ALGORITMOS DE INTEGRAÇÃO TEMPORAL PARA SOLUÇÃO ADAPTATIVA E PARALELA DAS EQUAÇÕES DE NAVIER-STOKES

Fábio César Canesin

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Civil, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Civil.

Orientador: Alvaro L. G. A. Coutinho

Rio de Janeiro
Junho de 2017

ALGORITMOS DE INTEGRAÇÃO TEMPORAL PARA SOLUÇÃO ADAPTATIVA E PARALELA DAS EQUAÇÕES DE NAVIER-STOKES

Fábio César Canesin

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA CIVIL.

Examinada por:

Prof. Alvaro Luiz Gayoso de Azeredo Coutinho, D.Sc.

Dr. José Jerônimo Camata, D.Sc.

Prof. Renato Nascimento Elias, D.Sc.

Prof. Regina Célia Cerqueira de Almeida, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2017

Canesin, Fábio César

Algoritmos de integração temporal para solução adaptativa e paralela das equações de Navier-Stokes/Fábio César Canesin. – Rio de Janeiro: UFRJ/COPPE, 2017.

XI, 46 p.: il.; 29, 7cm.

Orientador: Alvaro L. G. A. Coutinho

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Civil, 2017.

Referências Bibliográficas: p. 37 – 39.

1. Método dos elementos finitos. 2. Método variacional multi escala. 3. Simulação de largas escalas. 4. Newton-Krylov livre de Jacobiano. 5. Formulas de diferenciação retrógradas. I. Coutinho, Alvaro L. G. A.. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Civil. III. Título.

À Carla, por tudo.

Agradecimentos

Gostaria de agradecer a todos que de alguma forma ajudaram na realização dessa dissertação:

- A Schlumberger, por ter dado a flexibilidade necessária.
- Ao meu professor e orientador Alvaro L. G. A. Coutinho, pelo voto de confiança, apoio e paciência.
- Ao co-orientador José J. Camata, pelo código inicial e referências das quais esse trabalho se desenvolveu e ideias que contribuíram para a qualidade da implementação.
- Ao colega Adriano Cortes por ajudar a desvendar o FEMSystem e a equipe e orientados do Núcleo de Atendimento em Computação de Alto Desempenho (NACAD) que me receberam com cordialidade e criaram o material original no qual busquei apoio.

A minha esposa Carla pela compreensão e suporte, sem quem eu nada teria.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ALGORITMOS DE INTEGRAÇÃO TEMPORAL PARA SOLUÇÃO ADAPTATIVA E PARALELA DAS EQUAÇÕES DE NAVIER-STOKES

Fábio César Canesin

Junho/2017

Orientador: Alvaro L. G. A. Coutinho

Programa: Engenharia Civil

Escoamentos turbulentos são dominantes em aplicações industriais e no cotidiano, devido às vantagens inerentes as características do escoamento e a dificuldade de manutenção de escoamentos no regime laminar. A simulação numérica de escoamentos turbulentos apresenta desafios intrínsecos em função do elevado custo computacional para representação das estruturas do escoamento e não linearidades presentes. Na literatura existem metodologias diversas para um tratamento analítico das contribuições de pequenas escalas com o objetivo de atingir um custo computacional aceitável aos recursos disponíveis. Uma das metodologias mais recentes é o modelo variacional multi-escala (VMS) que traz a vantagem de não necessitar de um processo de filtragem de escalas, bem como ser uma generalização de estabilizações do tipo Petrov-Galerkin em formulações de elementos finitos. O presente trabalho busca avaliar formulações VMS atuais para a equação incompressível de Navier-Stokes junto de métodos de marcha no tempo de integração retrograda (BDF) em primeira e segunda ordem quanto a qualidade dos resultados e desempenho computacional, tanto escalabilidade como consumo energético. Foi ainda utilizada a solução do sistema linear de forma livre da formação da matriz Jacobiana (JFNK). A implementação foi realizada utilizando a biblioteca de código livre **libMesh**, escrita em C++ a biblioteca oferece diversas facilidades para o desenvolvimento eficiente para computação de alto desempenho, as execuções foram realizadas no supercomputador Lobo Carneiro e no protótipo Mont-Blanc, que faz uso de tecnologias emergentes.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

TIME INTEGRATION ALGORITHMS FOR PARALLEL AND ADAPTIVE SOLUTION OF NAVIER-STOKES EQUATIONS

Fábio César Canesin

June/2017

Advisor: Alvaro L. G. A. Coutinho

Department: Civil Engineering

Turbulent flows are dominant in industrial and everyday applications, due to the inherent advantages of flow characteristics and the difficulty of maintaining flow in the laminar regime. The numerical simulation of turbulent flows presents intrinsic challenges due to the high computational cost for representing the flow structures. In the literature there are several methodologies for an analytical characterization of the contribution of small scales with the objective of achieving a computational cost acceptable to the available resources. One of the most recent methodologies is the variational multi-scale modeling (VMS), which has the advantage of not requiring a filtering of small-scale effects as well as being a generalization of the Petrov-Galerkin stabilization in finite element formulations. The present work seeks to characterize the computational performance of the turbulent flows formulation with VMS modeling for the incompressible Navier-Stokes equations, the first and second order backward difference formulas (BDF) methods are compared, as well as the solution of the linear system using the Jacobian free Newton-Krylov (JFNK) technique and a semi-implicit strategy making use of the BDF discretization. The implementation of the studied reference case was performed using the opensource library **libMesh**, written in C ++ the library offers several facilities for efficient development of high-performance computing solvers.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Dinâmica dos fluidos computacional

O rápido avanço da capacidade de processamento dos computadores nas últimas décadas transformou o modo como engenheiros e pesquisadores realizam análises quantitativas em dinâmica de fluídos, fazendo-se cada vez mais o uso da dinâmica dos fluidos computacional, comumente chamada de CFD (do inglês *Computational Fluid Dynamics*).

Dentro do conjunto de técnicas utilizadas em CFD o Método dos Elementos Finitos (MEF), desenvolvido desde meados de 1950 na área de engenharia aeronáutica, tem se demonstrado como uma das mais poderosas alternativas (DONEA e HUERTA [6]). Recentemente a aplicação do MEF tem sido realizada com sucesso em virtualmente todas as áreas de CFD, com especial destaque nesse trabalho para desenvolvimentos relacionados a modelagem do fenômeno da turbulência.

De forma resumida os aspectos físicos da CFD são governados pelas equações de conservação aplicadas a dinâmica de fluidos, sendo essas:

- Conservação da massa;
- Conservação da quantidade de movimento (momento);
- Conservação da energia.

Na literatura as equações de conservação são geralmente descritas por equações diferenciais parciais (EDP), sendo cada EDP dotada de suas peculiaridades que refletem de forma direta na forma como são expressas e solucionadas através do MEF.

No contexto da dinâmica dos fluidos os estudos são geralmente divididos conforme as características dos fluidos e do escoamento em questão, sendo as classificações mais usuais para os escoamentos:

- Compressíveis, quando existe variação da massa específica no tempo e/ou espaço;
- Incompressíveis, quando a variação da massa específica é desprezada sem prejuízo à análise;
- Turbulento, quando existem flutuações na pressão e/ou momento;
- Laminar, quando não existem flutuações na pressão e/ou momento.

Os fluidos são geralmente classificados quanto a resposta dos esforços viscosos devido a taxa de deformação, sendo separados em dois grandes grupos:

- Newtonianos - quando os esforços viscosos são uma relação linear com a taxa deformação;
- Não-Newtonianos - quando os esforços viscosos são uma relação não linear com a taxa deformação.

O presente trabalho é dedicado a escoamentos incompressíveis turbulentos de fluidos Newtonianos.

1.2 Computação de alto desempenho

Apesar do mencionado avanço da capacidade de processamento os computadores pessoais ainda não são capazes de serem utilizados de forma prática para a solução de problemas complexos de CFD (mais de dezenas de milhões de graus de liberdade). Para tais cenários é necessária a utilização de sistemas especializados, construídos de forma a fazer uso de hardware e sistemas com desempenho em ordens de magnitude acima dos disponíveis nos computadores pessoais. A aplicação de tais soluções é chamada de computação de alto desempenho, comumente referida como HPC (do inglês *High Performance Computing*).

Os sistemas de HPC em sua imensa maioria fazem a utilização de sistemas paralelos, nos quais um grande conjunto de computadores (chamados de nós de cálculo) trabalham cooperativamente em um problema. Diversas metodologias de cooperação já foram propostas e aplicadas com variado sucesso em HPC. Quando observadas quanto à aplicação na área de CFD a forma dominante de cooperação é o conceito de troca de mensagens com memória distribuída, materializado no padrão MPI (do inglês *Message Passing Interface*).

Esse trabalho utilizou-se de dois sistemas: o supercomputador Lobo Carneiro, instalado na COPPE/UFRJ e o protótipo Mont-Blanc, instalado no centro de supercomputação de Barcelona, ambos foram utilizados com o padrão MPI.

Juntamente do grande ganho de performance alcançável com a aplicação do MPI em sistemas de HPC existe uma considerável escalada da complexidade de programação. Essa complexidade acaba por colocar um maior stress no desenvolvimento de pesquisas em CFD e pode prejudicar a produtividade dos pesquisadores. Para solucionar tal problema diversos grupos vêm trabalhando na produção de bibliotecas de programação para programas científicos e de engenharia que se encarregam de ocultar as complexidades do MPI e expõem uma interface mais amigável de desenvolvimento. Entre tais bibliotecas se destaca a PETSc, que vem sendo desenvolvida desde 1995 nos Estados Unidos no laboratório nacional Argonne (BALAY *et al.* [7]). Na PETSc é exposto um diverso conjunto de métodos e estruturas de dados para a solução paralela de sistemas lineares, não-lineares e aplicações relacionadas à discretização de equações diferenciais parciais.

1.3 Método dos elementos finitos para escoamentos incompressíveis turbulentos

Os principais métodos numéricos para solução de EDPs consistem na aproximação das equações diferenciais por sistemas algébricos de equações. O MEF se encontra no conjunto de métodos que trabalham com formulação variacional das equações de conservação. Tal formulação é comumente conhecida como forma fraca.

De forma simplificada o MEF consiste na aproximação das integrais resultantes da aplicação do método dos resíduos ponderados (MRP) no conjunto de EDPs descrito na forma fraca (DONEA e HUERTA [6]) como uma soma de subdivisões do espaço, chamados de elementos.

O sucesso na utilização do MEF para solução de problemas em CFD tem sido cada vez maior devido às vantagens apresentadas na representação de domínios complexos e na grande comunidade de pesquisadores dedicados a estudar o método.

O modelo variacional multi-escala (VMS) aplicado a turbulência para simulações de grandes escalas (LES) foi primeiramente apresentado por HUGHES *et al.* [2] como uma evolução dos estudos realizados na fundamentação teórica dos métodos estabilizados do MEF para aplicações em CFD. O grande diferencial da metodologia frente aos desenvolvimentos passados, segundo BAZILEVS *et al.* [4], é o fato de não ser necessário nenhum tratamento especial para o tensor de tensões das velocidades de grande escala, e por consequência a diminuição da complexidade no tratamento de condições de contorno. A modelagem VMS já foi estudada no NACAD por LINS *et al.* [8] onde é demonstrada a implementação do VMS já se valendo de um código otimizado através apenas da modificação dos parâmetros de estabilização e em GUERRA *et al.* [9] onde o VMS é aplicado a escoamentos com transporte de

partículas. Recentemente AHMED *et al.* [10] e RASTHOFER e GRAVEMEIER [11] realizaram extensa revisão da metodologia VMS. De forma geral o método consiste na aplicação de uma projeção nas equações escritas na forma variacional, separando os efeitos em duas ou três diferentes escalas, as menores escalas são então modeladas em função de parâmetros das grande escalas e considerações energéticas. Diversas formas de modelagem para os efeitos de pequenas escalas foram propostas, fazendo uso da definição de viscosidade de subescala, da solução de um problema equivalente na malha e na representação em função dos resídios de larga escalas, chamada de modelo variacional multi-escala baseado em resíduos (RB-VMS, do inglês *Residual Based Variational Multi-Scale*) . No presente trabalho, foi utilizada a metodologia RB-VMS com separação em duas escalas.

O presente trabalho utiliza ambas as formulações implícita propostas por BAZILEVS *et al.* [4] e a formulação semi-implícita de FORTI e DEDÈ [12], proposta para reduzir o custo computacional decorrente das não-linearidades presentes no problema.

1.4 Solução no tempo das equações discretas

A descrição física de problemas relacionados a dinâmica dos fluidos é geralmente dependente do tempo, sendo assim, as EDPs resultantes contém termos com derivadas no tempo. Após a aplicação do MEF para discretização do domínio existem duas opções: aplicar o MEF novamente para discretizar o tempo; ou aplicar o método das diferenças finitas (MDF). A grande maioria das formulações utilizadas em problemas práticos se utiliza do MDF para a discretização dos termos dependentes do tempo, e a formulação espaço-tempo é geralmente restrita a pesquisas no desenvolvimento do MEF em si próprio.

Como resultado se dá origem a soluções e pesquisas dedicadas a integração do conjunto de equações diferenciais ordinárias (ODE) resultantes da aplicação do MDF para discretização dos termos dependentes do tempo. No presente trabalho um dos itens estudados é o impacto na utilização de diferentes formas de integração no tempo.

1.5 Motivação

A grande maioria dos escoamentos de líquidos em aplicações de ciência e engenharia são turbulentos, sendo a modelagem de escoamentos incompressíveis adequada para grande parcela dos mesmos. Alguns exemplos de aplicação são:

- Escoamento de líquidos em dutos e tubulações industriais;

- Escoamentos aerodinâmicos em velocidades subsônicas;
- Aplicações em hemodinâmica;
- Aplicações em hidrodinâmica.

