Modelagem Computacional em Simulação Numérica

Computational Modelling in Numerical Simulation

Autores:
Marcelo Castro de Souza
Márcio Eduardo Régis Monteiro
Orientador:
Clovis Raimundo Maliska, Ph.D
Colaboradores:
Clovis Raimundo Maliska Jr.
Felipe Gallina
Márcio Alexandre Cano Miranda
Universidade Federal de Santa Catarina
Departamento de Engenharia Mecânica
SINMEC – Laboratório de Simulação Numérica em Mecânica dos Fluidos e Transferência de Calor
Campus Universitário Trindade
88040-900 – Florianópolis – SC – Brasil
maliska@sinmec.ufsc.br

Abstract

Heat Transfer and Fluid Mechanics problems are governed by equations with many parameters. However, the complexity of the Numerical Methods used to solve them do not need necessarily result in complexity of the computational source. Programming before planning is a common practice among who deals with software conception and development. The programmer wants, most of times, to see the results on the screen in the fastest possible way. This reflects on the results such as time of programming and source's usability by other people. This work shows how computational modelling, when done in a correct and conscious way, can facilitate the programmer's task and the reusability of the source. In this sense, the development of the new version of the software **Sinflow** (Convective Heat Transfer Simulator) by SINMEC is used as an example. It still holds the same objective of the first version – resolution of convective heat transfer problems – and the same capabilities – friendly interface and visualization, using for that modern resources of Object Oriented Programming in C++ and a powerful graphical library, COI-lib 2.0. The program can be used with both UNIX and WINDOWS operational systems. To keep the readability of the source and turn future modifications more easy, news structures and hierarchies of classes are being developed. The new model has a better internal concept, resulting in a faster comprehension of the scheme that the program uses to solve the problems.

Palavras-chave

Modelagem Computacional Simulação Numérica **Sinflow 2.0**

Introdução

É prática comum durante a construção de um programa computacional a pressa em se ver resultados. Sendo assim, pouca atenção é dada ao planejamento inicial pelo qual o código deve passar, definindo as tarefas específicas de cada módulo e o melhor modo de relacioná-los. O programador tende a trabalhar através do método da tentativa e erro, perdendo assim grande parte do seu tempo em intermináveis sessões de depuração. O objetivo deste trabalho é demonstrar como a modelagem computacional, quando realizada de modo correto e consciente, pode facilitar a tarefa da programação e o reaproveitamento de código. Para tanto, utilizar-se-á o modelo do **Sinflow 2.0**, programa de simulação numérica em Transferência de Calor Convectiva que vêm sendo desenvolvido no SINMEC. Ele presta-se à solução de problemas de Mecânica dos Fluidos e Transferência de Calor para escoamentos bidimensional ou tridimensional axissimétrico, laminar, com propriedades físicas constantes e incompressível. Utiliza para tanto o Método dos Volumes Finitos na discretização das

equações e coordenadas generalizadas em geometrias complexas simplesmente conexas. Seu desenvolvimento está baseado no paradigma da orientação ao objeto, utilizando C++ como linguagem de programação.

Modelagem Computacional

De acordo com [1], a qualidade de software está baseada em diversos fatores, agrupados em internos e externos. Como exemplos de fatores externos destacam-se:

- Corretude: delimitação do problema, definindo necessidades, requisitos e finalidades da aplicação;
- Robustez: criação de mecanismos que possam lidar com falhas de execução de maneira a finalizar de modo elegante a aplicação caso algo saia errado;
- Estendibilidade: facilidade de realizar adaptações ao código, mesmo quando do término do trabalho inicialmente proposto (complemento);
- Reusabilidade: possibilidade de uso de elementos já testados e comprovadamente funcionais através de seu agrupamento em, por exemplo, bibliotecas;
- Compatibilidade: facilidade de interação e integração do aplicativo com demais aplicativos já existentes e/ou sistemas operacionais.

