March 1990.

HWANG, J., YANG, K. “Numerical study of vertical structures around a wall-mounted cubic obstacle in channel flow”, Physics of Fluids, Vol. 16, No. 7, 2382-2394, 2004.

JEONG, J. E HUSSAIN, F. “On he identification of a vortex”, Journal of Fluid Mechanics 285,69-94, 1995.

LIMA E SILVA, A. L. F., 2002, “Desenvolvimento e implementação de uma nova metodologia para modelagem de escoamentos sobre geometrias complexas: método da fronteira imersa como Modelo Físico Virtual”, Tese de Doutorado, Universidade Federal de Uberlândia, Uberlândia, MG, Brasil.

MARCUS, A., VAN DAM, A. ; User-interface developments for the nineties, 1991.

MARTINS, V. ; O processo Unificado de Desenvolvimento de Software, Companhia de Informática do Paraná – CELEPAR: Paraná, 1999. Disponível em: http://www.pr.gov.br/celepar/batebyte/edicoes/1999/bb89/software.htm. Acessado em : 14/06/2003

MOK, C. *Designing business – multiple media, multiple disciplines.* São Jose: Adobe Press, 1996.

MUTO, C. A., PHP & MySQL: Guia Introdutório. Rio de Janeiro: Editora Brasport, 2006.

NIEDERAUER, J. PHP 5, Editora Novatec, Editora Brasport. 2005.

OLIVEIRA, G. P.; VEDOVOTO, J.M.; FERNANDES, A.P.; NETO, A.S. Desenvolvimento de uma interface gráfica para dinâmica dos fluidos computacional, utilizando a metodologia de fronteira imersa, In proceedings of 11th Brazilian Congress of Thermal Sciences and Engineering – ENCIT, 2006.

PERUZZI, J. T. *Manual sobre a importância do design no desenvolvimento de produtos.* Bento Gonçalves: SENAI/CETEMO/SEBRAE, 1998.

PETERS, R., The Simplest Subdivision Scheme for Smoothing Polyhedra - ACM Transactions on Graphics 16(4):420-431, 1997.

PITANGA, M, 2002, “Construindo supercomputadores com Linux”, Editora Brasport, 2004.

RADFAHRER, L.. *Design/Web/Design.* São Paulo: MarketPress, 1999.

RAY, J. Aprendendo em 21 dias Dreamweaver UltraDeb 4, Editora Campus, 2001.

RYPL, D.; BITTNAR Z. ``*Triangulation of 3D Surfaces: From Parametric to Discrete Surfaces''*, In: CD-ROM Proceedings of the Sixth International Conference on Engineering Computational Technology, Civil-Comp Press, 2002.

RYPL, D.; BITTNAR Z. ``*Triangulation of 3D Surfaces Reconstructed by Interpolating Subdivision''*, to appear in Computers and Structures.

SHAMES, I. H. Mecânica dos Fluidos. Volume 1. Princípios Básicos. São Paulo: Editora Edgard Blucher Ltda., 1973.

STERNE, J. Marketing na Web: Integrando a Web à Sua Estratégia de Marketing. São Paulo: Campus, 2000.

STOLLNITZ, E.J.; DEROSE, T.D.; SALESIN, D.H. ``*Wavelets for Computer Graphics: Theory and Applications''*, Morgan Kaufmann Publishers, Inc., 1996.

TOUSSAINT, G. T.; Computational Geometry on the web. Disponível em: <http://cgm.cs.mcgill.ca/~godfried/teaching/cg-web.html>

UNVERDI, O.; TRYGGVASON, S. Computations of multi-fluid flows. Physica D: Nonlinear Phenomena, 1992.

VEDOVOTO, J. M. Modelagem Matemática e Simulação Numerica de Escoamentos sobre Geometrias Complexas Tridimensionais Utilizando o Método da Fronteira Imersa e Processamento Paralelo, Dissertação em redação, Universidade Federal de Uberlândia, Uberlândia, 2007.

VEDOVOTO, J.M.; CAMPREGHER, R.; SILVEIRA NETO, A. “Mathematical modeling and numerical simulation of a three-dimensional flow over complex geometries using the immersed boundary method”, In proceedings of 11th Brazilian Congress of Thermal Sciences and Engineering – ENCIT, 2006.

ZORIN, D.; SCHRÖDER, P.; SWELDENS, W. ``*Interpolating Subdivision for Meshes with Arbitrary Topology''*, In: Computer Graphics Proceedings (SIGGRAPH '96), 189-192, 1996.

VIDA DIGITAL. Parte integral da Revista Veja. São Paulo: Abril, Dezembro 2001. Ano 33, nº 52.

http://pt.wikipedia.org/wiki/CFD acessado em: 20/11/2006.

GEUZAINE, C., et al. “Gmsh Reference Manual: The documentation for Gmsh 1.60”,2005. Disponível em:
<http://www.geuz.org/gmsh/> Acessado dia: 20/08/2005;

SCHÖBERL, J., “Netgen – 4.3”, 2003. Disponível em:
<http://www.hpfem.jku.at/netgen> Acessado dia: 23/08/2005;

<http://www.cs.cmu.edu/~quake/triangle.html> Acessado dia: 02/09/2005;

<http://www.borland.com.br/delphi> Acessado dia 05/10/2005;

<http://www.borland.com.br/kylix> Acessado dia 05/10/2005;

<http://www.trolltech.com/qt/designer.html> Acessado dia 08/10/2005;

<http://doc.trolltech.com/3.3/designer-manual.html> Acessado dia: 08/10/2005;

Anexo 1. Tutorial do GMSH

O GMSH é um software gerador de malhas não-estruturadas para métodos numéricos (MEF – Métodos de Elementos Finitos), incluindo uma ferramenta CAD para desenho de geometrias 2D e 3D. O Gmsh conta ainda com uma suíte de visualização de dados para pós-processamento de resultados, baseado em OPEN GL.

O GMSH possui uma organização constituída de 4 módulos: Geometry, Mesh, Solver e Post-processing como mostrado na figura 1 abaixo, além dessa organização em módulos, ele ainda apresenta uma tela utilizada para desenhos semelhante às utilizadas no AutoCAD, visto na figura 2.

Figura 1. Visualização dos módulos do Gmsh

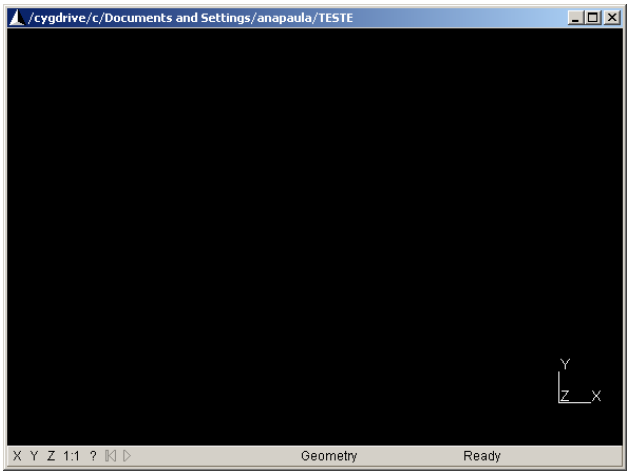


Figura 2. Tela de CAD

1.1. Como criar uma malha:

Para se criar qualquer tipo de malha, deve-se primerio criar a geometria respecitiva a superfície que se deseja malhar. A criação de uma geometria deve seguir os seguintes passos: Clica-se em Elementary, Add, New para criar “entidades” geométricas elementares (ponto, reta, arco de circunferência, splines, etc.), como na figura 3 abaixo.

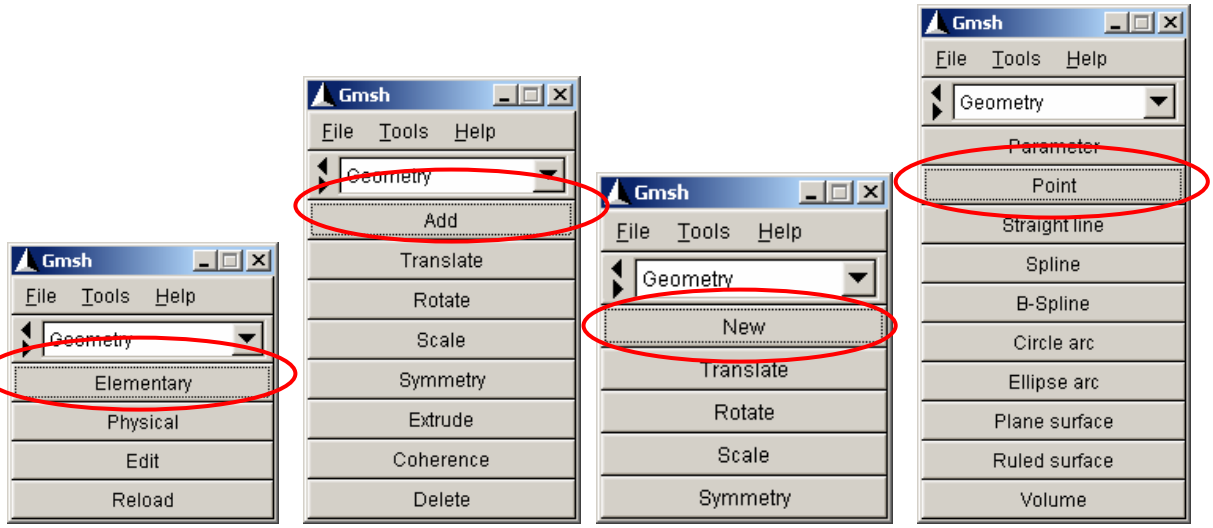


Fig. 3 – Seqüência de comandos para criar uma geometria

Ao se clicar em *Point*, visualizar-se-á uma janela onde serão inseridas as coordenadas dos pontos bem como seu comprimento. As coordenadas podem ser inseridas manualmente na tela de CAD, posicionando o mouse na coordenada desejada seguido da tecla “e”.