Além destes, outro ponto importante é que um dos produtos de escoamentos turbulentos é a indução de vibração em estruturas como pontes e sondas marítimas.

Dessa forma, avançar no desempenho e formulações para a descrição de escoamentos incompressíveis turbulentos é de crucial importância prática.

1.6 Objetivo

Avaliar o desempenho numérico e computacional para a integração temporal do sistema de equações diferenciais ordinárias no tempo, resultantes da aplicação do método de elementos finitos para a simulação de escoamentos incompressíveis em regime turbulento com modelagem RB-VMS de duas escalas. Verificar a eficácia na convergência das não linearidades do sistema, eficiência na implementação e desempenho paralelo das metodologias de Krylov-Newton livre da matriz Jacobiana (JFNK) e discretização semi-implícita com fórmulas de diferenciação retrógrada (BDF).

Ao final busca-se avaliar a estabilidade do sistema quanto a seleção do passo temporal para a solução de um caso representativo da aplicação do solver Navier-Stokes com turbulência VMS, busca-se também contribuir para a eficiência na implementação mediante documentação da reorganização necessária para adaptação do presente solver do NACAD/COPPE-UFRJ ao framework **FEMSystem** da biblioteca **libMesh**.

1.7 Metodologia

Por meio da formulação apresentada em FORTI e DEDÈ [12] e BAZILEVS *et al.* [4], partindo do código anteriormente desenvolvido no NACAD/COPPE-UFRJ utilizando a biblioteca libMesh (KIRK *et al.* [5]), o presente trabalho apresentará a seguinte metodologia:

- Desenvolvimento teórico das equações governantes;
- Desenvolvimento da formulação de elementos finitos;
- Desenvolvimento da formulação VMS;
- Desenvolvimento da formulação BDF semi-implícita;

- Apresentação dos algorítimos de solução do sistema não-linear;
- Desenvolvimento do caso referência;
- Verificação do modelo proposto;
- Análise dos resultados;
- Conclusão; e
- Propostas de trabalhos futuros.

A linguagem de programação aplicada no desenvolvimento é a C++ devido a utilização do pacote libMesh e as vantagens em reutilização de código e gerenciamento de desenvolvimento proporcionadas pela programação orientada a objetos (POO). Para o pós-processamento dos resultados é empregado o visualizador Paraview (http://www.paraview.org/). O desenvolvimento do trabalho como um todo se beneficia da ferramenta de controle de versão git, dessa forma é possível que trabalhos futuros se aproveitem da documentação das modificações realizadas.

A organização do presente trabalho segue a mesma sequência da metodologia de desenvolvimento proposta, sendo o capítulo 2 encarregado de apresentar a formulação matemática para as equações incompressíveis de Navier-Stokes quando aplicadas a escoamentos turbulentos, introduzindo quais deficiências a modelagem variacional multi escala tenta solucionar quando comparado ao método de simulação de largas escalas amplamente utilizado. O capítulo 3 aplica o método dos elementos finitos a formulação do capítulo anterior e de posse da formulação variacional desenvolve a modelagem VMS, no entanto o capítulo não trata da discretização no tempo tarefa esta delegada ao capítulo 4 que apresenta as formulações implícita e semi-implícita a serem estudadas bem como discute a solução de sistemas não lineares de forma a justificar a investigação do método livre de matriz Jacobiana dentro do presente trabalho. O capítulo 5 se encarrega de elucidar como as formulações finais obtidas no capítulo 4 são implementadas de forma prática dentro da biblioteca libMesh. Por fim o capítulo 6 apresenta os resultados da comparação de desempenho no caso referência e o capítulo 7 oferece conclusões e proposições de trabalhos futuros.

Capítulo 2

Formulação matemática

A representação das equações utilizadas no presente trabalho são descritas em um referencial Euleriano, onde a velocidade $\mathbf{v}$ é descrita para um ponto no espaço e assim associada apenas ao do ponto material instantaneamente correspondente. Essa forma acarreta em uma simplificação quanto a aplicação do MEF para representação dos complexos movimentos do fluxo turbulento por dispensar a necessidade de modificação da malha computacional, porém a formulação Euleriana introduz o operador convectivo que é acompanhado de diversos desafios numéricos (DONEA e HUERTA [6]).

Como em FORTI e DEDÈ [12] o presente trabalho utiliza uma descrição segundo a mecânica do contínuo clássica escrita de forma dimensional, a adimensionalização das equações facilita diversas analises quanto a estrutura e propriedades das mesmas, porém o desenvolvimento de simuladores adimensionais dificulta a aplicação de um mesmo solver a diversas geometrias complexas onde a definição dos parâmetros de adimensionalização não são evidentes.

2.1 Equações de conservação

O conjunto de equações para a conservação do momento do escoamento transiente de um fluido viscoso é dado pelas equações Eq. 2.1, a condição de incompressibilidade é data pela Eq. 2.2 sendo ambas equações definidas em uma região do espaço $\Omega \subset \mathbb{R}^3$ e no tempo $t \subset (0, \mathbb{T})$.

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) - \nabla \cdot \sigma(\mathbf{u}, p) = \mathbf{f} \tag{2.1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2.2}$$

A Eq. 2.2 é chamada também de equação da continuidade e representa a con-

servação de massa. Na Eq. 2.1 ∇ é o operador gradiente [1], $\otimes$ é o produto tensorial [2], ρ é a massa específica, σ é o tensor de tensões, p é a pressão e $\mathbf{f}$ é o vetor de forças de volume. O tensor de tensões para escoamentos Newtonianos é dado pela Eq. 2.3 sendo diretamente proporcional às taxas de deformação ϵ dada por Eq. 2.4, a constante μ é a viscosidade dinâmica e representa a taxa de difusão de quantidade movimento devido a características intrínsecas do fluido.

$$\sigma(\mathbf{u}, p) = -p\mathbf{I} + 2\mu\epsilon(\mathbf{u}) \tag{2.3}$$

$$\epsilon(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + (\nabla\mathbf{u})^T) \tag{2.4}$$

2.2 Equações de Navier-Stokes para escoamentos incompressíveis

A substituição das equações Eq. 2.4 e 2.3 na Eq. 2.1 juntamente com a condição de incompressibilidade resulta nas equações de Navier-Stokes (ENS) para fluidos Newtonianos incompressíveis Eq. 2.5, onde Δ é o operador de Laplace [3].

$$\rho\frac{\partial\mathbf{u}}{\partial t} + \rho\mathbf{u}\cdot\nabla\mathbf{u} + \nabla p - \mu\Delta\mathbf{u} = \mathbf{f} \tag{2.5}$$

2.3 Condições de Contorno e Iniciais

Dada uma região Ω no espaço fechada pela fronteira suave Γ (Fig. 2.1) as condições de contorno (CC) para as ENS são dadas pelas equações Eq. 2.6 e Eq. 2.7, onde Γ_D é o subconjunto de Γ onde são aplicadas condições de contorno essenciais (Dirichlet) e Γ_N é o subconjunto onde são aplicadas condições de contorno naturais (Neumann) sendo $\Gamma_D = \Gamma \setminus \Gamma_D$.

$$\mathbf{u} = \mathbf{g} \qquad \text{em} \quad \Gamma_D \times (0, T) \tag{2.6}$$

$$\sigma(\mathbf{u}, p)\mathbf{n} = \mathbf{h} \qquad \text{em} \quad \Gamma_N \times (0, T) \tag{2.7}$$

Na Eq. 2.7 $\mathbf{n}$ é o vetor unitário normal ao contorno. As condições iniciais são dadas pela Eq. 2.8.

[1] Definição: $\nabla = \frac{\partial}{\partial x_1}\mathbf{i} + \frac{\partial}{\partial x_2}\mathbf{j} + \frac{\partial}{\partial x_3}\mathbf{k}$, onde $\mathbf{i}$, $\mathbf{j}$ e $\mathbf{k}$ são os vetores unitários cartesianos.
[2] Definição: $[\mathbf{u} \otimes \mathbf{v}]_{ij} = u_i v_j$
[3] Definição: $\Delta = \nabla^2 = \nabla \cdot \nabla$

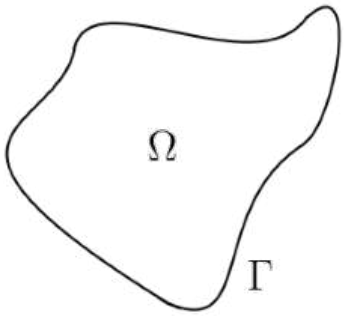

Figura 2.1: Região Ω fechada por Γ

$$\mathbf{u}(0) = \mathbf{u}_0 \qquad \text{em} \quad \Omega \times \{0\} \tag{2.8}$$

2.4 Turbulência

Turbulência é o termo utilizado para descrever variações tridimensionais observadas no movimento de fluidos, representada na Fig. 2.2. A turbulência se manifesta desde escalas microscópicas como medido no fluxo sanguíneo em artérias de ratos (MAY *et al.* [13]) até o movimento observado em imagens da atmosfera de Júpiter. Seu estudo científico e artístico é relatado há mais de dois mil anos, como no tratado de Lucretius *De rerum natura*. Apesar do grande número de estudos uma justificativa matemática formal completa da turbulência ainda não foi realizada, sendo um dos grandes problemas em aberto da mecânica do contínuo (BENZI e FRISCH [14]).

Figura 2.2: Turbulência em um feixe, modificado de DUISBURG-ESSEN [1].

2.4.1 Simulação de Largas Escalas

Atualmente a modelagem mais utilizada para a simulação numérica da turbulência é a chamada simulação de largas escalas (LES, do inglês *Large Eddy Simulation*) , que é baseada na teoria de cascata de energia de Kolmogorov e consiste na aplicação de um filtro nas ENS para separação dos efeitos de sub-escala (HUGHES *et al.* [2]). A aplicação de um filtro $\mathbf{g}$ de suporte $D_{\Delta_\mathbf{x}}$ como na Eq. 2.9 separa a velocidade $\mathbf{u}$ em uma componente de baixa frequência $\overline{\mathbf{u}}$ que representa a contribuição das largas escalas e uma componente de alta frequência $\mathbf{u}'$ que representa as pequenas escalas (Eq. 2.10), como exemplificado na Fig. 2.3.

$$\overline{\mathbf{u}}(\mathbf{x},t) = \int_{D_\Delta(\mathbf{x})} g(\mathbf{x},\mathbf{y})\mathbf{u}(\mathbf{y},t)d\mathbf{y} \tag{2.9}$$

$$\mathbf{u}' = \mathbf{u} - \overline{\mathbf{u}} \tag{2.10}$$

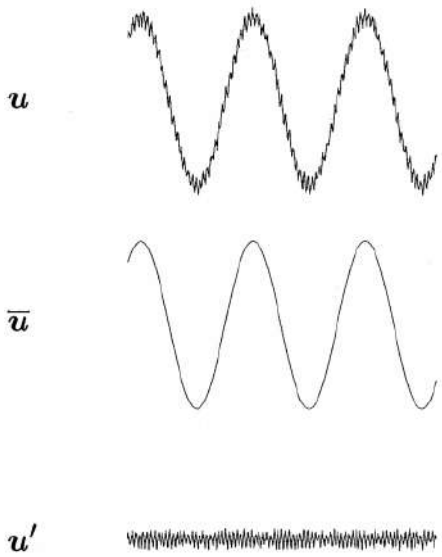

Figura 2.3: Filtragem da velocidade, de HUGHES *et al.* [2].

A aplicação nas ENS do referido processo de filtragem resulta na Eq. 2.11, onde o termo $\nabla \cdot \mathbf{T}$ é o chamado tensor de stress de sub-escala originário da filtragem do tensor de stress de Reynolds, sendo a diferença da filtragem do produto tensorial das velocidades e o produto tensorial das velocidades filtradas (Eq. 2.12).

$$\rho\frac{\partial\overline{\mathbf{u}}}{\partial t} + \rho\overline{\mathbf{u}}\cdot\nabla\overline{\mathbf{u}} + \nabla\overline{p} - \mu\Delta\overline{\mathbf{u}} = \mathbf{f} + \nabla\cdot\mathbf{T} \tag{2.11}$$

$$\mathbf{T} = \overline{\mathbf{u}}\otimes\overline{\mathbf{u}} - \overline{\mathbf{u}\otimes\mathbf{u}} \tag{2.12}$$

Para a solução da Eq. 2.11 é necessária a definição de mais uma equação para

T (problema de fechamento); Diversos modelos são propostos na literatura sendo o mais popular o modelo de viscosidade turbulenta de Smagorinsky (ν_S). A dificuldade na definição robusta de ν_S em termos de parâmetros de larga escala introduz a primeira grande fragilidade de modelos LES (HUGHES *et al.* [2]).

A segunda grande fragilidade do modelo LES (compartilhada com outros modelos que aplicam o conceito de filtragem) é a necessidade de tratamento especial aos contornos e regiões próximas do contorno. Uma das razões para tal necessidade é o fato da região de suporte do filtro ultrapassar os contornos do domínio como exemplificado na Fig. 2.4, pois soluções utilizando filtros dinâmicos são possíveis mas acarretam em complicações na implementação e custo computacional bem como na formulação matemática do problema.