Os fatores internos só podem ser reconhecidos pelos projetistas e programadores. São eles:

- Modularidade: as tarefas devem estar claras, bem delimitadas e definidas da maneira mais simples possível;
- Legibilidade: o código deve ser legível a outros programadores devendo, para tanto, ser escrito da forma mais simples possível e, sempre que possível, enriquecido com comentários;
- Documentação do Projeto: importante para validar os requisitos no momento de teste do programa.

Observando os fatores acima, estabelece-se o ciclo de vida para o desenvolvimento da aplicação, definindo-se um cronograma das atividades a serem desenvolvidas, de acordo com o fluxograma abaixo:

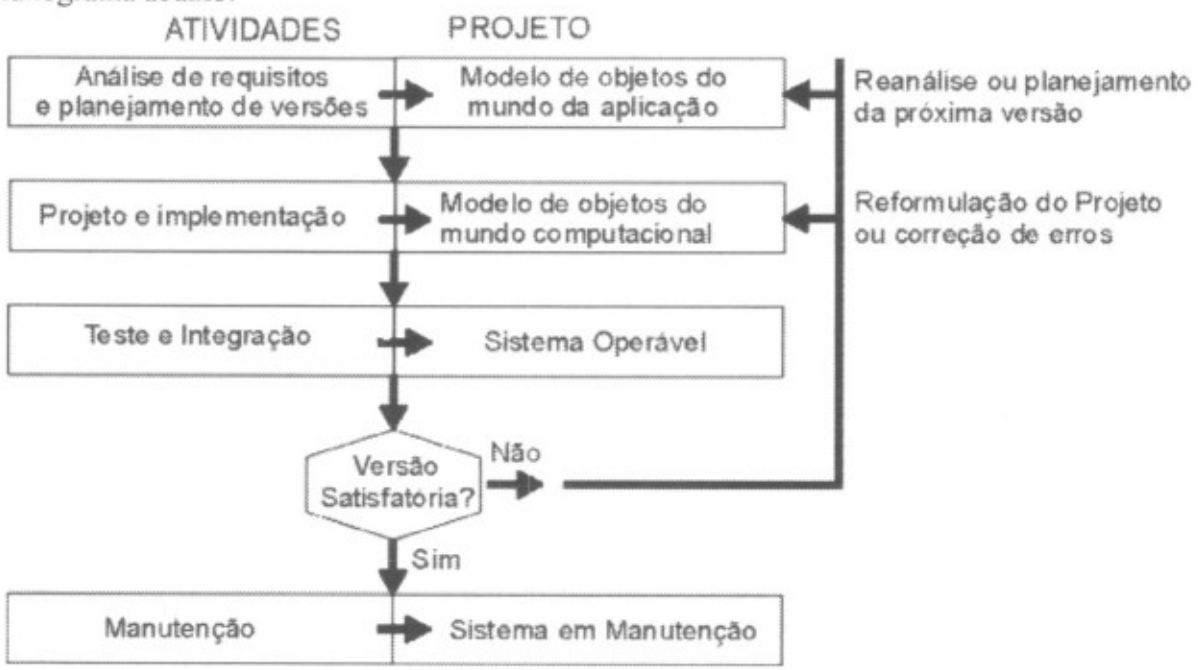

Figura 1: Ciclo de Vida

Modelagem Computacional & Simulação Numérica (Sinflow 2.0)

De acordo com o exposto anteriormente, a modelagem deve ser o mais racional possível e pensada em termos dos objetivos do programa. No caso do aplicativo do SINMEC, a resolução da equação [2]

$$\frac{\partial}{\partial t}(\rho\Phi)+\frac{\partial}{\partial t}(\rho u\Phi)+\frac{\partial}{\partial y}(\rho v\Phi)=\frac{\partial}{\partial x}\left(\Gamma^{\Phi}\frac{\partial\Phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(\Gamma^{\Phi}\frac{\partial\Phi}{\partial y}\right)+S^{\Phi} \qquad (1)$$

através do uso de coordenadas generalizadas e do Método dos Volumes Finitos é o objetivo principal para a obtenção de resultados em convecção com ou sem transferência de calor.