Abaixo segue um exemplo para o ponto com coordenadas (x,y,z) = (4,0,0) e comprimento 0.1, mostrado na figura 4.

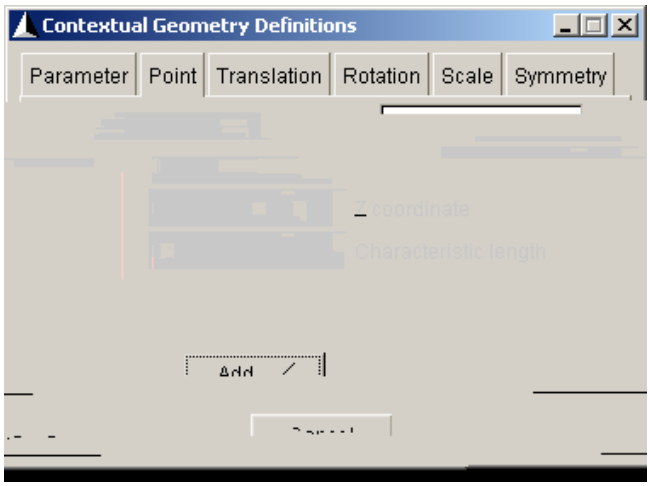


Fig. 4 - coordenadas dos pontos

Quando se cria uma seqüência de pontos, um arquivo de extensão ".geo", figura 5, é criado e pode ser visualizado no módulo "*Geometry*" com a opção "*Edit*". Sendo assim podemos criar geometrias, figura 6, sem utilizar interface gráfica, apenas editando um documento texto.

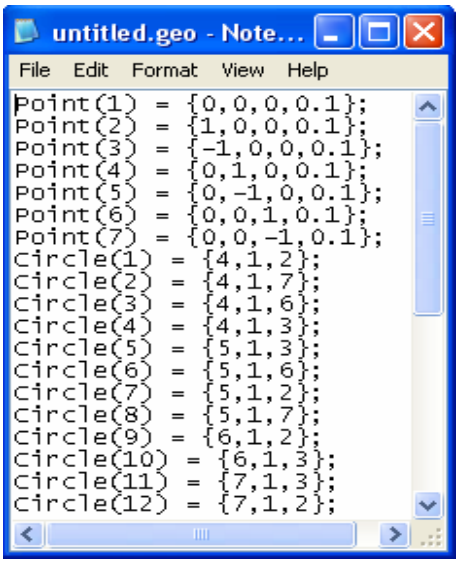


Fig. 5 – arquivo ".geo" da geometria

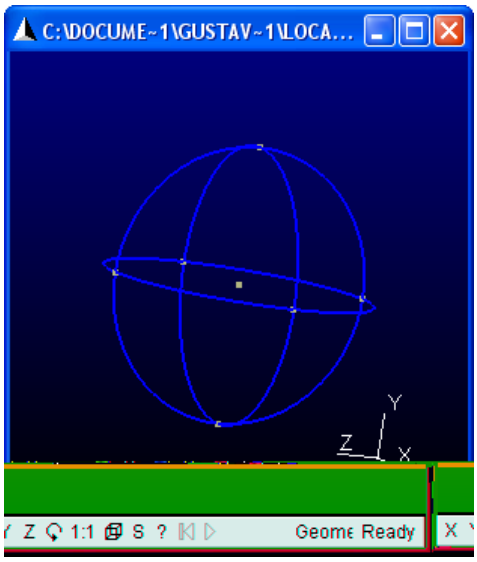


Fig. 6 – visualização da geometria

No caso de geometrias tridimensionais deve-se levar em consideração o eixo dos planos cartesianos, que disponibiliza uma referência da orientação da figura criada, como mostrado na figura 7.

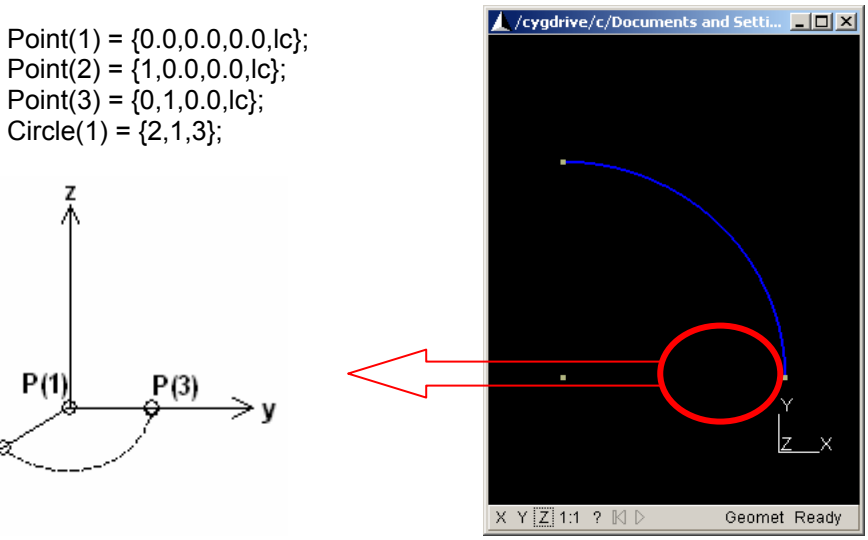
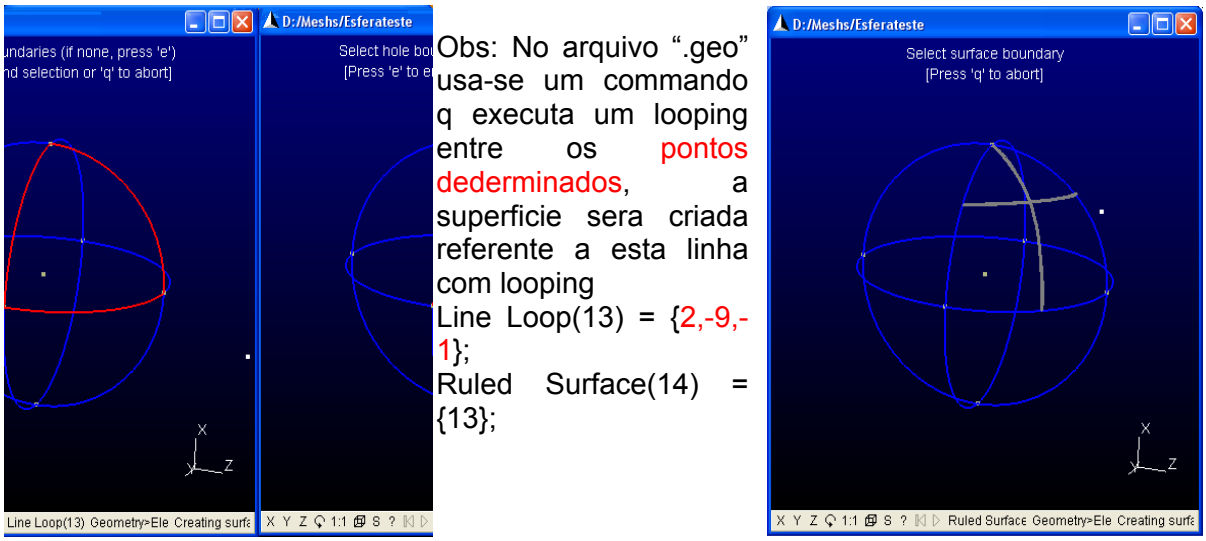


Fig. 7 – Referência com o eixo das coordenadas

Para indicar os limites de uma superfície seleciona-se a opção *Ruled surface* ou

Plane Surface, dependendo de o formato do plano ser curvo ou não. Ao clicar em alguma linha, o gmsh tenta construir a superfície. Para confirmar a criação da superfície, tecla-se “e”.



Obs: No arquivo “.geo” usa-se um commando q executa um looping entre os pontos dederminados, a superficie sera criada referente a esta linha com looping
Line Loop(13) = {2,-9,-1};
Ruled Surface(14) = {13};

Fig. 8 – Determinando uma superfície na geometria

Para criar a malha, deve-se selecionar o módulo Mesh, como não determinamos nenhum volume, vamos criar uma malha somente sobre a superfície:

Fig. 9 – Malha sobre uma superfície

Após determinar todas as superfícies que se deseja na geometria, pode-se gerar a malha completa da geometria.