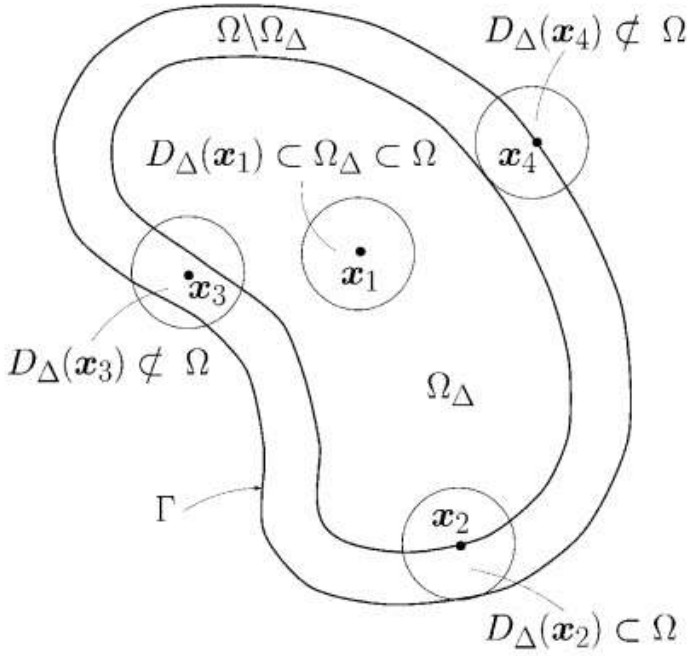

Figura 2.4: Intercessão entre contorno e suporte do filtro, de HUGHES *et al.* [2].

2.4.2 Modelo Variacional Multi-Escala

Uma formulação alternativa para as ENS incompressíveis em regime turbulento - que ao invés de utilizar o processo filtragem utiliza operadores de projeção para realizar a separação dos espaços de alta e baixa frequência quando as ENS são escritas na sua formulação variacional - foi proposta em HUGHES *et al.* [2] como uma generalização do processo de estabilização. A aplicação do método na simulação de regimes turbulentos foi analisada e consideravelmente expandida em BAZILEVS *et al.* [4] e finalmente os parâmetros de estabilização foram posteriormente estudados em HSU *et al.* [15]. O conjunto dos três referidos trabalhos formam uma solução

atrativa para escoamentos turbulentos incompressíveis, pois a formulação resultante não exige tratamento especial dos contornos nem definição de viscosidades turbulentas e, por fim, a formulação resultante é também válida no regime laminar.

Como o desenvolvimento e definição da formulação VMS dependem de exprimir as ENS na sua forma variacional, estas serão apresentadas no capítulo seguinte juntamente a aplicação do MEF.

Capítulo 3

Formulação de elementos finitos

3.1 Soluções aproximadas

De forma geral os métodos computacionais utilizados para a solução de EDPs em geometrias complexas resolvem uma versão aproximada do problema, que resulta de forma diferente em função do método utilizado. Nos métodos baseados em resíduos ponderados (MRP), tais como o método dos volumes finitos, método dos elementos de contorno, método dos elementos finitos e colocação, a aproximação é originária da interpolação dos domínios (espaço e tempo para as ENS) e das variáveis de solução (velocidade e pressão para as ENS). Nesses métodos uma incógnita $U(\mathbf{x})$ é aproximada como na Eq. 3.1, onde ϕ_i é a função de interpolação no nó i e $\mathbf{u_i}$ é o valor da variável.

$$U(\mathbf{x}) \approx U^v = \sum_{i=1}^{n_{\text{nós}}} \phi_i(\mathbf{x})\mathbf{u_i} \tag{3.1}$$

Os métodos baseados no MRP podem ser divididos em dois grandes grupos: os métodos com e sem malha. Malha é o nome dado a uma representação discreta da região de interesse Ω, onde a região é subdividida em diversos elementos Ω_e, cada método possui limitações diferentes quanto a forma dos elementos que a compõe a malha utilizada Ω_e, sendo que quando aplicada em domínios complexos resultam muitas vezes em uma aproximação do domínio $\Omega_P \approx \Omega$ como mostrado na Fig. 3.1. Métodos livre de malha são em sua maioria baseados na discretização do domínio utilizando o conceito de partícula, que representa pontos de interpolação no domínio e contorno.

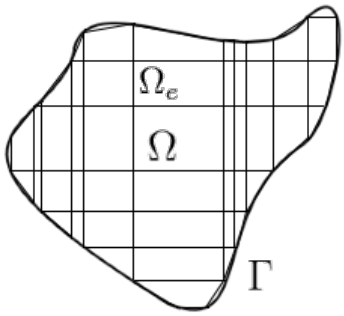

Figura 3.1: Discretização do domínio Ω em elementos Ω_e

3.2 Elementos finitos

O método dos elementos finitos baseia-se na formulação variacional do problema, onde o domínio Ω é representado por uma malha Ω_p com elementos Ω_e (Eq. 3.2) que não se sobrepõem (DONEA e HUERTA [6]), como formalizado pela Eq. 3.3. No presente trabalho é utilizada a formulação isoparamétrica em quem velocidade, pressão e espaço todos compartilham as mesmas funções de interpolação ϕ_i, chamadas de funções de base, de suporte compacto, significando que a função i para o nó j é tal que $\phi_i(x_j) = \delta_{ij}$ [1], exemplificado na Fig. 3.2.

$$\sum_{i=1}^{n_{\text{elem}}} \Omega_e = \Omega_p \tag{3.2}$$

$$\Omega_{e_i} \cap \Omega_{e_j} = \varnothing, \quad \forall i \neq j \tag{3.3}$$

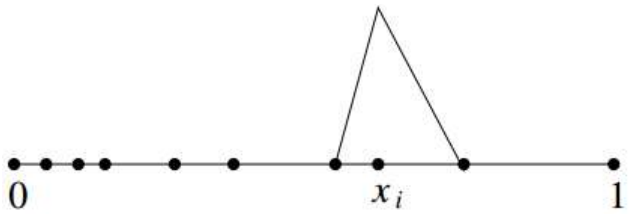

Figura 3.2: Função de interpolação ϕ_i com suporte compacto, de BRENNER e SCOTT [3].

O suporte compacto de ϕ_i e a subdivisão do domínio dada pela Eq. 3.2 nos permite representar as integrais da formulação variacional como uma soma da integral em cada elemento:

$$\mathbf{K} = \sum_{e=1}^{n_{\text{elem}}} \mathbf{K}_e \tag{3.4}$$

[1] $\delta_i j$ é o delta de Kronecker

onde

$$\mathbf{K}_e = \int_{\Omega_e} f(U^v) d\Omega \tag{3.5}$$

A necessidade de realizar a integração em cada elemento leva com que em aplicações práticas do MEF os elementos sejam representados em um espaço referencia ξ onde a definição das funções de base no elemento ϕ_N^e são bem definidas e a integração é realizada de forma numérica, utilizando alguma lei de quadratura onde os pesos W_i são dependentes do tipo de elemento utilizado e do mapeamento afim $\xi \rightarrow \Psi(\xi)$ de $\mathbf{K}$ para $\mathbf{K}_e$.

3.3 Equações incompressíveis de Navier-Stokes

Reescrevemos primeiramente as ENS pela Eq. 3.6, onde $\nu = \mu/\rho$ e $\mathbf{F} = \mathbf{f}/\rho$, dessa forma simplificamos algumas operações na implementação das ENS no MEF.

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) + \frac{1}{\rho}\nabla p - \nu \Delta \mathbf{u} = \mathbf{F} \tag{3.6}$$

O método dos elementos finitos baseia-se na formulação variacional do problema (BRENNER e SCOTT [3]), sendo o espaço de funções infinitos $\mathcal{V}_g$ e $\mathcal{V}_0$ dados respectivamente pelas Eq. 3.7 e Eq. 3.8. Sendo o operador $(\cdot, \cdot)$ o produto interno L^2 a formulação variacional das ENS é dada pela Eq. 3.9

$$\mathcal{V}_g := \{\mathbf{u} \in H^1(\Omega) \quad : \quad \mathbf{u}|_{\Gamma_D} = \mathbf{g}\} \tag{3.7}$$

$$\mathcal{V}_0 := \{\mathbf{u} \in H^1(\Omega) \quad : \quad \mathbf{u}|_{\Gamma_D} = \mathbf{0}\} \tag{3.8}$$

$$(\mathbf{w}, \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) + \frac{1}{\rho}\nabla p - \nu \Delta \mathbf{u}) + (q, \nabla \cdot \mathbf{u}) = (\mathbf{w}, \mathbf{F}) + (\mathbf{w}, \mathbf{h})_{\Gamma_N} \tag{3.9}$$

Deve-se então encontrar $\mathbf{U} = \{\mathbf{u}, p\} \in \mathcal{V}_g$, para todo $\mathbf{W} = \{\mathbf{w}, q\} \in \mathcal{V}_0$, sendo a possibilidade da escolha de infinitos $\mathbf{W}$ a razão de ser chamada formulação variacional (BRENNER e SCOTT [3]).

Utilizando a integração por partes no produto interno podemos reescrever a Eq. 3.9 como a Eq. 3.10, onde $\nabla^s = (\nabla(\cdot) + \nabla(\cdot)^T)/2$.

$$\begin{aligned}(\mathbf{w}, \frac{\partial \mathbf{u}}{\partial t}) - (\nabla \mathbf{w}, \mathbf{u} \otimes \mathbf{u}) - (\nabla \cdot \mathbf{w}, p/\rho) + (\nabla^s \mathbf{w}, 2\nu \nabla^s(\mathbf{u})) \\ + (q, \nabla \cdot \mathbf{u}) = (\mathbf{w}, \mathbf{F}) + (\mathbf{w}, \mathbf{h})_{\Gamma_N}\end{aligned} \tag{3.10}$$

Na Eq. 3.10 foi imbutida a concervação da massa e transportado o operador ∇ para a função de interpolação, de forma a não se ter a derivação de segunda ordem nem derivadas em p, permitindo assim bases lineares para $\mathbf{u}$. Pela formulação variacional permitir esse transporte da derivada da incógnita para a função de interpolação ela é conhecida como forma fraca.

A aplicação do MEF nas ENS como na Eq. 3.10 possui instabilidades que impõem restrições quanto a interpolação dos campos de velocidades e pressão. Formulações estabilizadas como SUPG/PSPG (*Streamline Upwind Petrov-Galerkin/Pressure Stabilized Petrov-Galerkin*) foram desenvolvidas para contornar os problemas de equações com operador convectivo, sendo o VMS proposto como uma generalização das estabilizações do tipo Petrov-Galerkin (BAZILEVS *et al.* [4]).

3.4 Formulação variacional multi-escala

A formulação VMS utilizada é baseada na separação em duas escalas e na modelagem RB-VMS (RASTHOFER e GRAVEMEIER [11]), partindo da formulação variacional, representa-se o espaço de solução como a soma direta das soluções de largas escalas e de pequenas escalas, utilizando uma projeção para largas escalas $\mathbb{P}$ temos a Eq. 3.11, onde $\mathbb{I}$ é o operador identidade, η^h é uma variável projetada em largas escalas e η' é a contribuição de pequenas escalas da mesma.

$$\begin{aligned} \mathcal{V} &= \mathcal{V}^h \oplus \mathcal{V}' \\ \mathbb{P} &: \mathcal{V} \rightarrow \mathcal{V}^h \\ \eta^h &= \mathbb{P}\eta \\ \eta' &= (\mathbb{I} - \mathbb{P})\eta \end{aligned} \tag{3.11}$$

onde $\mathcal{V}$ pode ser $\mathcal{V}_g$ ou $\mathcal{V}_0$, η é um incógnita qualquer, η^h é a contribuição de largas escalas, η' é a contribuição de pequenas escalas e $\mathbb{I}$ é o operador identidade.

Reescrevendo a Eq. 3.10 como:

$$A(\mathbf{W}, \mathbf{U}) = B(\mathbf{W}, \mathbf{F}) \tag{3.12}$$

onde

$$\begin{aligned} A(\mathbf{W}, \mathbf{U}) = (\mathbf{w}, \frac{\partial \mathbf{u}}{\partial t}) - (\nabla \mathbf{w}, \mathbf{u} \otimes \mathbf{u}) \\ -(\nabla \cdot \mathbf{w}, p/\rho) + (\nabla^s \mathbf{w}, 2\nu \nabla^s(\mathbf{u})) + (q, \nabla \cdot \mathbf{u}) \end{aligned} \tag{3.13}$$

$$B(\mathbf{W},\mathbf{F}) = (\mathbf{w},\mathbf{F}) + (\mathbf{w},\mathbf{h})_{\Gamma_N} \tag{3.14}$$

Aplicando as relações das Eqs. 3.11 temos:

$$A(\mathbf{W}^h,\mathbf{U}^h+\mathbf{U}') = B(\mathbf{W}^h,\mathbf{F}) \tag{3.15}$$

$$A(\mathbf{W}',\mathbf{U}^h+\mathbf{U}') = B(\mathbf{W}',\mathbf{F}) \tag{3.16}$$

Sendo a Eq. 3.15 a parcela representada na malha. Como $\mathbf{U}'$ representa as contribuições de pequenas escalas na Eq. 3.15 estas precisam ser aproximadamente representadas em termos dos campos de largas escalas sendo resolvidos. Em HUGHES *et al.* [2] a metodologia do VMS é apresentada juntamente com uma formulação onde $\mathbf{U}'$ é calculado através de parâmetros de larga escala mas ainda é introduzida uma viscosidade turbulenta ν_t. BAZILEVS *et al.* [4] realiza uma aproximação algébrica para a solução analítica de $\mathbf{U}'$ como dos campos de largas escalas, resultando na Eq. 3.17 e representado na Fig. 3.3.