Dado o objetivo, verificam-se os requisitos para alcançá-lo. Neste caso, elementos indispensáveis são um domínio discretizado (e respectivas métricas de transformação, quando do uso de coordenadas generalizadas), condições de contorno e iniciais e valores de propriedades físicas e parâmetros de controle (como intervalo de tempo, no caso de um problema transiente).

Definida a finalidade da aplicação e dispondo-se dos requisitos, faz-se a modelagem dos objetos da aplicação, transmitindo-se as necessidades para realização das tarefas (etapas para resolução da equação) em termos de entes capazes de realizá-las corretamente. Neste ponto então são definidos os diversos módulos que constituirão a aplicação.

A partir desta fase, procede-se ao projeto e implementação dos objetos, traduzindo-os para o mundo computacional de forma adequada. Este resultado é exposto abaixo para o caso do **Sinflow 2.0**.

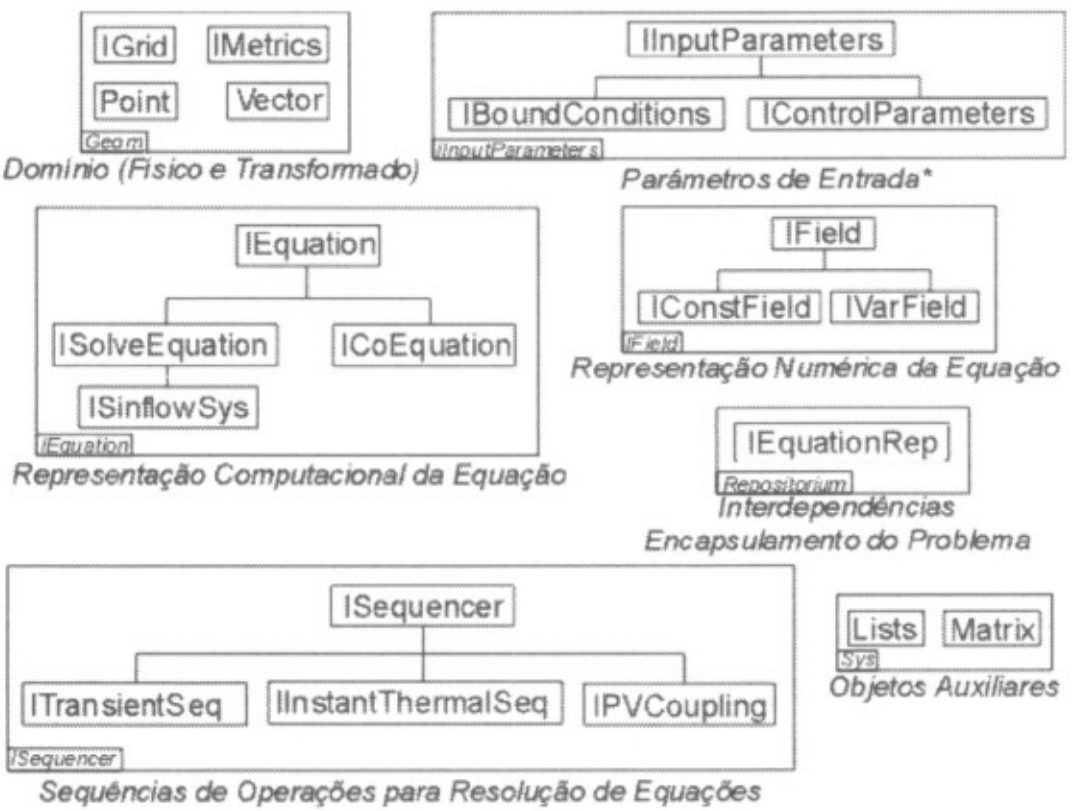

Figura 2: Módulos básicos e objetos principais do **Sinflow 2.0**

Os módulos do **Sinflow 2.0** possuem grande funcionalidade e são brevemente descritos a seguir. Observe-se que, embora integrados, os diversos módulos são independentes; modificações em um deles não devem acarretar modificações nos demais e o

interelacionamento é dado pelas tarefas realizadas por cada um, necessárias ao bom desempenho dos demais.