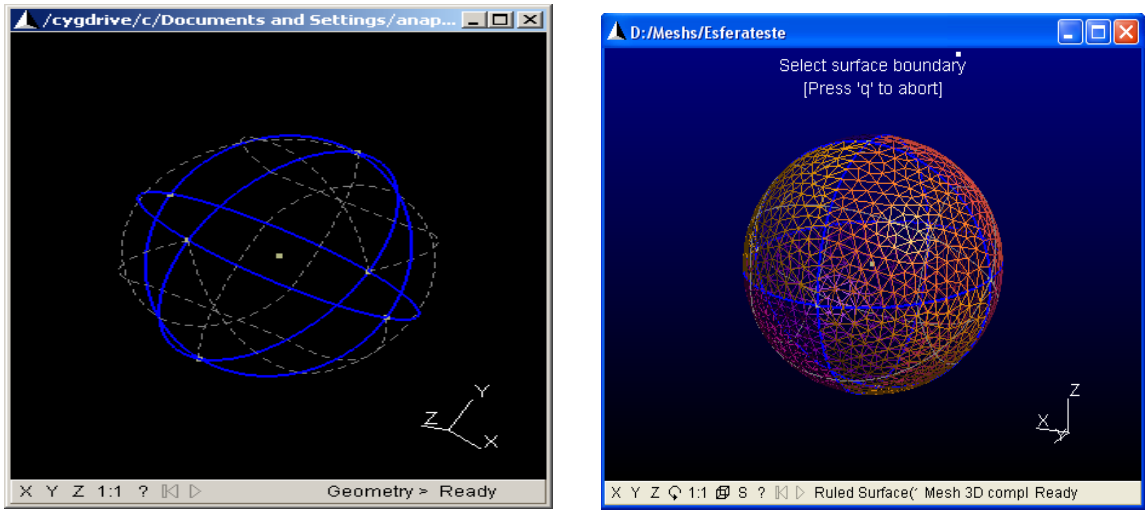


Fig. 10 – Malha completa após determinação das superfícies

1.2. Como refinar uma malha:

De acordo com a finalidade para qual serão usadas as geometrias e as malhas, existe a possibilidade de se aperfeiçoar a qualidade das mesmas. A malha triangular padrão do Gmsh, vista na figura 10, é gerada com elementos o mais uniforme possível, podendo haver algumas deformações.

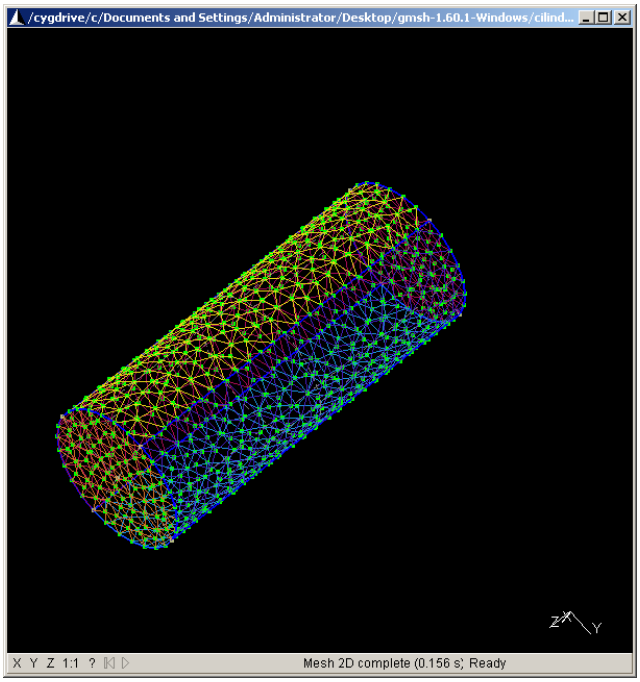


Fig. 11 – Malha triangular padrão do GMSH

A possibilidade de refinar a malha das geometrias é viável principalmente quando

essa geometria é formada por composição de várias faces, de forma a gerar geometrias complexas como automóveis e aviões. Tomando como exemplo a última geometria citada, podemos analisar que, pelo fato do avião possuir vários encontros de superfícies, e possibilidade de deformação da malha é muito grande, Figura 12.

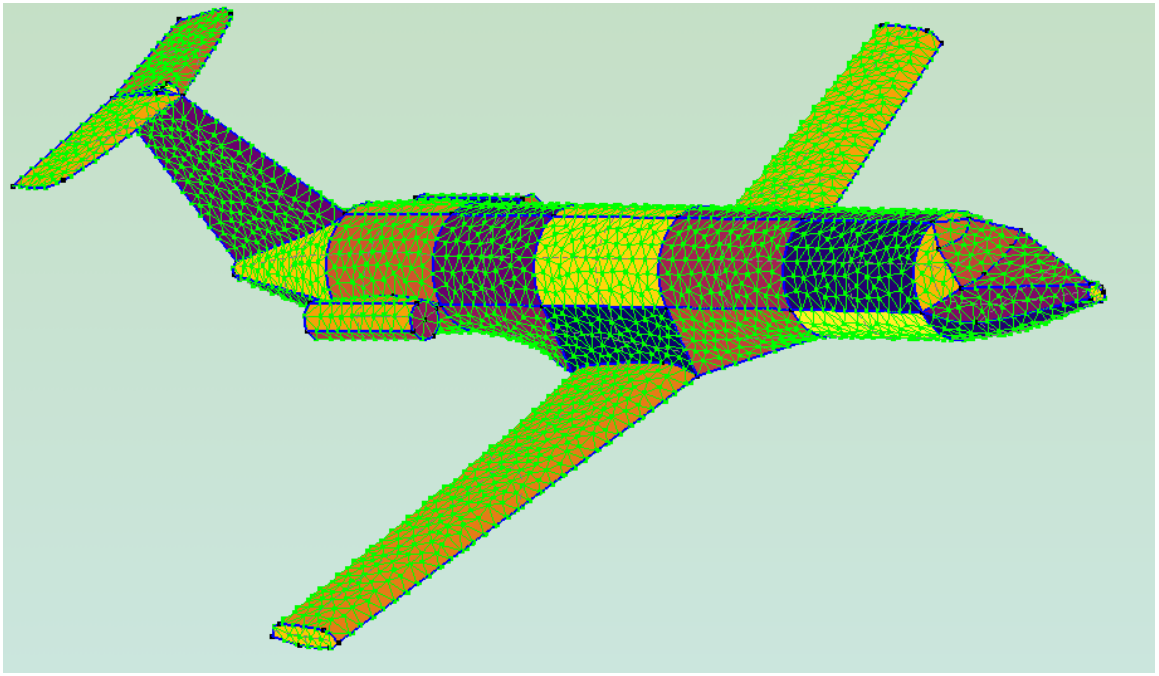


Figura 12. Detalhes de Geometria de avião composta por várias superfícies.

Para o arquivo de malhas, a necessidade de refinamento não é de grande importância, uma vez que o arquivo mais usado para simulação computacional em CFD é o de formato .STL. Sendo assim, a maior preocupação de refinamento de malha é na geração deste tipo de arquivo, onde se torna visível às deformações da malha, figura 13.

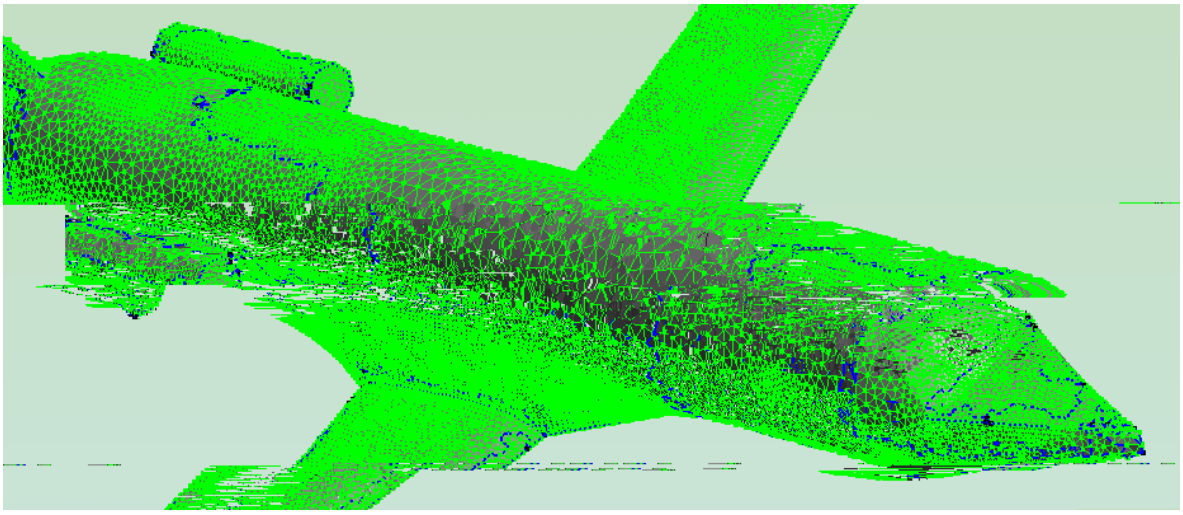


Figura 13. Detalhes do arquivo .STL com deformação da malha superficial.