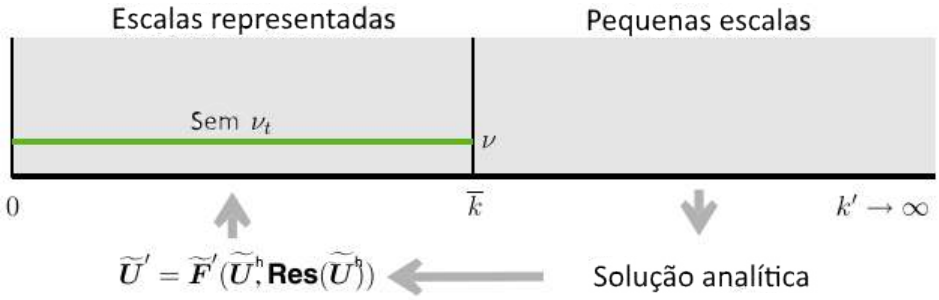

Figura 3.3: Contribuições de pequenas escalas, modificado de BAZILEVS *et al.* [4]

$$\mathbf{U}' \approx \tilde{\mathbf{U}}' = -\tau \mathbf{Res}(\mathbf{U}^h) \tag{3.17}$$

Na Eq. 3.17 τ é a matriz de parâmetros de estabilização Eq. 3.18, onde τ_M é o parâmetro de estabilização da conservação do momento e τ_C da conservação da massa, $\mathbf{Res}(\mathbf{U}^h)$ é o vetor de resíduos de largas escalas Eq. 3.19, onde $\mathbf{r}_M$ é o resíduo da conservação do momento e $\mathbf{r}_C$ é o resíduo da conservação do massa.

$$\tau = \begin{bmatrix} \tau_M \mathbf{I}_{3\times 3} & \mathbf{0}_3 \\ \mathbf{0}_3^T & \tau_C \end{bmatrix} \tag{3.18}$$

$$\mathbf{Res}(\mathbf{U}^h) = \begin{Bmatrix} \mathbf{r}_M(\mathbf{u}^h, p^h) \\ \mathbf{r}_C(\mathbf{u}^h) \end{Bmatrix} \tag{3.19}$$

sendo no presente trabalho:

$$\mathbf{r}_M(\mathbf{u}^h, p^h) = \frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h + \frac{1}{\rho}\nabla p^h - \nu \Delta \mathbf{u}^h - \mathbf{F}^h, \tag{3.20}$$

$$\mathbf{r}_C(\mathbf{u}^h) = \nabla \cdot \mathbf{u}^h, \tag{3.21}$$

$$\tau_M = \tau_M(\mathbf{u}^h) = (\frac{4}{\Delta t^2} + \mathbf{u}^h \cdot \mathbf{G}\mathbf{u}^h + C_l \nu^2 \mathbf{G} : \mathbf{G})^{-1/2}, \tag{3.22}$$

$$\tau_C = (\tau_M \mathbf{g} \cdot \mathbf{g})^{-1}, \tag{3.23}$$

$$\mathbf{G} = G_{ij} = \sum_{k=1}^{3} \frac{\partial \xi_k}{\partial x_i} \frac{\partial \xi_k}{\partial x_j} \tag{3.24}$$

$$\mathbf{g} = g_i = \sum_{j=1}^{3} \frac{\partial \xi_j}{\partial x_i} \tag{3.25}$$

sendo Δt o passo no tempo e $C_l = 10$.

Substituindo a Eq. 3.17 na Eq. 3.15 com as funções peso constantes no tempo, $\mathbf{u}' = 0$ em Γ e $(\nabla^s \mathbf{w}^h, 2\nu \nabla^s \mathbf{u}')_\Omega = 0$ temos a formulação variacional final para as ENS com VMS data pela Eq. 3.26.

$$A(\mathbf{W}^h, \mathbf{U}^h)^{Galerkin} + A(\mathbf{W}^h, \mathbf{U}^h)^{SUPG} + A(\mathbf{W}^h, \mathbf{U}^h)^{VMS} = B(\mathbf{W}^h, \mathbf{F}) \tag{3.26}$$

onde $A^{Galerkin}$ e F são dados pela Eq. 3.13 substituindo $\mathbf{U}$ por $\mathbf{U}^h$ e as parcelas A^{SUPG} e A^{VMS} surgem da substituição de $\mathbf{U}'$ pela Eq. 3.18. Dividimos a equação dessa forma pois naturalmente recuperamos os tradicionais parâmetros de estabilização SUPG, dados pela Eq. 3.27, e termos adicionais que representam os termos do tensor de tensões de Reynolds, dados pela Eq. 3.28 (BAZILEVS *et al.* [4]).

$$\begin{aligned} A(\mathbf{W}^h, \mathbf{U}^h)^{SUPG} = (\mathbf{u}^h \cdot \nabla \mathbf{w}^h + \nabla q^h, \tau_M \mathbf{r}_M(\mathbf{u}^h, p^h)) \\ +(\nabla \cdot \mathbf{w}^h, \tau_C \mathbf{r}_C(\mathbf{u}^h) \end{aligned} \tag{3.27}$$

$$\begin{aligned} A(\mathbf{W}^h, \mathbf{U}^h)^{VMS} = (\mathbf{u}^h \cdot (\nabla \mathbf{w}^h)^T, \tau_M \mathbf{r}_M(\mathbf{u}^h, p^h)) \\ -(\nabla \mathbf{w}^h, \tau_M \mathbf{r}_M(\mathbf{u}^h, p^h) \otimes \tau_M \mathbf{r}_M(\mathbf{u}^h, p^h)) \end{aligned} \tag{3.28}$$

Capítulo 4

Métodos de marcha temporal

No capítulo 3 ao introduzir o método dos elementos finitos foi dito que a mesma interpolação utilizada para o espaço pode também ser aplicada para o tempo, esse tipo de formulação é chamado de espaço-tempo. Apesar de facilitar algumas análises a respeito das propriedades do VMS (BAZILEVS *et al.* [4] e HUGHES *et al.* [2]) implementações espaço-tempo possuem aplicações a problemas tridimensionais limitadas, sendo então usual a aplicação do método das diferenças finitas para a discretização do tempo.

No presente trabalho utilizamos uma forma de interpolação baseada em diferenças finitas chamada de fórmulas de diferenciação retrógrada (BDF, FORTI e DEDÈ [12]), dada pela Eq. 4.1, onde o tempo t é dividido em intervalos Δt sendo cada intervalo chamado de t_n para $n = 0, 1, .., N_t$ e $t_n = t_{n-1} + \Delta t$, α_β é uma constante que depende da ordem β do método dada pela Eq. 4.3, ψ é a variável sendo discretizada. A seleção de $\beta = 1$ equivalente a formulação de Euler, de aplicação recorrente em CFD (DONEA e HUERTA [6]).

$$\frac{\partial \psi}{\partial t} \approx \frac{\alpha_\beta \psi_{n+1} - \psi_{nBDF,\beta}}{\Delta t} \tag{4.1}$$

onde

$$\psi_{nBDF,\beta} = \begin{cases} \psi_n & n \geq 0, \quad \text{para} \quad \beta = 1 \\ 2\psi_n - \frac{1}{2}\psi_{n-1} & n \geq 1, \quad \text{para} \quad \beta = 2 \\ 3\psi_n - \frac{3}{2}\psi_{n-1} + \frac{1}{3}\psi_{n-2} & n \geq 2, \quad \text{para} \quad \beta = 3 \end{cases} \tag{4.2}$$

e

$$\alpha_\beta = \begin{cases} 1, & \text{para} \quad \beta = 1 \\ 3/2, & \text{para} \quad \beta = 2 \\ 11/6, & \text{para} \quad \beta = 3 \end{cases} \tag{4.3}$$

Após realizar a discretização da derivada no tempo precisamos decidir como

avaliar as variáveis dependentes $\mathbf{u}^h$ e p^h nos outros termos da equação. Existindo três opções que dão origem as formulações: explícita, implícita e semi-implícita.

Se Δt é pequeno podemos tratar a equação apenas como uma marcha no tempo e utilizar $\mathbf{u}^h_{n-1}$ e p^h_{n-1}, tal formulação é chamada explícita e não será tratada no presente trabalho, metodologias tratando da mesma quanto a estabilidade e propriedades na sua aplicação para as ENS são apresentadas em DONEA e HUERTA [6].

4.1 Formulação implícita

A formulação implícita é comumente escolhida devido a sua estabilidade, nela selecionamos $\mathbf{u}^h$ e p^h como $\mathbf{u}^h_{n+1}$ e p^h_{n+1}, dando origem a um sistema matricial onde a escolha de Δt não é importante para a estabilidade.

Aplicando a formulação implicita e a Eq. 4.1 na Eq. 3.26 resulta na forma discreta no tempo dada pela Eq. 4.4 onde omitimos o subscrito $n+1$ e o sobrescrito h, sendo assim $\mathbf{u} = \mathbf{u}^h_{n+1}$ e $p = p^h_{n+1}$.

$$\begin{aligned}
(\mathbf{w}, \frac{\alpha_\beta \mathbf{u} - \mathbf{u}_{nBDF,\beta}}{\Delta t}) - (\nabla \mathbf{w}, \mathbf{u} \otimes \mathbf{u}) \\
+(\nabla^s \mathbf{w}, 2\nu \nabla^s(\mathbf{u})) - (\nabla \cdot \mathbf{w}, p/\rho) \\
+(q, \nabla \cdot \mathbf{u}) \\
+(\mathbf{u} \cdot \nabla \mathbf{w} + \nabla q, \tau_M \breve{\mathbf{r}}_M(\mathbf{u}, p)) \\
+(\nabla \cdot \mathbf{w}, \tau_C \mathbf{r}_C(\mathbf{u}) \\
+(\mathbf{u} \cdot (\nabla \mathbf{w})^T, \tau_M \breve{\mathbf{r}}_M(\mathbf{u}, p)) \\
-(\nabla \mathbf{w}, \tau_M \breve{\mathbf{r}}_M(\mathbf{u}, p) \otimes \tau_M \breve{\mathbf{r}}_M(\mathbf{u}, p)) \\
= (\mathbf{w}, \mathbf{f}) + (\mathbf{w}, \mathbf{h})_{\Gamma_N}
\end{aligned} \tag{4.4}$$

onde

$$\breve{\mathbf{r}}_M(\mathbf{u}, p) = \frac{\alpha_\beta \mathbf{u} - \mathbf{u}_{nBDF,\beta}}{\Delta t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{\rho} \nabla p - \nu \Delta \mathbf{u} - \mathbf{f} \tag{4.5}$$

A Eq. 4.4 é não linear em $\mathbf{u}^h_{n+1}$ e p^h_{n+1} para cada passo de tempo t_n, a solução do sistema é computacionalmente custosa sendo o desempenho da simulação fortemente dependente da técnica de solução empregada para o sistema não linear. FORTI e DEDÈ [12] sugerem uma formulação alternativa semi-implícita para contornar o alto custo computacional de formação da matriz Jacobiana e do vetor residual quando da aplicação do método de Newton para solução do sistema, tal formulação é elucidada a seguir.

4.2 Formulação semi-implícita

A formulação proposta por FORTI e DEDÈ [12] é baseada na extrapolação utilizando polinômios retrógrados de Newton-Gregory. Para as ordens $\beta = 1, 2$ e 3 a extrapolação ψ_{n+1}^{NG} de ψ_{n+1} é dada pela Eq. 4.6.

$$\psi_{n+1,\beta} = \begin{cases} \psi_n & \text{se} \quad n \geq 0, \quad \text{para} \quad \beta = 1 \\ 2\psi_n - \psi_{n-1} & \text{se} \quad n \geq 1, \quad \text{para} \quad \beta = 2 \\ 3\psi_n - 3\psi_{n-1} + \psi_{n-2} & \text{se} \quad n \geq 2, \quad \text{para} \quad \beta = 3 \end{cases} \tag{4.6}$$

Aplicando a extrapolação para $\mathbf{u}_{n+1}^h$ e p_{n+1}^h nos termos não lineares se obtêm a Eq. 4.7, onde omitimos o subscrito $n+1$ e o sobrescrito h. Sendo assim o sistema resultante é linear em $\mathbf{u}_{n+1}^h$ e p_{n+1}^h, precisando dessa forma ser resolvido apenas uma vez para cada passo de tempo.

$$\begin{aligned} (\mathbf{w}, \frac{\alpha_\beta \mathbf{u} - \mathbf{u}_{nBDF,\beta}}{\Delta t}) - (\nabla \mathbf{w}, \mathbf{u}_\beta \otimes \mathbf{u}) \\ +(\nabla^s \mathbf{w}, 2\nu \nabla^s(\mathbf{u})) - (\nabla \cdot \mathbf{w}, p/\rho) \\ +(q, \nabla \cdot \mathbf{u}) \\ +(\mathbf{u}_\beta \cdot \nabla \mathbf{w} + \nabla q, \tau_M^\beta \mathbf{r}_M^\beta) \\ +(\nabla \cdot \mathbf{w}, \tau_C^\beta \mathbf{r}_C) \\ +(\mathbf{u}_\beta \cdot (\nabla \mathbf{w})^T, \tau_M^\beta \mathbf{r}_M^\beta) \\ -(\nabla \mathbf{w}, \tau_M^\beta \hat{\mathbf{r}}_M^\beta \otimes \tau_M^\beta \tilde{\mathbf{r}}_M^\beta) \\ -(\nabla \mathbf{w}, \tau_M^\beta \hat{\mathbf{r}}_M^\beta \otimes \tau_M^\beta \frac{\alpha_\beta \mathbf{u}}{\Delta t}) \\ +(\nabla \mathbf{w}, \tau_M^\beta \mathbf{r}_M^\beta \otimes \tau_M^\beta \frac{\mathbf{u}_{nBDF,\beta}}{\Delta t}) \\ = (\mathbf{w}, \mathbf{f}) + (\mathbf{w}, \mathbf{h})_{\Gamma_N} \end{aligned} \tag{4.7}$$

onde

$$\tau_M^\beta := (\frac{4}{\Delta t^2} + \mathbf{u}_\beta \cdot \mathbf{G} \mathbf{u}_\beta + C_l \nu^2 \mathbf{G} : \mathbf{G})^{-1/2}, \tag{4.8}$$

$$\tau_C^\beta := (\tau_M^\beta \mathbf{g} \cdot \mathbf{g})^{-1}, \tag{4.9}$$

$$\mathbf{r}_M^\beta = \frac{\alpha_\beta \mathbf{u} - \mathbf{u}_{nBDF,\beta}}{\Delta t} + \mathbf{u}_\beta \cdot \nabla \mathbf{u} + \frac{1}{\rho} \nabla p - \nu \Delta \mathbf{u} - \mathbf{f}, \tag{4.10}$$

$$\hat{\mathbf{r}}_M^\beta = \mathbf{r}_M^\beta(\mathbf{u}_\beta, p_\beta), \tag{4.11}$$

$$\tilde{\mathbf{r}}_M^{\beta} = \mathbf{u}_{\beta} \cdot \nabla \mathbf{u} + \frac{1}{\rho} \nabla p - \nu \Delta \mathbf{u} - \mathbf{f}, \tag{4.12}$$

4.3 Solução do sistema não linear

A solução do sistema de equações não lineares resultantes da discretização da Eq. 4.4 no presente trabalho é feita através da aplicação do método iterativo de Newton.