IEquation

Desde que o objetivo principal da aplicação é a resolução de uma equação, ela incorpora um módulo que representa computacionalmente as equações envolvidas nos diversos tipos de problemas (hidrodinâmico, difusivo, convectivo). No entanto, buscando uma abstração mais simples e legível, nesta aplicação uma equação pode representar qualquer variável, dependente ou independente, com um valor constante ou não dentro do domínio. Por exemplo,

$$\rho = cte. \tag{2}$$

e

$$T = T(\alpha, \xi, \eta, ...) \tag{3}$$

são equações dentro do programa tratadas, teoricamente, de maneira idêntica. Este tipo de abstração permite a implementação de qualquer tipo de equação, por mais complexa que ela seja. O módulo IEquation, entretanto, não contém os valores numéricos das equações (valores discretos no domínio). Estes estão presentes em objetos do tipo IField, que estão associados às equações. No caso de uma equação como (3), dependente de outras variáveis, uma lista encadeada contém as suas dependências. Desse modo pode-se, por exemplo, calcular o valor pontual de uma variável dependente (como, por exemplo, a difusividade térmica) acessando-se os campos das variáveis independentes que a compõe.

Dentro de sua abstração básica, o módulo IEquation contém como derivados ISolveEquation (equação que apresenta um sistema linear a ser resolvido) e ICoEquation (utilizada, por exemplo, na correção de um campo de velocidades, quando um IEquation recebe outro IEquation). A partir desta classe, as especializações são específicas para o **Sinflow 2.0**, dividindo-se entre IThermalEqSys e IHydroEqSys e seus respectivos derivados.

Repositorium

Como o nome sugere, funciona como um repositório de equações. Todas as equações envolvidas no problema, dependentes ou independentes, variáveis (no tempo e/ou no espaço) ou não, encontram-se neste objeto. Desse modo o problema fica encapsulado, e tornar-se possível até mesmo a resolução de mais de um problema ao mesmo tempo. O Repositório contém tudo relativo ao problema, através de duas listas encadeadas: uma de objetos do tipo IEquation e outra de objetos do tipo ISolveEquation. Assim, se é necessário acessar o campo de uma equação qualquer do problema, é possível fazê-lo através do repositório. Ao mesmo tempo, é possível verificar se os campos contém valores, e se esses valores estão corretos.

IField

Responsável por armazenar os valores discretos da variável no domínio; uma equação, por si só, não tem esta capacidade por tratar-se de uma abstração. No caso de equações dependentes, a presença deste objeto permite diversas combinações. Por exemplo, a propriedade fisica difusividade térmica, α, é dependente das propriedades fisicas k, ρ e c_p em cada ponto do domínio. Assim, pode ser calculada através da avaliação pontual dos campos de suas variáveis independentes, feita através do Repositorium, que fuciona como elo de ligação entre as equações.

Geom

É o módulo responsável pela discretização do domínio, através dos objetos IGrid (criação da malha a partir de uma matriz de pontos) e IMetrics (cálculo das métricas de transformação do domínio fisico para o computacional por coordenadas generalizadas).

IInputParameters

Sendo um módulo ainda em estudo, tem como função básica a captação e armazenamento das condições de contorno e iniciais, bem como de parâmetros de controle (tal como intervalo de tempo para cálculo de regime transiente).