Portanto, nestes casos faz-se necessário refinar a malha computacional para gerar o respectivo arquivo em STL. Este refinamento é feito no GMSH, através da configuração de alguns parâmetros.

Para alterar estes parâmetros, basta através da interface do GMSH, acionar na barra de ferramentas a opção "*Tools*" e depois "*Options*", assim a caixa de configuração para essas informações se abrirá, como mostrado na figura 14 abaixo.

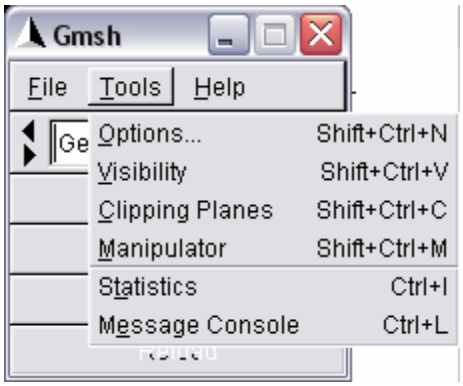


Figura 14. Chamando a área para configuração do refinamento.

Para refinamento, tem-se a opção de refinar a malha e o STL que compões a superfície. Na figura 15 abaixo, visualiza-se a área de configuração do refinamento da malha que pode ser alterada mudando o valor "*Characteristic length factor*".

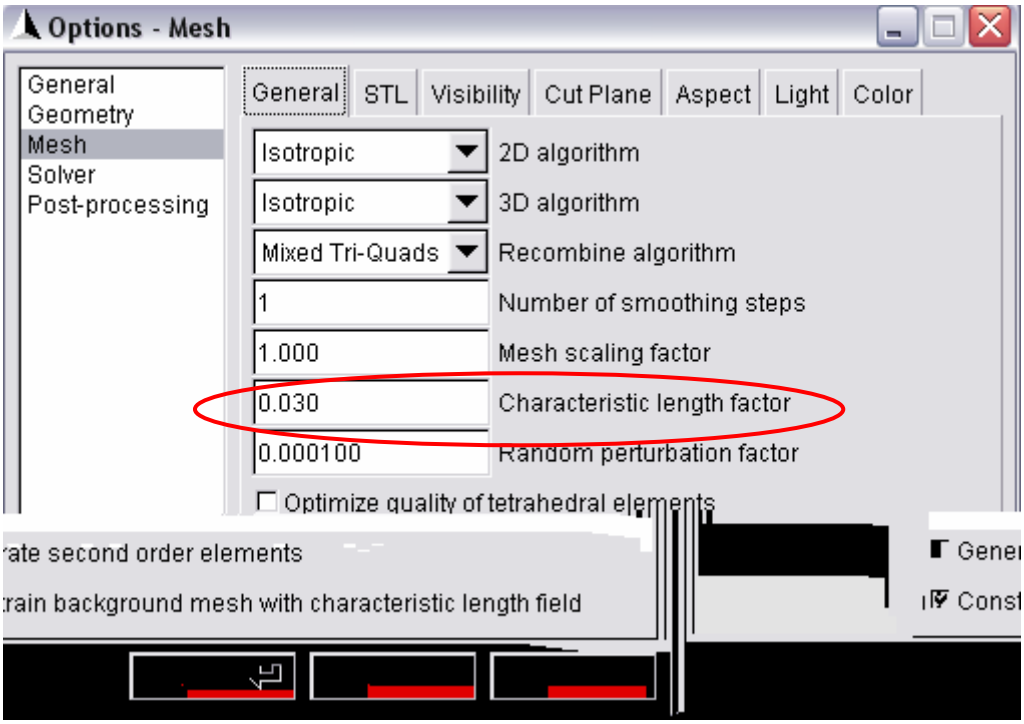


Figura 15. Área de configuração do tamanho da malha.

Porém para o refinamento do arquivo utilizado para simulações computacionais, o real interesse é modificar a estrutura da casca da geometria, através de alterações dos parâmetros abaixo selecionados. Nota-se que eles dependem do parâmetro modificado anteriormente na malha, pois se efetua uma divisão do fator de tamanho característico configurado para malha, pelos parâmetros de tamanho de elemento mínimo e tamanho de elemento objetivado, figura 16.

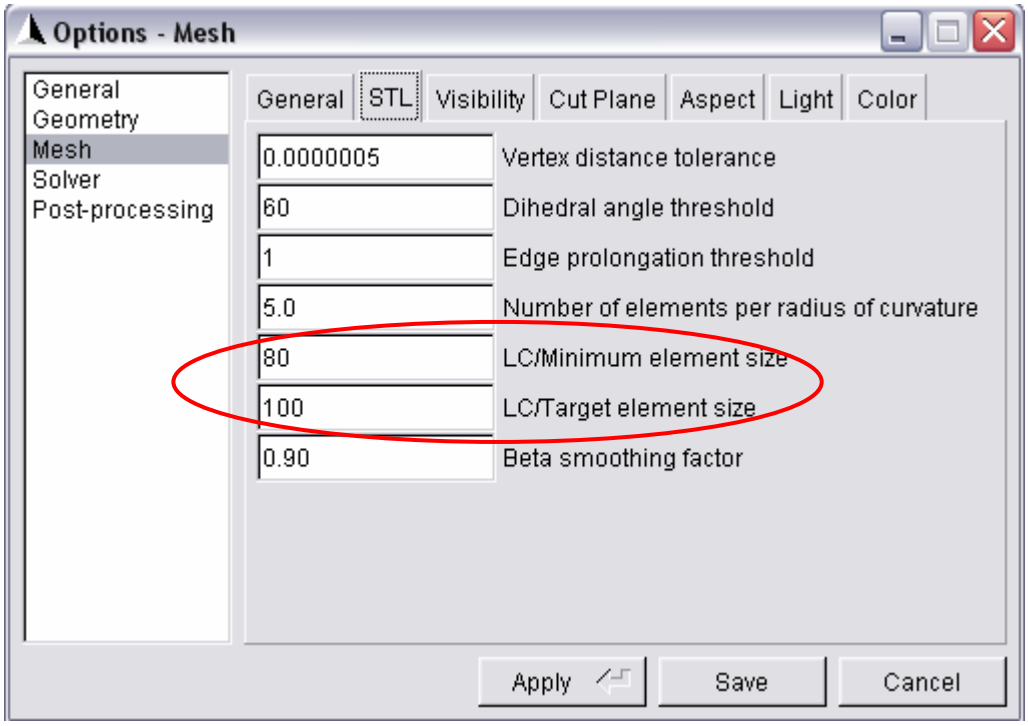


Figura 16. Área de configuração do refinamento no arquivo STL.

Sendo assim, configurando-se estes parâmetros, consegue-se obter uma malha homogênea a contento.

1.3. Bibliografia

GEUZAINE, C., et al. “Gmsh Reference Manual: The documentation for Gmsh 1.60”,2005. Disponível em:
<http://www.geuz.org/gmsh/> Acessado dia: 20/08/2005;

Anexo 2. Desenvolvimento de Cluster de PCs Para Solução de Problemas de Mecânica de Fluidos Computacilonal

2.1. Introdução

O objetivo geral deste tutorial é o desenvolvimento de um cluster (conjunto de PCs iterligados via rede) do tipo Beowulf para o Laboratório de Transferência de Calor e Massa e Dinâmica de Fluidos (LTCM), que resolva problemas contendo uma grande massa de dados.

O LTCM foi criado visando o desenvolvimento de modelos matemáticos, métodos numéricos e técnicas experimentais que possibilitem o estudo da transferência de calor e massa e da dinâmica dos fluidos. Neste contexto, o projeto de Iniciação Científica desenvolvido tem o papel de fornecer base computacional de grande porte e com grande poder de processamento a fim de simular tais modelos.

Dentre os problemas que são abordados no LTCM, os que exigem maior demanda de processamento são os relativos à dinâmica de fluidos, visto a complexidade das equações governantes.

O projeto abrange especificação das máquinas, configuração, manutenção e gerenciamento do cluster, estudo, implententação e análise de programas paralelos utilizando MPI e, finalmente, possibilitar a aplicação de técnicas paralelas em modelos matemáticos complexos como na equação de Navier-Stokes.

O rápido aumento na capacidade de um único processador tem diminuido nos últimos anos. Atualmente, para se obter um grande aumento em velocidade de processamento é necessária a utilização de computadores multiprocessados ou paralelos. A vantagem destes computadores em relação aos supercomputadores que utilizam técnicas de vetorização, é a escalabilidade e o baixo custo. Computadores paralelos disponíveis comercialmente podem ter milhares de processadores, gigabytes de memória e poder computacional medido em gigaflops. No entanto, algoritmos projetados para máquinas seriais podem não ser tão eficientes em máquinas paralelas.