O método de Newton para uma função $\mathbf{F}(\mathbf{u})$ consiste de iterar a expansão de Taylor a partir de $\mathbf{u}$ na iteração k:

$$\mathbf{F}(\mathbf{u}_{k+1}) = \mathbf{F}(\mathbf{u}_k) + \mathbf{F}'(\mathbf{u}_k)(\mathbf{u}_{k+1} - \mathbf{u}_k) + O(\Delta \mathbf{u})^2 \tag{4.13}$$

Desprezando os termos de segunda ordem e igualando o lado esquerdo a zero podemos escrever a Eq. 4.14 onde $\mathbf{J} \equiv \mathbf{F}'$ é a matriz Jacobiana e $\delta \mathbf{u}_k \equiv \mathbf{u}_{k+1} - \mathbf{u}_k$.

$$\mathbf{J} \delta \mathbf{u}_k = -\mathbf{F}(\mathbf{u}_k) \tag{4.14}$$

Resolve-se então a Eq. 4.14 para $\delta \mathbf{u}_k$ e utilizando a relação $\mathbf{u}_{k+1} = \mathbf{u}_k + \delta \mathbf{u}_k$ movese para a próxima iteração $k+1$. O iterador k é incrementado de $k = 0$ até que um numero máximo de iteração seja atingindo n_{iter}, o resíduo não linear atinja a tolerância tol_{res} dada pela Eq. 4.15 ou a atualização da variável seja menor que a tolerânca tol_{update} dada pela Eq. 4.16.

$$\frac{\|\mathbf{F}(\mathbf{u}_k)\|}{\|\mathbf{F}(\mathbf{u}_0)\|} < tol_{res} \tag{4.15}$$

$$\frac{\|\delta \mathbf{u}_k\|}{\|\mathbf{u}_k\|} < tol_{update} \tag{4.16}$$

A solução da da Eq. 4.14 é geralmente feita de forma a utilizar eficientemente as características da matriz Jacobinada e do resíduo, métodos de subespaço de Krylov são particularmente populares, em especial o variações do método GMRES para o caso de matrizes não simétricas (KNOLL e KEYES [16]).

4.4 Newton-Krylov livre de matriz Jacobiana

Para solução do sistema linear $\mathbf{A}\mathbf{x} = \mathbf{b}$ os métodos de Krylov iterativos utilizam a projeção no subespaço $\mathbf{K}_j$ de dimensão j dado pela Eq. 4.17, onde $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$.

$$\mathbf{K}_j = span(\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, ..., \mathbf{A}^{j-1}\mathbf{r}_0) \tag{4.17}$$

Aplicando a metodologia a Eq. 4.14 em uma iteração k, temos então:

$$\mathbf{r}_0 = -\mathbf{F}(\mathbf{u}) - \mathbf{J}\delta\mathbf{u}_0 \tag{4.18}$$

A aplicação do GMRES consiste ao problema em minimizar $\|\mathbf{J}\delta\mathbf{u}_j + \mathbf{F}(\mathbf{u})\|_2$ para cada iteração j do método, onde $\delta\mathbf{u}$ é dado pela equação Eq. 4.19 e β_i são escalares que minimizam o resíduo.

$$\delta\mathbf{u}_j = \delta\mathbf{u}_0 + \sum_{i=0}^{j-1} \beta_i(\mathbf{J})^i\mathbf{r}_0 \tag{4.19}$$

Examinando as Eq. 4.18 e 4.19 pode-se observar que a aplicação GMRES exige apenas operações com o Jacobiano do tipo produto matriz-vetor. Utilizando uma expansão de Taylor em primeira ordem podemos escrever o produto do Jacobiano $\mathbf{J}$ pelo vetor $\mathbf{v}$ através da Eq. 4.20, onde ϵ é uma pertubação pequena.

$$\mathbf{J}\mathbf{v} \approx [\mathbf{F}(\mathbf{u} + \epsilon\mathbf{v}) - \mathbf{F}(\mathbf{u})]/\epsilon \tag{4.20}$$

A Eq. 4.20 é a chamada forma livre de matriz, sendo a formação do Jacobiano $\mathbf{J}$ dispensada, na prática a Eq. 4.14 é resolvida na forma pré-condicionada dada pela Eq. 4.21.

$$(\mathbf{J}\mathbf{P}^{-1})(\mathbf{P}\delta\mathbf{u}) = -\mathbf{F}(\mathbf{u}) \tag{4.21}$$

O sistema é então resolvido para $\mathbf{w}$ na forma da Eq. 4.22 e em seguida para $\delta\mathbf{u}$ dado pela Eq. 4.23.

$$(\mathbf{J}\mathbf{P}^{-1})\mathbf{w} = -\mathbf{F}(\mathbf{u}) \tag{4.22}$$

$$\delta\mathbf{u} = \mathbf{P}^{-1}(\mathbf{w}) \tag{4.23}$$

Sendo o produto contendo o Jacobiano é dado pela aproximação da Eq. 4.24. Dessa forma ainda necessária a formação da matriz de precondicionamento $\mathbf{P}$, e a formulação é como um todo chamada de livre de matriz Jacobiana.

$$(\mathbf{J}\mathbf{P}^{-1})\mathbf{v} \approx [\mathbf{F}(\mathbf{u} + \epsilon\mathbf{P}^{-1}\mathbf{v}) - \mathbf{F}(\mathbf{u})]/\epsilon \tag{4.24}$$

Diversas técnicas para o cálculo da matriz de pré-condicionamento $\mathbf{P}$ e para a pertubação ϵ são propostas na literatura, sendo uma revisão da técnicas realizada por KNOLL e KEYES [16]. O presente trabalho utiliza as técnicas de pré-condicionamento implementadas na biblioteca `PETSc`, uma apresentação das mesmas pode ser consultada em GASTON *et al.* [17]. A vantagem de utilizar a biblioteca `PETSc` se da na facilidade de realizar experimentos

numéricos com diferentes configurações de precondicionadores, quando utilizando uma forma com formação de Jacobiano basta utilizar o commando de linha `-pc_type` para selecionar entre uma variedade de precondicionadores. Na formulação PJFNK do solver não linear SNES é necessário utilizar o comando `-snes_mf_operator` e configurar a matriz de precondicionamento corretamente na função `SNESSetJacobian`. A assinatura simplificada da função é `SNESSetJacobian(SNES snes, Mat Amat, Mat Pmat, PetscErrorCode, void)`, aqui `Amat` é a matriz Jacobiana e `Pmat` é a matriz de pré-condicionamento. No presente trabalho `Amat` foi aproximada utilizando diferenças finitas no vetor de resíduos.

Capítulo 5

Aspectos da implementação

O desenvolvimento de simuladores eficientes para aplicações em computação de alto desempenho envolve a intercessão de diversas áreas da ciências da computação, matemática e engenharia. Sendo necessária a colaboração de grupos e o estabelecimento de tecnologias de suporte, que acelerem o processo de desenvolvimento sem comprometer a qualidade da analise e desempenho dos simuladores. Sendo assim se faz necessária a utilização e desenvolvimento de bibliotecas dedicadas a funcionalidades especificas que integram os solvers, no presente trabalho a discretização e gerenciamento das estruturas de dados relacionadas a malha são realizadas utilizando a biblioteca `libMesh`, já a solução do sistema não linear fica sobre a responsabilidade da biblioteca `PETSc`.

5.1 A biblioteca `libMesh`

A biblioteca `libMesh` for criada com o objetivo de facilitar a implementação de métodos numéricos com malha para a solução de EDPs, em especial o MEF. Seu principal objetivo é integrar e simplificar a utilização das bibliotecas científicas de mais alta qualidade disponíveis (KIRK *et al.* [5]). A biblioteca é desenvolvida no modelo de software aberto, é desenvolvida na linguagem C++ fazendo extenso uso da técnica de metaprogramação de forma a prover abstrações que facilitam o desenvolvimento de solvers complexos sem comprometer o desempenho na execução.

Dentre as opções de bibliotecas para o MEF a `libMesh` diferencia-se por apresentar suporte robusto ao desenvolvimento de solvers habilitados ao refinamento e desrefinamento adaptativo da malha (AMR/C), podendo fazer uso de diversos estimadores de erro e algorítimos para seleção dos elementos a serem adaptados já implementados ou facilitando a utilização de algorítimos codificados pelo usuário. A principal limitação quanto ao AMR/C é o fato de não ser possível o desrefinamento em nível menor que o fornecido pela malha original, nível 0 na Fig. 5.1.

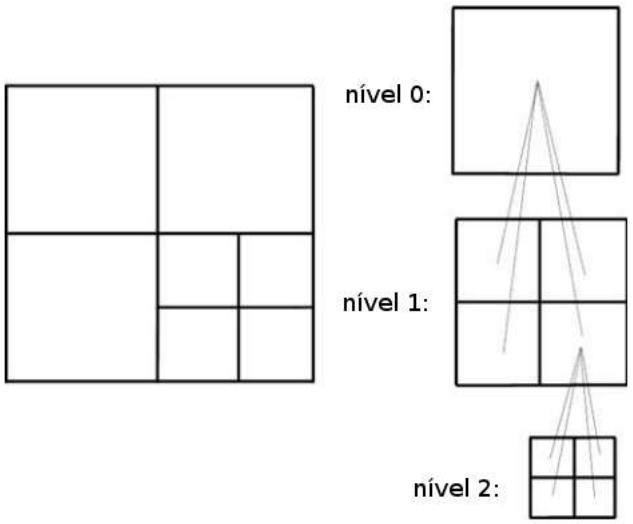

Figura 5.1: Hierarquia dos níveis de refino, adaptado de KIRK *et al.* [5].

5.2 Framework FEMSystem

Elevando a abstração das classes básicas do `libMesh` foi construída um framework específico para aplicações de elementos finitos, chamado de `FEMSystem`. O objetivo do `FEMSystem` é prover uma interface de desenvolvimento de aplicações (API) para o MEF consistente e independente da física (BAUMAN e STOGNER [18]), para isso primeiramente o problema precisa ser escrito na forma da Eq. 5.1

$$\begin{aligned} M(\mathbf{u})\dot{\mathbf{u}} &= F(\mathbf{u}), \quad \text{em} \quad \Omega \times (0,T) \\ G(\mathbf{u}) &= 0, \quad \text{em} \quad \Omega \times (0,T) \\ \mathbf{u}(\mathbf{x},0) &= \mathbf{u}_0(\mathbf{x}), \quad \forall \mathbf{x} \in \bar{\Omega} \\ \mathbf{u}(t) &= \mathbf{g}(t), \quad \text{em} \quad \Gamma_D \times (0,T) \\ \sigma(\mathbf{u},t)\cdot\mathbf{n} &= h(t), \quad \text{em} \quad \Gamma_N \times (0,T) \end{aligned} \tag{5.1}$$

A partir da formulação Eq. 5.1 se constrói a forma variacional equivalente dada pela Eq. 5.2. O usuário deve então criar uma subclasse do `FEMSystem` e reimplementa (*overload*) os métodos segundo a tabela 5.1. A implementação é realizada do ponto de vista do ponto de integração no domínio do elemento, sendo o processo de integração e montagem da matriz (*assembly*) delegados para a biblioteca.

$$\begin{aligned} M(\mathbf{u}_h, \dot{\mathbf{u}}_h; \mathbf{v}_h) &= F(\mathbf{u}_h; \mathbf{v}_h), \quad \forall \mathbf{v}_h \in \mathbf{V}^h \\ G(\mathbf{u}_h; \mathbf{v}_h) &= 0, \quad \forall \mathbf{v}_h \in \mathbf{V}^h \end{aligned} \tag{5.2}$$

É possível ainda modelar contribuições de variáveis escalares nos métodos

Termo	Método ou Classe
$M(\mathbf{u}_h, \dot{\mathbf{u}}_h; \mathbf{v}_h)$	`mass_residual`
$F(\mathbf{u}_h; \mathbf{v}_h)$	`element_time_derivative`
$G(\mathbf{u}_h; \mathbf{v}_h)$	`element_constraint`
$\mathbf{u}_h(t) = \mathbf{g}(t)$	`DirichletBoundary`
$\sigma(\mathbf{u}_h, t) \cdot \mathbf{n} = h(t)$	`side_time_derivative` e `side_constraint`

Tabela 5.1: Correspondência entre termos e objetos do `libMesh`

`nonlocal_time_derivative` e `nonlocal_constraint`, porém estes não foram utilizados no presente trabalho. As formulações implementadas para os resíduos estão descritas no apêndice A.

A motivação lógica por detrás da separação em classes da tabela 5.1 é possibilitar o desenvolvimento de formulações físicas independentes do algorítimo de integração temporal utilizado. A integração temporal é delegada para classe `TimeSolver`, que realiza a formação dos vetores de resíduo, calcula a atual taxa de evolução do vetor solução, o vetor de incógnitas atualizado e provêm um método para marchar no tempo, a solução de fato do sistema é então delegada a uma terceira classe `DiffSolver` que é encarregada de realizar a montagem e solução do sistema não linear. A codificação do algorítimo BDF está descrita no apêndice B, a classe para solução não linear utilizando o `PETSc` já estava inicialmente codificada no `libMesh`, sendo que a contribuição desse trabalho foi a solução de falhas e a montagem correta para utilizar o PJFNK.