ISequencer

Este módulo é responsável por ordenar e iniciar a resolução das equações do problema. Objetos do tipo básico ISequencer (virtual puro, ou seja, não pode ser instanciado) possuem dois modos de serem criados (construtoras): um deles possui como parâmetro um IEquationRep, que inicia uma lista encadeada de sequências armazenadas neste repositório. O outro possui como parâmetro outro objeto, derivado de ISequencer. Isto permite criar uma sequência dentro de outra, adicionando a primeira como mais um ítem da lista destes objetos. Este artifico é usado, por exemplo, em problemas transientes onde há uma sequência de resolução para cada intervalo de tempo e uma maior, que engloba todo o problema. É importante ressaltar que objetos derivados têm as mesmas funcionalidades do pai. Para utilizar-se um ISequencer, seus objetos derivados devem conter no mínimo uma equação a ser resolvida. e/ou outra sequência a ser iniciada.

Dentro de sua hierarquia, ISequencer possui 3 objetos principais: ITransientSeq, IInstantSeq e ICorrectorSeq. ITransientSeq é a classe reponsável por gerenciar os passos de uma simulação transiente, iniciar os campos, resolver o problema em um instante de tempo, verificar a convergência , atualizar os campos do instante anterior e seguir até a convergência final, no regime permanente. Estas funcionalidades são especializadas conforme o tipo de problema. A função responsável por resolver o problema em um instante chama um objeto derivado de um IInstantSeq, que pode ser a sequência para resolução da equação da temperatura no caso térmico, do acoplamento pressão-velocidade no caso hidrodinâmico e os dois sucessivamente, no caso convectivo. ICorrectorSeq é o objeto que deriva uma sequência de correção da velocidade – IVelocityCor – e uma de correção da pressão – IPressureCor. Ambos são utilizados nos cálculos do acoplamento pressão-velocidade.

O acoplamento pressão-velocidade ainda é subdividido em ISimple, IPrime, ISimplec e ISimpler, cada qual com seu modo de tratamento.

Sys

Contém objetos auxiliares, fundamentais à aplicação, tais como Matrix e Lists (listas encadeadas).

Definidos os módulos e seus principais componentes, procede-se ao processo de implementação, que deve ser acompanhado de perto pelo processo de teste e integração já exposto acima. O **Sinflow 2.0** já tem os módulos Geom, IField, Repositorium e Sys completamente implementados, e IEquation e ISequencer parcialmente implementados. Testes já foram realizados na resolução de problemas puramente difusivos e os resultados são idênticos aos observados no Heat Transfer v1.1 e no **Sinflow** v1.0.

Conclusão

A concepção de um programa computacional não é tarefa fácil. A dificuldade maior reside em traduzir para o mundo do computador os problemas que se quer resolver,

obedecendo certos critérios e mantendo o trabalho legível. Encontrar a abstração mais simples nem sempre é rápido e experiência e maturidade são fatores de grande importância neste tipo de atividade. Atualmente, o **Sinflow 2.0** encontra-se na fase de projeto e implementação, acompanhada de perto pela fase de teste e integração. Por tratar-se de um código extenso, testes devem ser realizados para validação de novas implementações e/ou modificações de implementações já realizadas. O trabalho terá continuidade no sentido de finalizar a nova versão do programa, bem como de disponibilizar uma biblioteca de métodos numéricos para resolução de problemas de Transferência de Calor Convectiva através do Método dos Volume Finitos utilizando Coordenadas Generalizadas.

Bibliografia

ESSS – Engineering Simulation and Scientific Software, C++ Pocket Book – Programação de Sistemas, pp.16-34, Florianópolis, 1996.

MALISKA, C. R. Transferência de Calor e Mecânica dos Fluidos Computacional, Livros Técnicos e Científicos Editora, Rio de Janeiro, 1995.

SINMEC, Manual teórioco do Software Sinflow v.1.0, Florianópolis, 1997.