Uma opção para a implementação de um sistema paralelo seria a utilização de máquinas com mais de um processador utilizando memória compartilhada, ou seja, um PC

com dois ou mais processadores utilizando recursos compartilhados como: memória, dísco rígido e periférico. No entanto, esta implementação torna-se inviável quando se deseja escalabilidade no sistema. O número de processadores que pode ser adicionado em uma máquina é limitado.

A melhor opção, considerando as necessidades do laboratório, é a implementação de um cluster de PCs, mais especificamente um cluster Beowulf.

A tecnologia de cluster começou nas máquinas de alto desempenho (supercomputadores). O desenvolvimento dessa tecnologia em PC's foi motivado devido ao alto preço dos supercomputadores, começando em 1994 com o desenvolvimento do cluster Beowulf pela NASA.

2.2. Cluster Beowulf

Em 1994, a NASA, agência espacial norte-americana, necessitava de um equipamento com poder de processamento da ordem de um gigaflop. Uma máquina de tecnologia RISC com tal nível de desempenho custava em torno de um milhão de dólares. Diante disso, os pesquisadores Thomas Sterling e Donald J. Becker decidiram então interligar 16 computadores pessoais, cada um com um microprocessador 486, usando Linux e uma rede ethernet. Este conjunto atingiu a marca de 70 megaflops. Para a época, era uma velocidade comparável a de pequenos supercomputadores comerciais. O custo total do sistema foi de quarenta mil dólares ou seja, dez por cento do preço de uma máquina equivalente para a época.

Uma característica chave de um cluster Beowulf é o uso do Linux e de bibliotecas para troca de mensagens de livre distribuição, permitindo assim alterações no sistema operacional.

Um cluster pode possuir vários tipos de configurações diferentes, tanto na montagem do hardware quanto na configuração do sistema. Os tipos mais comuns de Clusters linux são:

Cluster de processamento paralelo: cada vez que o cluster recebe uma tarefa para executar, já previamente preparada para rodar em processamento paralelo, o cluster divide os pedaços da tarefa para cada uma das máquinas realizar. Dessa forma, com várias máquinas trabalhando ao mesmo tempo em uma única tarefa, o tempo necessário para executá-la torna-se consideravelmente menor.

Cluster de disponibilidade: esse tipo de cluster funciona como um gerenciador de tarefas, ou seja, cada máquina trabalha sozinha, porém a distribuição de tarefas é feita de

tal forma que os processadores estão sempre trabalhando na capacidade total. Nesse tipo de cluster é vital a implementação de um sistema de filas com vários níveis de prioridades diferentes, onde o servidor de filas irá gerenciar qual processador ficará com qual tarefa e o quanto de sua capacidade será utilizado para cada tarefa. Esse tipo de cluster é ideal para trabalhar com grandes quantidades de tarefas que exigem pequenas ou médias capacidades de processamento.

Dentre os tipos de Clusters Linux citados, o que mais se encaixa aos propósitos do LTCM é o Cluster de processamento paralelo, visto a necessidade de dividir tarefas entre vários computadores. Esta necessidade é justificada devido ao grande volume de dados que deverão ser alocados e processados pelo sistema.

2.3. Os passos rumo à paralelização

O primeiro passo consiste em buscar uma plataforma que torne viável o uso de ambientes distribuídos e utilizando apenas produtos de distribuição gratuita. A plataforma escolhida é o Linux, que oferece uma vasta quantidade recursos, e a distribuição inicialmente utilizada foi RedHat 7.1. Foi escolhido, também, a biblioteca MPI para a troca de mensagens entre os processos disponíveis no cluster. Para iniciar os testes foi adquirido 3 computadores idênticos ligados a uma rede Ethernet trabalhando a uma taxa de 100Mbits/s.

Antes de iniciar a caminhada rumo à paralelização de programas voltados ao Laboratório, foram realizados testes usando programas paralelos distribuídos gratuitamente na Internet. Estes testes iniciais foram importantes para se verificar a eficiência do sistema. Comprovado o ganho em desempenho de programas paralelos, passou-se a etapa de implentação de programas que estão relacionados à natureza dos problemas do LTCM. Foi implementado um programa paralelo, usando MPI, que avalia a condução de calor em um domínio tridimensional. Em virtude dos resultados, seguiu-se para uma nova etapa: a implementação do primeiro programa paralelo do LTCM para solução das equações de Navier-Stokes (já estando em fase avançada de testes). Essas equações descrevem com bastante precisão o comportamento do escoamento de fluidos.

Para esta etapa do projeto, o LTCM conta com novas máquinas que são capazes de simular problemas que ultrapassem os 10 milhões de nós. Este trabalho vai detalhar passo a passo a construção deste cluster, desde suas especificações à implentação dos programas paralelos.

2.4. Especificações

2.4.1. Hardware

Um cluster Beowulf pode ser implementado usando máquinas homogêneas ou heterogêneas. Um cluster é dito homogêneo quando todos os seus nós possuem as mesmas características e a mesma rede de comunicação interligando-os. Um cluster heterogêneo possui nós que apresentam características diferentes ou diferentes redes de comunicação entre grupos de máquinas. A implicação é que, em máquinas homogêneas, realizar o balanceamento de carga torna-se mais fácil, visto que todos os nós apresentam as mesmas características. Já em cluster heterogêneo, uma análise mais profunda deve ser feita levando-se em consideração as diferentes capacidades de processamento de cada nó.

Atualmente, o LTCM conta com 5 computadores, um “front-end” (mestre) e quatro escravos com características que o constitui um cluster homogêneo. As configurações são listadas logo abaixo.

Configurações de hardware do mestre:

- Placa-mãe: Intel 865PERL;
- Placa de rede: e1000 1000MB/s on-board;
- Placa de rede: 3Com 100MB/s;
- Processador: Intel Pentium 4 2.8GHz;
- Memória: 1024 MB DDR;
- HD: 80GB IDE;
- CDR-W: HP cd-writer 9300 series;
- Placa de vídeo: Radeon 9200 128 MB DDR AGP8x.
- Monitor: Samsung SincMaster 753v de 17";
- Periféricos: teclado e mouse ótico.

Configurações de hardware dos escravos:

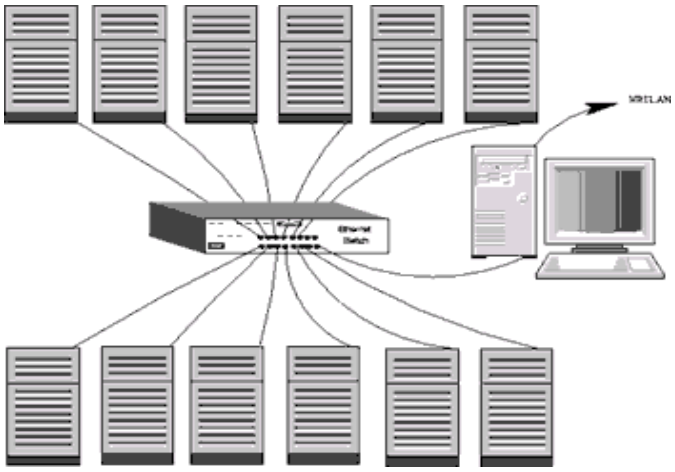
- Placa-mãe: Intel 865PERL;
- Placa de rede: e1000 1000MB/s on-board;
- Processador: Intel Pentium 4 2.8GHz;
- Memória: 1024 MB DDR;
- HD: 80GB IDE;

- CDROM: LG 52X;
- Placa de vídeo: Geforce 64 MB AGP4x.

Estrutura da Rede Local

- Switch: 3COM 8port Gigabit modelo 3C16477
- Cabos: 5 cabos par trançado de 1Gigabit

A arquitetura do cluster pode ser ilustrada na figura abaixo.



Como já foi dito, a escolha das configurações deve-se à enorme demanda de memória e processamento de dados, além da alta comunicação necessária entre os nodos do cluster.

2.4.2. Software

O sistema operacional utilizado é o Linux por ser público, gratuito, e possuir código fonte aberto. A distribuição utilizada é o RedHat 9.0.

O computador principal (front-end) é o equipamento na qual está instalado:

- NFS;
- Serviços de acesso remoto: rlogin, ssh, ftp, rsh, telnet;
- Bibliotecas paralelas: MPICH 1.2.5;

- Compilador Fortran da ntel para Linux;
- Aplicação de gerência de nós: bWatch;
- Analisadores de Performance: NetPipe, NetPerf, Vampir, VampirTrace;
- Aplicativos de visualização: Paraview, Opendx;

Explicitando um pouco mais, o computador mestre funciona como servidor de arquivos para os computadores escravos, utilizando o protocolo NFS , e também como porta de entrada para os usuários que desejem utilizá-lo.

A tarefa dos computadores escravos é a execução exclusiva das tarefas paralelizadas. Em geral, não é necessária a utilização de monitor, mouse e teclado nos computadores escravos, sendo todos eles gerenciados e controlados remotamente pelo computador mestre através de serviços como ssh. Como os computadores escravos não são acessados diretamente, foi reservado o endereçamento IP 192.168.1.1/5 para este fim.