5.3 A biblioteca `PETSc`

Desenvolvido desde 1994, inicialmente no laboratório nacional Argonne (em colaboração com a universidade de Chicago) e atualmente com desenvolvedores em diversas instituições pelo mundo o `PETSc` é uma biblioteca de código-livre bem estabelecida na computação cientifica, sendo utilizado para simulações de alta fidelidade em aplicações como física de plasmas e segurança nuclear.

A biblioteca consiste de uma diversidade de componentes para simulações numéricas, desde estruturas de dados paralelas para vetores, matriz e malhas até métodos de marcha no tempo e solucionadores de sistemas de equações não-lineares.

No presente trabalho fazemos uso das facilidades para solução de sistemas não lineares do `PETSc`, materializadas no subframework `SNES`, sendo uma grande vantagem a possibilidade de compor diferentes estratégias e combinações de métodos a passando apenas argumentos na linha de comando diferentes, sem ser necessário recompilar ou modificar o programa. A definição do infinitesimal na formulação JFNK é dada pela constante PETSC_SQRT_MACHINE_EPSILON que para dupla

precisão tem por definição 1^{-7}, valor utilizado em todas as simulações desse trabalho.

No `SNES` são disponibilizadas diferentes potencialidades de configuração, como estudaremos um método semi-implícito foram utilizadas configurações no *line search* para limitar o passo não linear a cada iteração quando o sistema não apresentar convergência, tal algorítimo é chamado de *backtracking* e pode ser acionado utilizando-se do argumento `-snes_linesearch_type bt`.

Capítulo 6

Resultados numéricos

6.1 Caso de validação: problema de Kim-Moin

O problema de Kim-Moin é uma das raras soluções analítias para as equações incompressíveis de Navier-Stokes. A solução analítica para os campos de pressão e velocidade são dados pela Eq. 6.1 e definidas em $\Omega = [1, 0] \times [0, 1]$, sendo que para o presente trabalho os parâmetros de $a = 0.02$ e $\nu = 0.01$ foram escolhidos para permitir a comparação com os resultados de FÖRSTER *et al.* [19].

$$\begin{aligned} u_x(x, y, t) &= -cos(a\pi x)sin(a\pi y)e^{-2a^2\pi^2 t\nu} \\ u_y(x, y, t) &= sin(a\pi x)cos(a\pi y)e^{-2a^2\pi^2 t\nu} \\ p(x, y, t) &= -\frac{1}{4}(cos(2a\pi x) + cos(2a\pi y))e^{-4a^2\pi^2 t\nu} \end{aligned} \tag{6.1}$$

Os erros foram calculados segundo a Eq. 6.2, onde $||.||$ é a norma L^2. A tabela 6.1 apresenta os resultados ao final de 100 passos temporais de $0.01s$ para malhas de 32×32 elementos lagrangeano Q1 e 16×16 elementos lagrangeano Q2, foi incluído também o resultado adaptativo para 1024 (32^2) elementos Q1 utilizando de 4 passos adaptativos. Das diversas soluções de FÖRSTER *et al.* [19] foram utilizados na tabela 6.1 os melhores resultados para o erro na velocidade e seu erro correspondente em pressão. Em todos os casos para solução do sistema não-linear foi utilizado o método de newton inexado com backtracking, com uso do GMRES precondicionado com LU inexato para a solução do sistema linear em cada iteração. Foram utilizadas a tolerância relativa de 1×10^{-10} e a tolerância absoluta de 1×10^{-12} para o passo não linear e relativa de 1×10^{-12} para o passo linear. É importante notar que o presente trabalho utiliza uma aproximação numérica de diferenças finitas para o Jacobiano. Outra fonte de imprecisão é a diferença da interpolação do BDF e a aproximação numérica do Jacobiano quando se tratando dos termos dependentes do tempo.

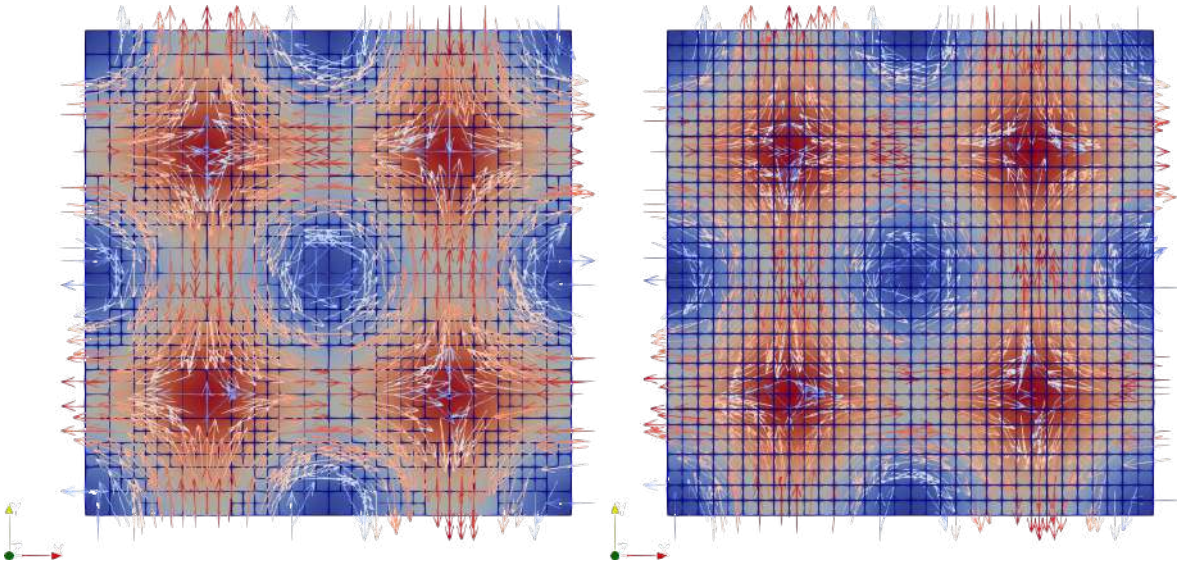

Figura 6.1: Problema de Kim-Moin, solução AMR e de malha fixa.

$$
\begin{aligned}
err_{\mathbf{u}} &:= ||\mathbf{u}^h - \mathbf{u}||e^{2a^2\pi^2 t\nu} \\
err_p &:= ||p^h - p||e^{4a^2\pi^2 t\nu}
\end{aligned}
\tag{6.2}
$$

	$err_{\mathbf{u}}$	err_p
Q1 SUPG FÖRSTER *et al.* [19]	0.004614	0.002460
Q1 BFD1 implícito	0.007276	0.009018
Q1 BFD1 implícito AMR	0.005439	0.007009
Q1 BFD1 semi-implícito	0.007276	0.009029
Q1 BFD2 implícito	0.008326	0.009018
Q1 BFD2 semi-implícito	0.008326	0.010488
Q2 SUPG FÖRSTER *et al.* [19]	0.002617	0.002544
Q2 BFD1 implícito	0.002514	0.003597
Q2 BFD1 semi-implícito	0.002512	0.003621
Q2 BFD2 implícito	0.003576	0.005072
Q2 BFD2 semi-implícito	0.003573	0.007755

Tabela 6.1: Erros em velocidade e pressão para caso de Kim-Moin.

Dos resultados reproduzidos na tabela 6.1 é possível notar que apesar das referidas fontes de erros todos os testes obtiveram convergência, nenhuma instabilidade foi notada quanto a seleção do passo temporal na formulação semi-implícita.

O problema em um grid 100×100 de elementos Q1 foi solucionado nos sistemas HPC Lobo Carneiro e Mont Blanc para avaliação do uso da plataforma Mont-Blanc com o `libMesh` e `PETSc`. O supercomputador Lobo Carneiro foi projetado com base nas tecnologias atualmente lideres no mercado sendo formado por nós de cálculo

contendo processadores Intel Xeon E5-2670 v3, membro da família de processadores de arquitetura x86-64, os nós são conectados através de rede InfiniBand FDR. O projeto Mont-Blanc (RAJOVIC *et al.* [20]) consiste de uma série de pequenos sistemas para testes, com a intenção de fazer uso de tecnologias emergentes da área de computação móvel para construir sistemas mais eficientes energeticamente, o sistema utilizado no presente trabalho (devido a requisitos de memória) é composto por nós contendo processadores Cavium Thunder X, Fig. 6.2, membro da família de processadores de arquitetura ARMv8.

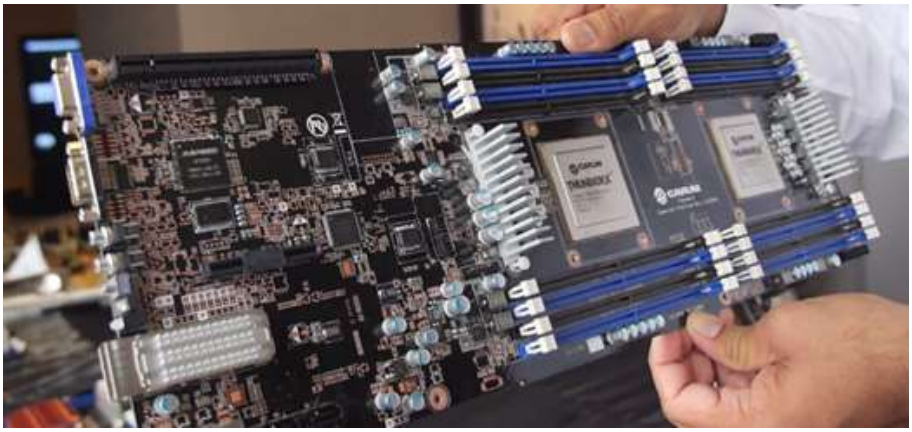

Figura 6.2: Nó de calculo Thunder X, protótipo Mont-Blanc.

Na discretização do tempo foram utilizados 1600 passos de $0.005s$, os resultados encontra-se dispostos na tabela 6.2 onde a escalabilidade é dada como percentual em relação ao desempenho perfeitamente linear, as configurações de solução são as mesmas descritas anteriormente. A relação do número de elementos por núcleo utilizado foi consideravelmente baixa, porém essa configuração nos permite comparar a performance de I/O dos sistemas, núcleos AMRv8 são de desempenho individual inferior e são geralmente recomendados em menor relação com o número de graus de liberdade do problema RAJOVIC *et al.* [20], sendo assim o sistema de I/O precisa ter uma elevada performance.

O fraco desempenho quanto a consumo energético do Mont-Blanc pode ser justificado pelo fato da medição ser realizada em um conjunto de quatro nós. Para minimizar os efeitos os nós foram sempre todos alocados mesmo que não realizando trabalho. Nota-se também uma grande queda na escalabilidade em 128 e 256 processos, cada nó de calculo possui dois processadores totalizando 96 núcleos por nó, sendo assim 128 e 256 processos representam 2 e 3 nós respectivamente. A nível de comparação no Lobo Carneiro as simulações foram realizadas em 2, 4, 8 e 16 nós respectivamente, e as medições de energia são individuais aos nós.

Nº de tarefas MPI	Energia consumida	Escalabilidade	Tempo de execução
Mont-Blanc	kWh	-	h
32	0.6103	Referência	1.151
64	0.4715	67.17%	0.857
128	0.6137	27.26%	1.056
256	1.8140	5.02%	2.867
Lobo Carneiro			
32	0.0532	Referência	0.128
64	0.0745	67.79%	0.094
128	0.1404	33.70%	0.095
256	0.4321	10.83%	0.148

Tabela 6.2: Consumo energético e escalabilidade forte no problema de Kim-Moin.

6.2 Caso referência: Cavidade 3D Re 12000

O problema da cavidade tridimensional representado pela Fig. 6.3 foi resolvido para $Re = 12000$, com o número de Reynolds definido como $Re = LU_0/\nu$. Este é um problema clássico estudado por diversos pesquisadores, no problema tridimensional complexas estruturas turbulentas se desenvolvem para casos com o número de Reynolds elevado. Os resultados aqui apresentados são comparados com os disponíveis em LINS *et al.* [8], na solução aqui obtida toda a face superior da cavidade possui velocidade $U_0 = 1m/s$ e as demais paredes tem condição de não escorregamento. Condições de contorno mais elaboradas para a face superior como a utilizada em LINS *et al.* [8] ajudam no tratamento da descontinuidade das condições de contorno, no entanto foi feita a opção pela configuração mais simples pois no presente trabalho não buscamos o desenvolvimento de um *benchmark* para o problema da cavidade, mas sim avaliar a robustez do solver desenvolvido.

e

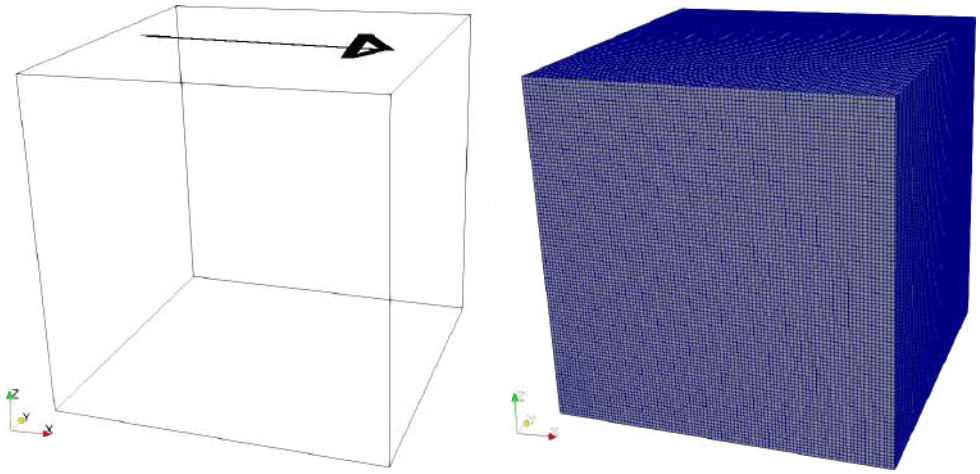

Figura 6.3: Cavidade tridimensional.

A malha representada na Fig. 6.3 é composta de 1 milhão ($100 \times 100 \times 100$) de hexaedros lineares em uma geometria cubica com $1m$ de lado. Para solução do sistema não-linear foi utilizado o método de Newton inexato com *backtracking*, para solução dos sistemas lineares foi utilizado o GMRES com precondicionador LU inexato. Foi utilizada uma tolerância relativa de 1×10^{-6} entre os passos não-lineares e um tolerância absoluta de 1×10^{-8}. O problema foi resolvido com passo temporal de $0.1s$ e 1000 passos, as velocidades médias são a média temporal em cada componente.

A Fig. 6.4 apresenta os valores em um corte normal a y no centro da cavidade, os resultados são comparados com os obtidos experimentalmente por PRASAD e KOSEFF [21] e para a solução de simulação numérica direta (DNS) de BOUFFANAIS *et al.* [22].

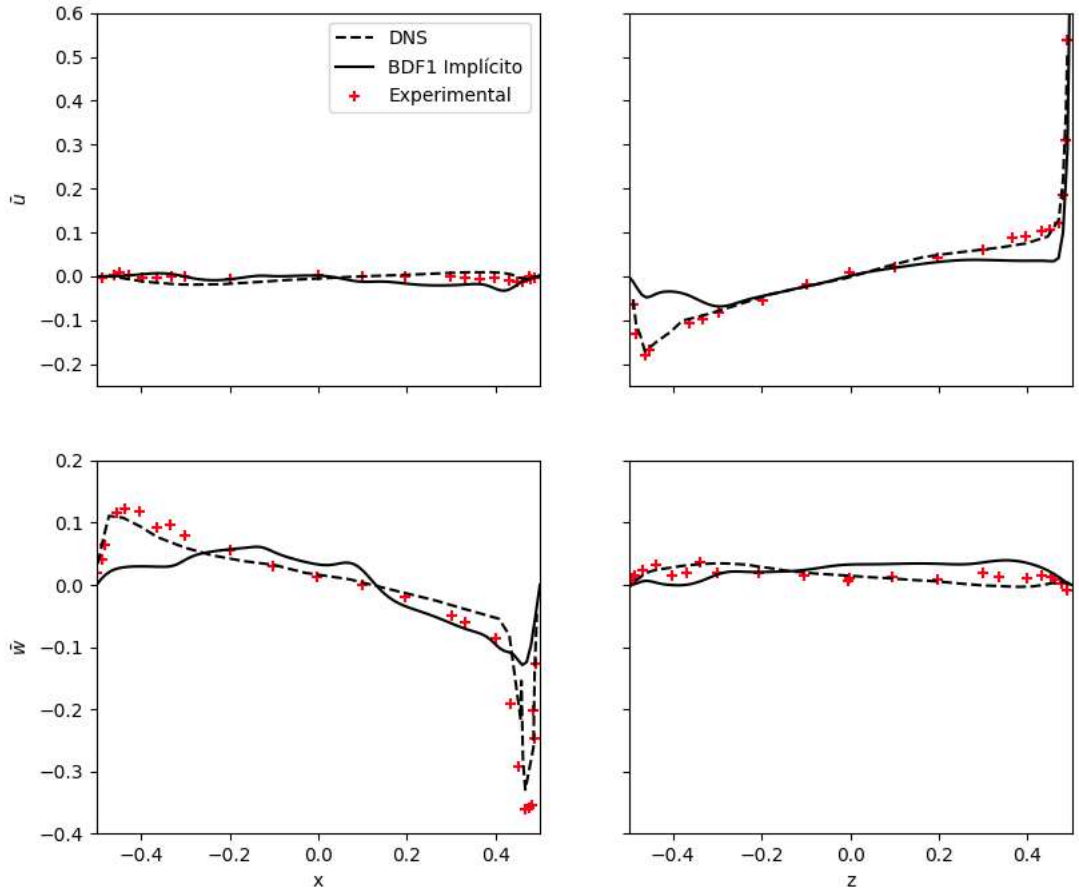

Figura 6.4: Comparação com *benchmarks*, cavidade $Re = 12000$.

A Fig. 6.5 apresenta as linhas de corrente na velocidade média, onde é possível observar estruturas turbulentas do problema.

Figura 6.5: Linhas de corrente, cavidade $Re = 12000$.

Capítulo 7

Conclusões e trabalhos futuros

7.1 Conclusões

Foram reproduzidas com sucesso as formulações implícita de BAZILEVS *et al.* [4] e semi-implícita de FORTI e DEDÈ [12] utilizando integrador BDF de primeira e segunda ordem solucionados pela metodologia JFNK. O código foi todo desenvolvido de forma transparente e documentado de forma a facilitar a transferência tecnológica. Contribuindo assim para o entendimento e documentação do framework `FEMSystem` para trabalhos futuros. Durante o desenvolvimento do presente trabalho o fato da biblioteca **libMesh** ser de código aberto foi bastante benéfico, pois a interação com os desenvolvedores permitiu a solução rápida de falhas afetando o desenvolvimento do trabalho.

A utilização do BDF2 não trouxe grandes ganhos em precisão frente ao BDF1 quando observados os resultados do problema de Kim-Moin e da cavidade, espera-se que a codificação do Jacobiano analítico (para utilização na fase de pré-condicionamento) deve melhor o desempenho quanto a precisão numérica da formulação de segunda ordem.

Nos teste realizados não foram encontrados problemas de instabilidade quanto a seleção do passo temporal para a formulação semi-implícita, apesar disso recomenda-se cautela pois os passos temporais utilizados no presente trabalho foram em geral conservativos. Ambas as formulações implícita e semi-implícita demostraram perfil comparável de precisão numérica.

Por fim o teste realizado no protótipo Thunder X do projeto Mont-Blanc mostra que as plataformas ARM da atual geração ainda não são competitivas para aplicações HPC do MEF quando utilizando a biblioteca **PETSc**, é possível que soluções desenvolvidas especificamente para a plataforma sejam mais competitivas porém o custo de reimplementar toda a infraestrutura necessária não deve ser subestimado. Para um mesmo problema posto na plataforma Thunder X é exigido um numero uma

ordem de magnitude maior de núcleos para obter tempos de execução comparáveis, mas as características de escalabilidade forte do sistema não são satisfatórias quando comparadas ao estado da arte de processadores x86-64 utilizando os algorítimos do presente trabalho. Os benefícios esperados com consumo energético não foram observados na prática devido a dificuldade de monitor o consumo energético com precisão e de forma comparável entre os sistemas. Como consequência do presente trabalho atualmente realizamos estudos adicionais de escalabilidade, quantificando outros parâmetros e se algorítimos como pre-condicionadores específicos podem ajudar no perfil de escalabilidade. Apesar disso já é largamente divulgado o projeto de uma nova versão da arquitetura, Thunder X2 (CAVIUM [23]), que busca tornar a plataforma mais atrativas para aplicações de HPC melhorando o desempenho de acesso a memória e rede.

7.2 Trabalhos futuros

Para continuidade e extensão deste trabalho sugere-se:

1. Desenvolver e implementar método para seleção automática do passo temporal que garanta estabilidade da formulação semi-implícita.

2. Adicionar as formulações analíticas do Jacobinano para formação do precondicionador.

3. Implementar e comparar o desempenho dos parâmetros de estabilização τ alternativos, como os propostos em HSU *et al.* [15].

4. Adicionar condições de contorno específicas e aplicar a casos de engenharia.

5. Estender para o tratamento de interação fluido-estrutura (adicionando a possibilidade de aplicar o solver a problemas de aeroelasticidade).

6. Estender para o tratamento de fluídos levemente compressíveis (adicionando a possibilidade de aplicar o solver a problemas acústicos).

Referências Bibliográficas

[1] DUISBURG-ESSEN, U. "Fluiddynamik Galerie". 2016. Disponível em: <https://www.uni-due.de/ivg/fluiddynamik/de/galerie.php>.

[2] HUGHES, J. T., MAZZEI, L., JANSEN, E. K. "Large Eddy Simulation and the variational multiscale method", *Computing and Visualization in Science*, v. 3, n. 1, pp. 47–59, 2000. ISSN: 1433-0369. doi: 10.1007/s007910050051. Disponível em: <http://dx.doi.org/10.1007/s007910050051>.

[3] BRENNER, S. C., SCOTT, L. R. *The mathematical theory of finite element methods*. Texts in applied mathematics 15. Springer-Verlag New York, 2008. ISBN: 0387759336,978-0-387-75933-3.

[4] BAZILEVS, Y., CALO, V., COTTRELL, J., et al. "Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows", *Computer Methods in Applied Mechanics and Engineering*, v. 197, n. 1–4, pp. 173 – 201, 2007. ISSN: 0045-7825. doi: http://dx.doi.org/10.1016/j.cma.2007.07.016. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0045782507003027>.

[5] KIRK, B. S., PETERSON, J. W., STOGNER, R. H., et al. "libMesh: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations", *Engineering with Computers*, v. 22, n. 3–4, pp. 237–254, 2006. http://dx.doi.org/10.1007/s00366-006-0049-3.

[6] DONEA, J., HUERTA, A. *Finite Element Methods for Flow Problems*. Wiley, 2003. ISBN: 0471496669,9780471496663.

[7] BALAY, S., ABHYANKAR, S., ADAMS, M. F., et al. "PETSc Web page". 2016. Disponível em: <http://www.mcs.anl.gov/petsc>.

[8] LINS, E. F., ELIAS, R. N., GUERRA, G. M., et al. "Edge-based finite element implementation of the residual-based variational multiscale method", *International Journal for Numerical Methods in Fluids*, v. 61, n. 1, pp. 1–22, 2009. ISSN: 1097-0363. doi: 10.1002/fld.1941. Disponível em: <http://dx.doi.org/10.1002/fld.1941>.

[9] GUERRA, G. M., ZIO, S., CAMATA, J. J., et al. "Numerical simulation of particle-laden flows by the residual-based variational multiscale method", *International Journal for Numerical Methods in Fluids*, v. 73, n. 8, pp. 729–749, 2013. ISSN: 1097-0363. doi: 10.1002/fld.3820. Disponível em: <http://dx.doi.org/10.1002/fld.3820>.

[10] AHMED, N., CHACÓN REBOLLO, T., JOHN, V., et al. "A Review of Variational Multiscale Methods for the Simulation of Turbulent Incompressible Flows", *Archives of Computational Methods in Engineering*, pp. 115–164, 2015. ISSN: 1134-3060. doi: 10.1007/s11831-015-9161-0. Disponível em: <http://link.springer.com/10.1007/s11831-015-9161-0>.

[11] RASTHOFER, U., GRAVEMEIER, V. *Recent Developments in Variational Multiscale Methods for Large-Eddy Simulation of Turbulent Flow*, v. 0. Springer Netherlands, 2017. ISBN: 1183101792094. doi: 10.1007/s11831-017-9209-4.

[12] FORTI, D., DEDÈ, L. "Semi-implicit BDF time discretization of the Navier–Stokes equations with VMS-LES modeling in a High Performance Computing framework", *Computers & Fluids*, v. 117, pp. 168 – 182, 2015. ISSN: 0045-7930. doi: http://dx.doi.org/10.1016/j.compfluid.2015.05.011. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0045793015001668>.

[13] MAY, P., ARROUVEL, C., REVOL, M., et al. "Detection of hemodynamic turbulence in experimental stenosis: an in vivo study in the rat carotid artery", *Journal of vascular research*, v. 39, n. 1, pp. 21—29, 2002. ISSN: 1018-1172. doi: 10.1159/000048990. Disponível em: <http://dx.doi.org/10.1159/000048990>.

[14] BENZI, R., FRISCH, U. "Turbulence", *Scholarpedia*, v. 5, n. 3, pp. 3439, 2010. revision 137205.

[15] HSU, M.-C., BAZILEVS, Y., CALO, V., et al. "Improving stability of stabilized and multiscale formulations in flow simulations at small time steps", *Computer Methods in Applied Mechanics and Engineering*, v. 199, n. 13–16, pp. 828 – 840, 2010. ISSN: 0045-7825. doi: http://dx.doi.org/10.1016/j.cma.2009.06.019. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0045782509002254>. Turbulence Modeling for Large Eddy Simulations.

[16] KNOLL, D., KEYES, D. "Jacobian-free Newton–Krylov methods: a survey of approaches and applications", *Journal of Computational Physics*, v. 193,

n. 2, pp. 357 – 397, 2004. ISSN: 0021-9991. doi: http://dx.doi.org/10.1016/j.jcp.2003.08.010. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0021999103004340>.

[17] GASTON, D., NEWMAN, C., HANSEN, G. "MOOSE Framework Preconditioners". 2016. Disponível em: <http://mooseframework.org/wiki/MooseSystems/Preconditioners/>.

[18] BAUMAN, P. T., STOGNER, R. H. "GRINS: A Multiphysics Framework Based on the libMesh Finite Element Library", *SIAM Journal on Scientific Computing*, 2016.

[19] FÖRSTER, C., WALL, W. A., RAMM, E. "Stabilized finite element formulation for incompressible flow on distorted meshes", *International Journal for Numerical Methods in Fluids*, v. 60, n. 10, pp. 1103–1126, 2009. ISSN: 1097-0363. doi: 10.1002/fld.1923. Disponível em: <http://dx.doi.org/10.1002/fld.1923>.

[20] RAJOVIC, N., RICO, A., MANTOVANI, F., et al. "The Mont-blanc Prototype: An Alternative Approach for HPC Systems". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '16, pp. 38:1–38:12, Piscataway, NJ, USA, 2016. IEEE Press. ISBN: 978-1-4673-8815-3. Disponível em: <http://dl.acm.org/citation.cfm?id=3014904.3014955>.

[21] PRASAD, A. K., KOSEFF, J. R. "Reynolds number and end-wall effects on a lid-driven cavity flow", *Physics of Fluids A: Fluid Dynamics*, v. 1, n. 2, pp. 208–218, 1989. doi: 10.1063/1.857491. Disponível em: <http://dx.doi.org/10.1063/1.857491>.

[22] BOUFFANAIS, R., DEVILLE, M. O., FISCHER, P. F., et al. "Large-Eddy Simulation of the Lid-Driven Cubic Cavity Flow by the Spectral Element Method", *Journal of Scientific Computing*, v. 27, n. 1, pp. 151, 2006. ISSN: 1573-7691. doi: 10.1007/s10915-005-9039-7. Disponível em: <http://dx.doi.org/10.1007/s10915-005-9039-7>.

[23] CAVIUM. "Thunder X2". 2017. Disponível em: <http://www.cavium.com/ThunderX2_ARM_Processors.html>.

Apêndice A

Implementações FEMSystem

As formulações descritas a seguir possuem em comum as seguinte definições:

$$\mathbf{G} = G_{ij} = \sum_{k=1}^{3} \frac{\partial \xi_k}{\partial x_i} \frac{\partial \xi_k}{\partial x_j} \tag{A.1}$$

$$\mathbf{g} = g_i = \sum_{j=1}^{3} \frac{\partial \xi_j}{\partial x_i} \tag{A.2}$$

$$r_C = \nabla \cdot \mathbf{u} \tag{A.3}$$

A construção dos componentes das equações, como velocidades e resíduos, foi feita de forma que a codificação se aproxima-se ao máximo da notação matemática da formulação, facilitando a leitura e entendimento da parte fundamental do solver. Sendo assim quando representada em código a multiplicação de vetores a operação realizada implicitamente é o produto interno entre os termos e a multiplicação de vetores com escalares é realizada termo a termo. As classes redefinidas estão listadas na tabela A.1.

Termo	Método ou Classe
$M(\mathbf{u}_h, \dot{\mathbf{u}}_h; \mathbf{v}_h)$	`mass_residual`
$F(\mathbf{u}_h; \mathbf{v}_h)$	`element_time_derivative`
$G(\mathbf{u}_h; \mathbf{v}_h)$	`element_constraint`

Tabela A.1: Métodos codificados no **FEMSystem**

Os vetores de resíduo dos termos acima na componente i e suas correspondente codificação na competente x no laço de integração dos pontos de quadratura são explicitados a seguir para as duas formulações implementadas.

A.1 Formulação implícita

$$
\begin{aligned}
M_i = & \dot{u_i}\phi_a \\
& +\tau_M \dot{u_i}(\mathbf{u}\cdot\nabla\phi_a) \\
& +\tau_M \nabla q\cdot\dot{\mathbf{u}} \\
& +\tau_M u_i \nabla\phi_a\cdot\dot{\mathbf{u}} \\
& -\tau_M^2\nabla\phi_a\cdot(\dot{u_i}\mathbf{r}_M + r_{s,i}\dot{\mathbf{u}})
\end{aligned}
\tag{A.4}
$$

```
Mu(a) -= JxW[qp] * (
   dudt * phia
  +tauM * dudt * (U * Dphi)
  +tauM * u * (Dphi * dUdt)
  -tauM*tauM * Dphi * (dudt*rM + rS(0)*dUdt)
);
```

$$
\begin{aligned}
F_i = & u_i(\mathbf{u}\cdot\nabla\phi_a) \\
& +\frac{p}{\rho}\nabla_i\phi_a \\
& -\nu\nabla\phi_a\cdot(\nabla u_i + \frac{\partial\mathbf{u}}{\partial x_i}) \\
& -\tau_M r_{s,i}(\mathbf{u}\cdot\nabla\phi_a) \\
& -\tau_M r_s\cdot\nabla q \\
& -\tau_C r_C\nabla_i\phi_a \\
& -\tau_M u_i(\mathbf{r}_s\cdot\nabla\phi_a) \\
& +\tau_M^2 r_{s,i}(\mathbf{r}_s\cdot\nabla\phi_a) \\
& +\phi_a f_i
\end{aligned}
\tag{A.5}
$$

```
Fu(a) += JxW[qp] * (
   ub * (U * Dphi)
  +p * Dphi(0)
  -nu * Dphi * (grad_u + U_x)
  -tauM * rS(0) * (Ub * Dphi)
  -tauC * rC * Dphi(0)
  -tauM * ub * (rS * Dphi)
  +tauM*tauM * rM(0) * (rS * Dphi)
  +phia * f(0)
);
```

$$
G = \psi(\nabla\cdot\mathbf{u}) \tag{A.6}
$$

```
Fp(i) -= JxW[qp]*( psi[i][qp] * (grad_u(0) + grad_v(1)));
if (dim == 3)
    Fp(i) -= JxW[qp]*( psi[i][qp] * grad_w(2));
```

sendo:

$$\mathbf{r}_M = \dot{\mathbf{u}} + \mathbf{r}_s \tag{A.7}$$

$$\mathbf{r}_s = \mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{\rho}\nabla p - \nu \Delta \mathbf{u} - \mathbf{F}, \tag{A.8}$$

$$\tau_M = (\frac{4}{\Delta t^2} + \mathbf{u} \cdot \mathbf{G}\mathbf{u} + C_l \nu^2 \mathbf{G} : \mathbf{G})^{-1/2}, \tag{A.9}$$

$$\tau_C = (\tau_M \mathbf{g} \cdot \mathbf{g})^{-1}, \tag{A.10}$$

A.2 Formulação semi-implícita

$$\begin{aligned} M_i = & \dot{u_i}\phi_a \\ & + \tau_M \dot{u_i}(\mathbf{u}_\beta \cdot \nabla \phi_a) \\ & + \tau_M \nabla q \cdot \dot{\mathbf{u}} \\ & + \tau_M u_{\beta i} \nabla \phi_a \cdot \dot{\mathbf{u}} \\ & - \tau_M^2 (\nabla \phi_a \cdot \mathbf{r}_s)\dot{\mathbf{u}}_i \\ & + \frac{\tau_M^2}{\Delta t}(\nabla \phi_a \cdot \mathbf{u}_{nBDF})\dot{\mathbf{u}}_i \\ & - \frac{\alpha \tau_M^2}{\Delta t}(\nabla \phi_a \cdot \mathbf{u}) r_{Mi} \\ & + \frac{\tau_M^2}{\Delta t}(\nabla \phi_a \cdot \mathbf{u}_{nBDF}) r_{si} \end{aligned} \tag{A.11}$$

```
Mu(a) -= JxW[qp] * (
    dudt * phia
   +tauM * dudt * (U_b * Dphi)
   +tauM * u_b * (Dphi * dUdt)
   -tauM*tauM * (Dphi * rS) * dudt
   +(tauM*tauM/dt) * (Dphi * U_bdf) * dudt
   -(alpha*tauM*tauM/dt) * (Dphi * U) * rM(0)
   +(tauM*tauM/dt) * (Dphi * U_bdf) * rS(0)
);
```

$$
\begin{aligned}
F_i = u_{\beta i}(\mathbf{u}\cdot\nabla\phi_a) \\
+\frac{p}{\rho}\nabla_i\phi_a \\
-\nu\nabla\phi_a\cdot(\nabla u_i + \frac{\partial \mathbf{u}}{\partial x_i}) \\
-\tau_M r_{s,i}(\mathbf{u}_\beta\cdot\nabla\phi_a) - \tau_M r_s\cdot\nabla q \\
-\tau_C r_C\nabla_i\phi_a \\
-\tau_M u_{\beta i}(\mathbf{r}_s\cdot\nabla\phi_a) \\
+\tau_M^2 r_{Mi}(\mathbf{r}_s\cdot\nabla\phi_a) \\
+\phi_a f_i
\end{aligned}
\tag{A.12}
$$

```
Fu(a) += JxW[qp] * (
   u_b * (U * Dphi)
  +p * Dphi(0)
  -nu * Dphi * (grad_u + U_x)
  -tauM * rS(0) * (U_b * Dphi)
  -tauC * rC * Dphi(0)
  -tauM * u_b * (rS * Dphi)
  +tauM*tauM * rM(0) * (rS * Dphi)
  +phia * f(0)
);
```

$$
G = \psi(\nabla\cdot\mathbf{u}) \tag{A.13}
$$

```
Fp(i) -= JxW[qp]*( psi[i][qp] * (grad_u(0) + grad_v(1)));
if (dim == 3)
    Fp(i) -= JxW[qp]*( psi[i][qp] * grad_w(2));
```

sendo:

$$
\psi_{nBDF,\beta} = \begin{cases} \psi_n & n\geq 0, \quad \text{para} \quad \beta = 1 \\ 2\psi_n - \frac{1}{2}\psi_{n-1} & n\geq 1, \quad \text{para} \quad \beta = 2 \end{cases} \tag{A.14}
$$

$$
\alpha_\beta = \begin{cases} 1, & \text{para} \quad \beta = 1 \\ 3/2, & \text{para} \quad \beta = 2 \end{cases} \tag{A.15}
$$

$$
\psi_{n+1,\beta} = \begin{cases} \psi_n & n\geq 0, \quad \text{para} \quad \beta = 1 \\ 2\psi_n - \psi_{n-1} & n\geq 1, \quad \text{para} \quad \beta = 2 \end{cases} \tag{A.16}
$$

$$
\mathbf{r}_M = \frac{\alpha_\beta\mathbf{u} - \mathbf{u}_{nBDF,\beta}}{\Delta t} + \mathbf{u}_\beta\cdot\nabla\mathbf{u}_\beta + \frac{1}{\rho}\nabla p_\beta - \nu\Delta\mathbf{u}_\beta - \mathbf{f},, \tag{A.17}
$$

$$\mathbf{r}_s = \mathbf{u}_\beta \cdot \nabla \mathbf{u} + \frac{1}{\rho}\nabla p - \nu \Delta \mathbf{u} - \mathbf{f}, \tag{A.18}$$

$$\tau_M := (\frac{4}{\Delta t^2} + \mathbf{u}_\beta \cdot \mathbf{G}\mathbf{u}_\beta + C_l \nu^2 \mathbf{G} : \mathbf{G})^{-1/2}, \tag{A.19}$$

$$\tau_C := (\tau_M \mathbf{g} \cdot \mathbf{g})^{-1}, \tag{A.20}$$

Apêndice B

Implementações TimeSolver

O algorítimo BDF consiste na aproximação das derivativas temporais pela Eq. B.1

$$\frac{\partial \psi}{\partial t} \approx \frac{\alpha_\beta \psi_{n+1} - \psi_{nBDF,\beta}}{\Delta t} \tag{B.1}$$

onde

$$\psi_{nBDF,\beta} = \begin{cases} \psi_n & n \geq 0, \quad \text{para} \quad \beta = 1 \\ 2\psi_n - \frac{1}{2}\psi_{n-1} & n \geq 1, \quad \text{para} \quad \beta = 2 \\ 3\psi_n - \frac{3}{2}\psi_{n-1} + \frac{1}{3}\psi_{n-2} & n \geq 2, \quad \text{para} \quad \beta = 3 \end{cases} \tag{B.2}$$

e

$$\alpha_\beta = \begin{cases} 1, & \text{para} \quad \beta = 1 \\ 3/2, & \text{para} \quad \beta = 2 \\ 11/6, & \text{para} \quad \beta = 3 \end{cases} \tag{B.3}$$

Assim como na implementação do `FEMSystem` a codificação foi realizada de forma a tornar a implementação o mais próxima possível da formulação matemática, reproduzimos abaixo a parte central do algorítimo, como no apêndice A não reproduziremos a formação das estruturas de dados e implementação dos métodos acessórios.

```
  if ( beta == 1 || !(_system.time > _system.deltat))
    {
      // Local nonlinear solution at old timestep
      DenseVector<Number> old_elem_solution(n_dofs);
      for (unsigned int i=0; i != n_dofs; ++i)
        old_elem_solution(i) =
          old_nonlinear_solution(context.get_dof_indices()[i]);

      context.get_elem_solution_rate() = context.get_elem_solution();
      context.get_elem_solution_rate() -= old_elem_solution;
      context.elem_solution_rate_derivative = 1 / _system.deltat;
      context.get_elem_solution_rate() *=
        context.elem_solution_rate_derivative;

      bdf_time_derivative = 1.0;
    }
  else if (beta == 2)
    {
      // Local nonlinear solution at old timestep
      DenseVector<Number> old_elem_solution(n_dofs);
      for (unsigned int i=0; i != n_dofs; ++i)
        old_elem_solution(i) = 4.0/3.0 *
          old_nonlinear_solution(context.get_dof_indices()[i]);

      // Local nonlinear solution at older timestep
      DenseVector<Number> older_elem_solution(n_dofs);
      for (unsigned int i=0; i != n_dofs; ++i)
        older_elem_solution(i) = 1.0/3.0 *
          older_nonlinear_solution(context.get_dof_indices()[i]);

      context.get_elem_solution_rate() = context.get_elem_solution();
      context.get_elem_solution_rate() -= old_elem_solution;
      context.get_elem_solution_rate() += older_elem_solution;
      context.elem_solution_rate_derivative = 3.0 / (2.0 * _system.deltat);
      context.get_elem_solution_rate() *=
        context.elem_solution_rate_derivative;

      bdf_time_derivative = 1.5;
    }
```