A coordenação das tarefas é feita por troca de mensagens de um computador para outro. Há duas bibliotecas populares de troca de mensagens: a Interface de Passagem de Mensagens (MPI – Message Passing Interface) e a Máquina Virtual Paralela (PVM – Parallel Virtual Machine), ambas disponíveis gratuitamente por seus mantenedores via Internet.

Como a maioria dos programas desenvolvidos no laboratório são em Fortran 90, foi escolhido o compilador Fortran da Intel. Este compilador, além possuir licença gratuita apresenta várias opções de otimização.

Para monitoração foi utilizada a ferramenta bWatch. Sua tarefa é observar a carga e o uso da memória em todos os nós do cluster. Ferramentas de análise como NetPipe também foi instalada para avaliar a performance da rede.

Para a visualização gráfica dos resultados 3D, está sendo utilizado o Paraview e o Opendx, que apresentam características de processamento paralelo.

2.5. Instalação e configuração do cluster

Este tópico descreve, de forma resumida, os passos seguidos para a implentação do cluster. Será abordadas a instalação do Linux, configuração da rede local, configuração de alguns serviços e intalação de pacotes adicionais.

2.5.1. Instalando o Linux

O primeiro passo na montagem do cluster é a instalação do Linux em todas as máquinas. Durante esse processo, é interessante verificar se todos os pacotes necessários estão incluídos, principalmente o NFS e os serviços de acesso remoto (rlogin, ssh, ftp, rsh, telnet). A distribuição Linux escolhida para a implementação do cluster no LTCM é Red Hat 9.

Instalado o Linux em todos os nós, o próximo passo é a configuração da rede, que também pode ser feito durante a instalação, caso tenha sido detectada a placa de rede.

2.5.2. Configurando serviços

2.5.2.1. RSH

A configuração da rede local está definida da seguinte forma:

IP da rede local: 192.168.1.0
Máscara de subrede: 255.255.255.0
Nome do servidor: bwf01
Nomes dos nós escravos: bwf02, bwf03, bwf04 e bwf05.

Os IPs de cada máquina foram definidos da seguinte forma:

bwf01 192.168.1.1
bwf02 192.168.1.2
bwf03 192.168.1.3
bwf04 192.168.1.4
bwf05 192.168.1.5

Em todos os computadores foi adicionado as seguintes linhas ao arquivo /etc/hosts/:

127.0.0.1 localhost localhost.localdomain
192.168.1.1 bwf01
192.168.1.2 bwf02
192.168.1.3 bwf03

192.168.1.4 bwf04
192.168.1.5 bwf05

Este arquivo contém as informações de todos os computadores que estão conectados à rede local.

O próximo passo será configurar o arquivo /etc/hosts.equiv, que define a relação de confiança entre os hosts através de equivalência, sem a necessidade de autenticação por senha. Isto representa um grande risco de segurança, mas é requerido pelo protocolo de acesso remoto RSH (Remote Shell) para que seja possível acessar todas as máquinas do cluster. Este arquivo deve estar presente em todas as máquinas que farão parte do seu sistema. O formato é o seguinte:

bwf01
bwf02
bwf03
bwf04
bwf05

Outro arquivo que deverá ser editado é o arquivo .rhosts. Este arquivo deverá constar em cada diretório de trabalho de cada usuário, como /home e /root. O ponto (.) na frente do arquivo o torna “invisível” ao comando ls. Este arquivo também será usado pelo protocolo RSH para execução de comandos remotos, por algumas aplicações de monitoramento. Seu conteúdo é o seguinte:

bwf01
bwf02
bwf03
bwf04
bwf05

É importante salientar que os arquivos hosts, hosts.equiv e .rhosts devem ser replicados em todas as máquinas que formam o cluster.

Habilite os serviços de rsh, rlogin, ssh e rexec com o comando ntsysv.

Finalmente, acrescente as seguintes linhas ao final do arquivo /etc/securetty/:

rsh
rlogin

2.5.2.2. Instalando e configurando o compilador Fortran da Intel

O compilador Fortran da Intel pode ser baixado, junto com a licença, no *site* http://www.intel.com/software/products/compilers/flin/. Feito o download, basta descompactar e incluir no diretório criado a licença gratuita. A descrição deste tópico refere-se à instalação e configuração do compilador Fortran versão 7.x.

A instalação do pacote é bem simples, basta executar um script de instalação dentro do diretório que foi criado digitando:

```
sh install.sh
```

Erros na instação podem ocorrer devido a falta da licença de uso.

O próximo passo é compiar o arquivo ifc (que está localizado em /opt/intel/compiler70/ia32/bin/ no caso da instalação padrão) em /usr/bin. Isso será necessário posteriormente para que a configuração do mpich verifique a existência do compilador, permitindo que os códigos em Fortran paralelos sejam compilados usando o ifc.

Um problema verificado foi a incompatibilidade entre o compilador Intel ifc versão 7.1 e o Red Hat 9.0 ou outra versão Linux com o mesmo clib (isto está também documentado no *site* da Intel). Para resolver o problema basta criar um arquivo aaa.c em /opt/intel/compiler70/ia32/lib contendo as seguintes linhas:

```
int errno=0;
const unsigned short int __ctype_b = 0;
```

Em seguida o arquivo é compilado da seguinte forma:

```
gcc -O3 -c aaa.c
```

Finalmente edit o arquivo ifc.cfg em /opt/intel/compiler70/ia32/bin da seguinte forma:

```
/opt/intel/compiler70/ia32/lib/aaa.o -Xlinker -rpath -Xlinker /opt/intel/compiler70/ia32/lib -I/opt/intel/compiler70/ia32/include
```

2.5.2.3. Instalando o MPICH

MPICH é uma implementação portável do MPI (biblioteca de passagem de mensagens desenvolvida para ser padrão de comunicação entre processos em ambientes de memória distribuída) distribuída livremente. A versão gratuita desta biblioteca pode ser obtida no *site* http://www-unix.mcs.anl.gov/mpi/mpich/.

O programa baixado pode ser descompactado no diretório /usr/local da seguinte forma:

```
tar -zxvf mpich.tar.gz
```

Agora basta entrar no diretório mpich-1.2.5 recém-criado e executar a seqüência de comandos abaixo:

```
./configure -fc=ifc -f90=ifc --prefix=/usr/local/mpich-1.2.5
make
make install
```

As opções -fc e -f90 permitem que a configuração do mpich inclua a utilização do compilador ifc previamente instalado.

Agora basta editar o arquivo machines.LINUX, localizado no diretório /usr/local/mpich-1.2.5/util/machines/, e acrescentar os nomes dos computadores que farão parte do cluster.

```
bwf01
bwf02
bwf03
bwf04
bwf05
```

Caso se deseje utilizar SSH no lugar de RSH, basta configurar o mpich da seguinte forma:

```
./configure -rsh=ssh -fc=ifc -f90=ifc --prefix=/usr/local/mpich-1.2.5
make
make install
```

2.5.2.4. Configurando o Servidor do Sistema de Arquivos (NFS)

O servidor NFS tem a função de permitir o acesso transparente a discos remotos, centralizando assim a administração, pois é possível ter todos os diretórios de trabalho em uma única máquina e compartilhá-lo com diversos clientes conectados por uma rede local.

Para o cluster do LTCM será necessário o compartilhamento de dois diretórios. Um contém a instalação da versão gratuita da biblioteca MPI (mpich) e o outro a instalação do compilador Fortran da Intel. A utilização do NFS torna desnecessária a instalação destes pacotes em outras máquinas.

Para isso, no computador mestre, o arquivo /etc/exports foi editado acrescentando as seguintes linhas:

```
/usr/local/mpich-1.2.5  *(rw,no_root_squash,sync)
/opt/intel              *(rw,no_root_squash,sync)
```

Com arquivo salvo basta iniciar o serviço NFS digitando o seguinte comando:

```
/etc/init.d/nfs start
```

Para configurar os clientes, em cada nó escravo deve ser editado o arquivo /etc/fstab para montar automaticamente o sistema de arquivos remoto. Para tanto, basta adicionar as seguintes linhas ao final do arquivo:

```
192.168.1.1:/usr/local/mpich-1.2.5 /usr/local/mpich-1.2.5 nfs exec,dev,suid,rw 1 1
192.168.1.1:/opt/intel  /opt/intel nfs exec,dev,suid,rw 1 1
```

Não esquecendo que devem ser criados os diretórios /usr/local/mpich-1.2.5 e /opt/intel, onde serão os pontos de montagem do sistema de arquivos remoto.

E, finalmente, para verificar se o cluster está funcionando basta entrar no diretório /usr/local/mpich-1.2.5/util/ e digitar:

```
cd /usr/local/mpich-1.2.5/util/
./tstmachines -v
```

Uma mensagem como a ilustrada abaixo, indica a correta configuração e o bom funcionamento do cluster.

```
Trying true on bwf01 ...
Trying true on bwf02 ...
Trying true on bwf03 ...
Trying true on bwf04 ...
Trying true on bwf05 ...
Trying ls on bwf01 ...
Trying ls on bwf02 ...
Trying ls on bwf03 ...
Trying ls on bwf04 ...
Trying ls on bwf05 ...
Trying user program on bwf01 ...
Trying user program on bwf02 ...
Trying user program on bwf03 ...
Trying user program on bwf04 ...
Trying user program on bwf05 ...
```

2.6. Desenvolvimento de programas paralelos

Desenvolver programas paralelos baseados em bibliotecas *Message-Passing* não é nada trivial. Por isso, antes de começar a trabalhar com programas paralelos, é preciso entender alguns conceitos fundamentais de processamento paralelo.

2.6.1. Paralelismo

Paralelismo é uma técnica usada para dividir tarefas grandes e complexas em tarefas pequenas obtendo, dessa forma, resultados mais rápidos. Essas tarefas podem ser distribuídas em vários processadores para serem executadas simultaneamente.

As principais vantagens dessa técnica são:

- Aumento do desempenho no processamento;
- Solução de problemas computacionais grandes e complexos.

2.6.2. Ambiente paralelo

O ambiente paralelo consiste em vários processadores interligados em rede, constituído por uma plataforma para manipulação de processos paralelos. Uma das formas de esbelecer a comunicação entre os processadores é por troca de Mensagens (Message Passing). Este é o método de comunicação baseada no envio e recebimento de mensagens através da rede seguindo as regras do protocolo de comunicação entre vários processadores que possuam memória própria. O programador é responsável pela sincronização das tarefas. Como exemplo temos:

PVM - Parallel Virtual Machine
MPI - Message Passing Interface
MPL - Message Passing Library

2.6.3.Programação Paralela

O objetivo da programação paralela é dividir algoritmos grandes e complexos em pequenas tarefas executadas simultaneamente por vários processadores que se comunicam entre si.

De forma bem simples, os passos para se desenvolver um programa paralelo podem ser resumidos da seguinte forma:

- Decompor o algoritmo ou os dados;
- Distribuir as tarefas ou dados entre vários processadores que trabalham simultaneamente;
- Coordenar os processos em execução e a comunicação entre os processadores (sincronização).

2.6.4. Obstáculos do Paralelelismo

2.6.4.1. Sincronização entre os processos

As tarefas executadas em paralelo, num determinado instante, aguardam a finalização mútua para coordenar os resultados ou trocar dados e reiniciar novas tarefas em paralelo.

É necessário que haja a coordenação dos processos e da comunicação entre eles

para evitar que a comunicação seja maior do que o processamento e que consequentemente haja uma queda no desempenho dos processos em execução.

2.6.4.2. Compiladores e ferramentas em amadurecimento

Existem poucos compiladores e ferramentas prontas para paralelização automática que resolvam definitivamente o problema do paralelismo.

2.6.4.3. A conversão de algoritmos seqüenciais em paralelos

Há um considerável tempo gasto do programador em analisar o código fonte para paralelizar e recodificar.

Basicamente é necessário rever o programa seqüencial, avaliar como será particionado, quais os pontos de sincronização, quais as partições que poderão ser executadas em paralelo e qual a forma de comunicação que será empregada entre as tarefas paralelas.

2.6.4.4. Nem tudo é paralelizável

É necessário que haja uma análise do algoritmo a ser paralelizado para seja possível a paralelização nos dados ou nas funções do mesmo, levando sempre em consideração a quantidade de comunicação em relação ao processamento.

2.6.4.5. Tempo de processamento

No processamento paralelo, o tempo de CPU quase sempre aumenta, no entanto pode-se reduzir o tempo total de processamento.

2.6.4.6. Tecnologia Nova

Por ser uma tecnologia recente, há uma dificuldade em se desenvolver aplicações

usando paralelismo.

2.6.4.7. Perda de Portabilidade

A programação paralela utiliza multiprocessadores com arquiteturas de comunicação entre os processadores baseados em memória compartilhada (shared memory) e memória distribuída (distributed memory).

Os programas adaptados a arquitetura de comunicação entre os processadores baseado em memória compartilhada não podem ser executados em uma máquina com memória distribuída, pois não utilizam os mecanismos de troca de mensagens, impossibilitando assim a portabilidade.

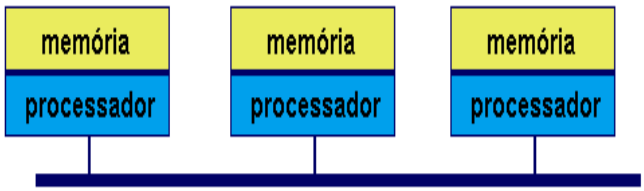
No caso inverso, memória distribuída para memória compartilhada, seria possível a portabilidade; porém o mecanismo de troca de mensagem seria ignorado, pois todos os dados já poderiam ser acessados pelos processadores através da memória compartilhada, isto apenas traria um fluxo desnecessário de comunicação.

2.6.4.8. Depuração *(Debug)*

Os processos são distribuídos e executados em vários processadores simultaneamente, entretanto não existe uma forma eficiente de acompanhar passo-a-passo a alteração das variáveis durante a execução do processamento das diversas tarefas paralelas.

2.6.4.9. Modelo de memória distribuída

Num sistema distribuído os processadores operam independentementes, onde cada um possui sua própria memória. Os dados são compartilhados através da rede usando um mecanismo de troca de mensagens (Message Passing).



2.7. Lei de Amdahl

Lei de Amdahl determina o potencial de aumento de velocidade a partir da porcentagem paralelizável do programa.

$$Speedup = \frac{1}{1 - \%\,paralelizável}$$

Consideremos uma aplicação que leva T unidades de tempo quando executado em modo serial em um único processador.

Quando a aplicação é paralelizável, assumimos que o parâmetro S constitui o percentual serial e P o percentual que pode ser paralelizado.

Assim temos:

Tempo Sequencial:

T = Ts

Tempo Paralelo:

Assumindo N processadores em implementação paralela, o tempo de execução é dado pela seguinte expressão (assumindo speedup perfeito na parte da aplicação que pode ser paralelizada).

$$Tp = \left(S + \frac{P}{N}\right) \bullet T$$

O speedup que é obtido por N processadores é:

$$Sp = \frac{Ts}{Tp} = \frac{1}{S + \frac{P}{N}}$$

Ou seja:

$$Speedup = \frac{1}{\frac{P}{N} + S}$$

Onde:

P = % paralelizável
N = Nº de processadores
S = % serial

Assume-se que a parte serial é fixa, então o speedup para infinitos processadores é limitado em:

Speedup = 1/S

Por exemplo, se Speedup = 10%, então o speedup máximo é 10, até se usarmos um número infinito de processadores.

Tabela Comparativa: Para compreender melhor o significado da Lei de Ahmdahl, veja como se comportariam os casos hipotéticos da tabela abaixo:

N	P=0,50	P=0,90	P=0,99
10	1,82	5,26	9,17
100	1,98	9,17	50,25
1.000	1,99	9,91	90,99
10.000	1,99	9,91	99,02

2.8. Granulosidade

A granulosidade de um aplicação paralela é uma medida usada para indicar a relação entre o tamanho de cada tarefa e o tamanho total do programa, ou seja, é a razão entre computação e comunicação.

2.8.1. Granulosidade fina

Isto ocorre quando um problema é divido em tarefas muito pequenas, ou seja, os cálculos executados em cada processador são mínimos. Uma implementação paralela com esta característica facilita o balanceamento de carga em contrapartida, há uma maior frequência de comunicação, a performance do programa tende a diminuir e o custo de sincronização é alto.

2.8.2. Granulosidade grossa

Neste caso o problema é dividido em tarefas que sejam maiores possíveis. Isto tem como características: aumentar a preformance do programa, processamento menor que a comunicação, menor custo de sincronização, no entanto dificulta o balanceamento de carga.

Para se determinar qual a melhor granulosidade deve-se analisar o grau de dependência entre os processos, definir o grau de sincronização necessário. Em alguns casos a própria natureza do problema pode limitar a granulosidade possível.

2.9. Balanceamento de carga

O balanceamento de carga é um fator importante a ser analisado quando se deseja implementar um programa paralelo. Refere-se à distribuição otimizada dos processos entre os processadores fazendo com que o tempo de execução dos processos em paralelo seja eficiente.

2.10. Escalabilidade

É a propriedade de um sistema paralelo de aumentar o speedup a medida que aumentamos o número de processadores.
Obstáculos:

- *Hardware*

  Um dos maiores gargalos é o tráfego de informação na rede que interliga os processadores. O aumento do número de processadores pode degradar a capacidade de comunicação da rede, diminuindo a performance do programa.

- *Software*

  Alguns algoritmos trabalham melhor em certas escalas de paralelismo. O aumento do número de processadores pode acarretar uma queda na eficiência durante a execução do programa.

2.11. Modelo Message Passing

O modelo de Message Passing é um conjunto de processos que possuem acesso à memória local. As informações são enviadas da memória local do processo para a memória local do processo remoto.

A comunicação dos processos é baseada no envio e recebimento de mensagens.

A transferência dos dados entre os processos requer operações de cooperação entre cada processo de forma que operação de envio deve casar com uma operação de recebimento.

O modelo computacional Message Passing não inclui sintaxe de linguagem nem biblioteca e é completamente independente do hardware. O paradigma de passagem de mensagem apresenta-se apenas como uma das alternativas mais viáveis, devido a muitos pontos fortes como:

- Generalidade: Pode-se construir um mecanismo de passagem de mensagens para qualquer linguagem, como ferramentas de extensão das mesmas (bibliotecas);
- Adequação à ambientes distribuídos.

Porém, apresenta limitações:

- O programador é diretamente responsável pela paralelização;

- Custos de comunicação podem tornar extremamente proibitiva a transmissão de mensagens em um dado ambiente.

2.12. MPI

MPI é uma biblioteca de Message Passing desenvolvida para ambientes de memória distribuída, máquinas paralelas massivas, NOWs (network of workstations) e redes heterogêneas. Define um conjunto de rotinas para facilitar a comunicação (troca de dados e sincronização) entre processos paralelos.

A biblioteca MPI é portável para qualquer arquitetura, tem aproximadamente 125 funções para programação e ferramentas para se analisar a performance.

A biblioteca MPI possui rotinas para programas em Fortran 77 e ANSI C, portanto pode ser usada também para Fortran 90 e C++. Os programas são compilados e linkados a biblioteca MPI.

Todo paralelismo é explícito, ou seja, o programador é responsável por identificar o paralelismo e implementar o algoritmo utilizando chamadas aos comandos da biblioteca MPI.

2.12.1. Conceitos básicos de MPI

Processo:

Por definição, cada programa em execução constitui um processo. Considerando um ambiente multiprocessado, podemos ter processos em inúmeros processadores.

Observação:

O MPI Padrão estipulou que a quantidade de processos deve corresponder à quantidade de processadores disponíveis.

Mensagem (*message*):

É o conteúdo de uma comunicação, formado de duas partes:

- Envelope:

Endereço (origem ou destino) e rota dos dados. O envelope é composto de três parâmetros:

- Identificação dos processos (transmissor e receptor);

- Rótulo da mensagem;
- Comunicator.

- Dado:

Informação que se deseja enviar ou receber. É representado por três argumentos:

- Endereço onde o dado se localiza;
- Número de elementos do dado na mensagem;
- Tipo do dado.

Rank:

Todo o processo tem uma identificação única atribuída pelo sistema quando o processo é inicializado. Essa identificação é contínua representada por um número inteiro, começando de zero até N-1, onde N é o número de processos.

Rank é o identificador único do processo, utilizado para identificar o processo no envio (send) ou recebimento (receive) de uma mensagem.

Group:

Group é um conjunto ordenado de N processos. Todo e qualquer group é associado a um communicator muitas vezes já predefinido como "MPI_COMM_WORLD". Inicialmente, todos os processos são membros de um group com um communicator.

Communicator:

O communicator é um objeto local que representa o domínio (contexto) de uma comunicação (conjunto de processos que podem ser contactados).

O MPI utiliza esta combinação de grupo e contexto para garantir segurança na comunicação e evitar problemas no envio de mensagens entre os processos.

A maioria das rotinas de PVM exigem que seja especificado um communicator como argumento. O MPI_COMM_WORLD é o comunicador predefinido que inclui todos os processos definidos pelo usuário numa aplicação MPI.

Application Buffer:

É o endereço de memória, gerenciado pela aplicação, que armazena um dado que o processo necessita enviar ou receber.

System Buffer:

É um endereço de memória reservado pelo sistema para armazenar mensagens.

Dependendo do tipo de operação de envio/recebimento (send/receive), o dado no buffer da aplicação (application buffer) pode necessitar ser copiado de/para o buffer do sistema (system buffer) através das rotinas send buffer/receive buffer. Neste caso a comunicação é assíncrona.

Blocking Comunnication:

Em uma rotina de comunicação blocking, a finalização da chamada depende de certos eventos.

Em uma rotina de envio de mensagem, o dado tem que ter sido enviado com sucesso, ou ter sido salvo no buffer do sistema (system buffer), indicando que o endereço do buffer da aplicação (application buffer) pode ser reutilizado.

Em uma rotina de recebimento de mensagem, o dado tem que ser armazenado no buffer do sistema (system buffer), indicando que o dado pode ser utilizado.

Non-Blocking Comunnication:

Em uma rotina de comunicação non-blocking, a finalização da chamada não espera qualquer evento que indique o fim ou sucesso da rotina.

As rotinas non-blocking comunnication não esperam pela cópia de mensagens do buffer da aplicação (application buffer) para o buffer do sistema (system buffer), ou a indicação do recebimento de uma mensagem.

Overhead:

Overhead é um desperdício de tempo que ocorre durante a execução dos processos. Os fatores que levam ao overhead são: comunicação, tempo em que a máquina fica ociosa (tempo idle) e computação extra.

Existem os seguintes tipos de overhead:

- *System Overhead*

É o tempo gasto pelo sistema na transferência dos dados para o processo destino.

- *Synchronization Overhead*

É o tempo gasto para a sincronização dos processos.

Observação:

O MPI sempre apresenta um overhead de sicronização inicial, quando espera até que todas as máquinas estejam disponíveis para iniciar o processamento e inicializar os processos. O mesmo ocorre quando da conclusão: existe um delay até que todos os

processos possam encerrar adequadamente e terminarem a execução.

2.12.2. Utilizando o MPI

O primeiro passo para a utilização do MPI é incluir no código fonte a biblioteca MPI. Para programas em C utiliza-se a biblioteca **mpi.h** e para fortran **mpif.h**.

O passo seguinte é escrever o programa fazendo chamadas às rotinas MPICH (versões gratuita do MPI) e compilar com a sintaxe adequada, incluindo os parâmetros desejados.

Para a compilação e linkedição utilizar os comandos para:

Linguagem C (C++)

mpicc [fonte.c] -o [executável] [parâmetros]

O comando mpicc aceita todos os argumentos de compilação do compilador C .

FORTRAN

mpif77 [fonte.f] -o [executável] [parâmetros]

mpif90 [fonte.f90] -o [executável] [parâmetros]

O comando mpif77 e mpif90 aceita todos os argumentos de compilação do compilador Fortran da Intel (ifc).

Para execução de programas paralelos basta utilizar o comando **mpirun** para linguagens C e FORTRAN:

mpirun -[argumentos] [executável]

Argumentos:

h	- Mostra todas as opções disponíveis
arch	- Especifica a arquitetura da(s) máquina(s)
machine	- Especifica a(s) máquina(s)
machinefile	- Especifica o arquivo que contém o nome das máquinas
np	- Especifica o número de processadores
leave_pg	- Registra onde os processos estão sendo executados
nolocal	- Não executa na máquina local
t	- Testa sem executar o programa, apenas imprime o que será executado
dbx	- Inicializa o primeiro processo sobre o dbx

Exemplos:

mpirun -np 5 teste
Executa o programa teste em 5 processadores

Em locais de arquitetura mista, é possível mesclar os parâmetros, de forma a executar sob o MPI, um mesmo programa compilado diferentemente nas várias arquiteturas.

Exemplo:
mpirun -arch risc6000 -np 3 -arch sun4 -np 2 teste.%a onde %a substitui a arquitetura.

2.13. Implementações

Para começar a desenvolver programas paralelos, uma série de etapas foram necessárias a fim de se obter uma base sólida de conhecimento. Testes de performance foram realizados e vários programas implementados. Será destaque neste tópico, duas implementações.

2.14. Processamento paralelo e CFD

Para o projeto de um algoritmo paralelo, a lei de Amdahl diz que para se obter uma alta eficiência, a porção do código que não pode ser paralelizado deve ser muito pequena.

Como o LTCM trabalha com CFD (Computational Fluids Dinamic), a melhor forma de implementar o algoritmo paralelo é subdibidr o domínio da solução em subdomínios e, assim, determinar cada subdomínio para cada processador. Neste caso, o mesmo código roda em todos os processadores e cada um deles executa cálculos sobre seu próprio conjunto de dados. Uma vez que cada processador precisa de dados que residem em outros subdomínios, a troca de dados entre os processadores é necessária.

Métodos explícitos são relativamente fáceis de paralelizar, visto que todas as operações são executadas sobre os dados do passo de tempo anterior. Neste caso, somente é necessário trocar os dados nas regiões de interface entre os subdomínios vizinhos assim que é completado cada passo de tempo. A sequência de operações e os resultados devem ser idênticos sobre um ou vários computadores. A parte mais difícil do problema é, normalmente, a solução da equação elíptica de Poisson para a pressão.

Métodos implícitos são mais difíceis de paralelizar. Enquanto o cálculo da matriz de coeficientes e do termo fonte usa somente dados da iteração anterior e podem ser

calculados eficientemente em paralelo, a solução do sistema de equações lineares não é fácil de paralelizar.

2.15. Bibliografia

CENAPAD-NE: Apostila do Workshop MPI. Fevereiro 1998

http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=733
http://www.clubedohardware.com.br/super.html

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo