UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA CIVIL, ARQUITETURA E URBANISMO
DEPARTAMENTO DE RECURSOS HÍDRICOS

DESENVOLVIMENTO DE UM PROGRAMA ORIENTADO POR OBJETOS PARA ANÁLISE DE ESCOAMENTO EM CONDUTOS FORÇADOS

Autor: **Edwin Antonio Aranda Saldaña**
Orientador: **Prof. Dr. Edevar Luvizotto Junior**

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA CIVIL, ARQUITETURA E URBANISMO
DEPARTAMENTO DE RECURSOS HÍDRICOS

DESENVOLVIMENTO DE UM PROGRAMA ORIENTADO POR OBJETOS PARA ANÁLISE DE ESCOAMENTO EM CONDUTOS FORÇADOS

Autor: **Edwin Antonio Aranda Saldaña**

Orientador: **Prof. Dr. Edevar Luvizotto Junior**

Dissertação de Mestrado apresentado à Comissão de Pós-graduação da Faculdade de Engenharia Civil, Arquitetura e Urbanismo da Universidade Estadual de Campinas, como requisito para obtenção do título de Mestre em Engenharia Civil.

Campinas, Fevereiro de 2006.

SP - Brasil

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Aranda Saldaña, Edwin Antonio
Ar14d Desenvolvimento de um programa orientado por objetos para análise de escoamento em condutos forçados / Edwin Antonio Aranda Saldaña.--Campinas, SP: [s.n.], 2006.

Orientador: Edevar Luvizotto Junior
Dissertação (Mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Civil, Arquitetura e Urbanismo.

1. Modelos matemáticos. 2. Engenharia hidráulica. 3. Simulação por computador. 4. Programação orientada a objetos (Computação). 5. Transitórios hidráulicos. I. Luvizotto Junior, Edevar. II. Universidade Estadual de Campinas. Faculdade de Engenharia Civil, Arquitetura e Urbanismo. III. Título.

Titulo em Inglês: Development of a computational program based on the object oriented programming for the analysis of flow in pipe systems
Palavras-chave em Inglês: Method of characteristics, Inert dynamic model, Object oriented programming, Hydraulic systems
Área de concentração: Recursos Hídricos
Titulação: Mestre em Engenharia Civil
Banca examinadora: Paulo Vatavuk e Podalyro Amaral deSouza
Data da defesa: 24/02/2006

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA CIVIL, ARQUITETURA E URBANISMO

DESENVOLVIMENTO DE UM PROGRAMA ORIENTADO POR OBJETOS PARA ANÁLISE DE ESCOAMENTO EM CONDUTOS FORÇADOS

EDWIN ANTONIO ARANDA SALDAÑA

Dissertação de Mestrado aprovada pela Banca Examinadora, constituída por:

Prof. Dr. EDEVAR LUVIZOTTO JUNIOR
Presidente e Orientador
Universidade Estadual de Campinas/UNICAMP

Prof. Dr. PAULO VATAVUK
Universidade Estadual de Campinas/UNICAMP

Prof. Dr. PODALYRO AMARAL DE SOUZA
Escola Politécnica da USP/EPUSP

Campinas, 24 de Fevereiro de 2006.

DEDICATÓRIA

Aos meus pais, Antonio e Olga Teresa,
Aos meus irmãos, José Miguel, Marleny Doris, Patrícia.
Ao meu cunhado Rodrigo Marins P. Siloto

Agradecimentos

Meus agradecimentos a todos aqueles que, direta ou indiretamente, contribuíram para a realização deste trabalho e, em especial:

Ao Prof.º Edevar Luvizotto Junior, pela valiosa orientação, pelas discussões e direcionamento eficaz, que contribuíram na minha formação como pesquisador, e pela grande amizade durante todos estes anos.

Ao Prof.º Paulo Vatavuk, pela amizade e discussões constantes, pela disposição durante o andamento da pesquisa.

Aos amigos da UNICAMP: Bruno Preto, Carlos Andrade, Edna Chiarelli, Fabrício Alves, Francesco Felice, Elias Antonio Nicolas, Fernando Coelho, Ítalo Montalvão, Desireé Ramello, Jeferson Cassiano, Luiz Fernando Karps Pasquotto, Luis Fernando Cabeça, Luis Pedro, Luiz Fernando Resende (menino), Joaquim Marins, Reinaldo Oliveira, Paula Solleira, Rogério Almeida, Roger Larico, Rolando Quispe, Ulises Bobadilla, Viviane Aguirre, Wilvert Rios.

A família Marins Peixoto Siloto pelo apoio constante, incondicional durante minha permanência no Brasil.

A família Bogorell Venâncio pelo carinho e incentivo constante.

A Faculdade de Engenharia Civil pelo aceite no curso de mestrado.

Aos amigos,

A DEUS ... pela força sempre presente.

Muitíssimo Obrigado

Lista de Símbolos

A	área da seção transversal do tubo.
A'	variável associada a equação do ENO não-tubo.
a	celeridade.
a_0	coeficiente da equação ajustada da curva carga x vazão da bomba
B	coeficiente do método das característica.
B_E	coeficiente associado a equação do ENO.
B_N	coeficiente associado a equação do NÓ.
B_A, B_B, B_E, B_D	coeficientes associados a equação do NÓ.
B_I e B_{II}	representação genérica das constantes B_A, B_B, B_E e B_D.
B'	variável associada a equação do ENO não-tubo.
b_0	coeficiente da equação ajustada da curva carga x vazão da bomba
C	código de identificação do ENO.
C_1	coeficiente associado ao reservatório hidro-pneumático.
C_A, C_B, C_E, C_D	coeficientes associados a equação do NÓ.
C_I e C_{II}	representação genérica das constantes C_A, C_B, C_E e C_D.
C'	variável associada a equação do ENO não-tubo.
C^+ e C^-	retas características positiva e negativa.
c_0	coeficiente da equação ajustada da curva carga x vazão da bomba
D	diâmetro do tubo.
$D(t)$	demanda variável no tempo.
E	módulo de elasticidade do material do tubo.
E_E	coeficiente associado a equação do ENO.
E_N	coeficiente associado a equação do NÓ.
e	espessura da parede do tubo.
F	coeficiente associado a equação do ENO não-tubo.

f	fator de atrito da fórmula universal de perda de carga.
G	coeficiente associado a equação do ENO não-tubo.
g	aceleração da gravidade.
H	carga hidráulica.
H_{atm}	carga atmosférica.
H_B	variação carga produzida pela bomba
H_i	carga hidráulica na seção i na iteração anterior.
H_{Pi}	carga hidráulica na seção i na iteração atual.
H_R	nível do reservatório.
H_S,H_R,H_T	cargas nos pontos S, R, T da curva carga x vazão.
I	número de ordem.
i	identificador da seção.
i	identificador do NÓ.
J	vetor
j	identificador do tubo.
K_V	módulo de elasticidade volumétrico.
K_D	coeficiente de perda de carga associado à válvula.
K_S	coeficiente de perda de carga localizada.
K	coeficiente experimental
k	identificador do tubo.
L	comprimento do tubo.
MC	número de tubos que converge ao NÓ.
MD	número de tubos que diverge ao NÓ.
N1	NÓ de montante.
N2	NÓ de jusante.
n	coeficiente poli tropico.
P	pressão.
Q	vazão.
Q_i	vazão na seção i na iteração anterior.

Q_{Pi}	vazão na seção i na iteração atual.
Q_E, Q_{PE}	vazão do ENO não-tubo.
Q_R,Q_T	vazão nos pontos R e T sobre a curva de carga x vazão da bomba.
R	constante de atrito.
t	tempo.
V	velocidade média da seção transversal de um conduto.
V_1	velocidade média da seção transversal de um conduto no ponto 1
V_2	velocidade média da seção transversal de um conduto no ponto 2
x	distância média ao longo do eixo do tubo.
z, z_P	nível de água no dispositivo de controle.
zp	nível de água na iteração atual.
Δx	intervalo de comprimento de tubo.
Δt	intervalo de tempo.
ΔH	perda de carga.
$\Delta \forall$	volume de ar.
ρ	massa específica.
ε	rugosidade relativa da parede do tubo.
σ	tensão.
γ	peso específico
$\forall_O$	volume de ar no inicio Δt

Resumo

O presente trabalho apresenta as bases para a elaboração de um modelo hidráulico computacional baseado na filosofia de programação orientada por objetos (POO) para o tratamento de problemas de escoamento de fluidos em condutos forçados em regime permanente e transitório. Na modelação hidráulica foi empregado um modelo dinâmico inercial elástico com solução através do Método das Características (MOC). Os estudos de casos apresentados mostram o potencial da ferramenta proposta para a análise de escoamentos de instalações hidráulicas em condutos forçados.

Palavras-chave: método das características, modelo dinâmico inercial, programação orientada a objetos, sistemas hidráulicos.

Abstract

The present work shows the basis for the elaboration of a computational hydraulic model based on the object oriented programming (POO) philosophy for the management of flow problems such as flow rate in pipe systems in steady and unsteady states. The hydraulic modeling is based on the inertial dynamic model with solved through the Method of Characteristics (MOC). The different cases of studies presented in this dissertation show the potential of this tool proposal for the analysis of pipe flow installations in hydraulic systems.

Keywords: method of characteristics, inert dynamic model, object oriented programming, hydraulic systems.

Sumário

Lista de Figuras

Capítulo 1

Introdução

1.1 Considerações sobre o Tema

Esse trabalho propõe o desenvolvimento e implementação de um programa orientado por objeto para análise de escoamentos em condutos forçados em regime permanente e transitório

No desenvolvimento de programas computacionais, além das dificuldades mais comuns para obtenção da solução das equações do escoamento existem as de elaboração de entrada de dados, processamento e visualização de resultados, interação entre usuário e máquina durante o processo de análise e da implementação de novas potencialidades a programas pré-existentes. Com isto, um programa que poderia ser muito útil não é suficientemente explorado porque, entre outros fatores, não é receptivo a extensões para o tratamento de outros problemas.

Do ponto de vista de desenvolvimento de pesquisa em grupo envolvendo *software*, é de fundamental importância que o código fonte gerado por um pesquisador seja facilmente entendido e utilizado novamente por outros em novas aplicações. A programação *procedural* (ou tradicional), normalmente utilizada no desenvolvimento de aplicações, não tem sido adequada sob este ponto de vista. O código fonte pode ser reutilizado, mas o custo desse procedimento requer alto investimento de tempo para o seu entendimento e readaptação, com prováveis erros em sua manipulação, ou restrições à estrutura de dados impostos pela utilização de bibliotecas de rotinas geradas por terceiros.

A filosofia da Programação Orientada por Objetos (POO) se baseia no fato de que as ações (métodos) sobre os dados são definidas junto com os mesmos, levando ao encapsulamento que caracteriza o objeto (ou mais apropriadamente sua classe). Quando um objeto é incluído em uma nova aplicação, apenas os dados necessários para sua criação e utilização e os métodos a que ele responde devem ser de conhecimento público, evitando-se assim a manipulação de sua estrutura de dados por outras porções de códigos distintas daquelas que definem seus métodos. Esse encapsulamento dos dados pelo objeto facilita a manutenção e verificação dos programas.

Adicionalmente, a Programação Orientada por Objetos permite que sejam criadas classes derivadas de classes existentes, especializando-as com o acréscimo de dados e métodos sem a necessidade de manipular o código já existente, testado e considerado estável. O conceito de classe e herança melhora o gerenciamento de dados e a modularização (SCHOLZ, 1992).

A aplicação da Programação Orientada por Objetos aos problemas de análise de escoamento de condutos forçados permite que novas potencialidades sejam adicionadas a um sistema de modo mais simples, com maior rapidez e com menor possibilidade de inclusão de erros em código já testado. Isso é obtido por intermédio do uso da herança através da qual as classes mais específicas representam as novas potencialidades.

Devido às características deste projeto, a biblioteca de objetos poderá ser utilizada por terceiros, sendo então de fundamental importância a documentação detalhada de cada classe, com seus métodos e restrições de uso.

No caso particular de estudo desta dissertação, os escoamentos em condutos forçados em regime permanente e transitório, a programação orientada por objetos, se mostra bastante promissora, o que serviu de motivação para o estudo.

Um sistema hidráulico está sujeito a fenômenos transitórios cujas pressões podem causar sérios problemas aos equipamentos e às tubulações. Por isso, é necessário analisar o fenômeno transiente com o objetivo de descrevê-lo matematicamente afim de eliminar ou minimizar os seus efeitos através de regras próprias de manobras ou de equipamentos projetados especialmente para esse fim (LESSA, 1984).

Desde a década de 60, com surgimento e aprimoramento dos computadores digitais, novos métodos numéricos computacionais foram elaborados, tais como o método explícito de diferenças finitas, o método implícito de diferenças finitas e o método das características (LESSA, 1984). STREETER e WYLIE (1967) investigaram exaustivamente o método das características, obtendo sucesso nas soluções de uma variedade de problemas de escoamento transitório. LUVIZOTTO JR.,(1995) usou o MOC (método das características) e um modelo topológico associado na elaboração de rotinas de simulação para definição de regras operacionais de redes hidráulicas em regime permanente e período extensivo.

A idéia central deste trabalho se fundamenta em explorar a POO para integração das análises em regime permanente e transitório utilizando o MOC. Concentrou-se a atenção no estudo de um sistema orientado por objeto em análises de sistemas de abastecimento de água. Este sistema é formado por uma biblioteca de classes, escritas em um ambiente de programação *Delphi* (*Object Pascal*), baseado no modelo dinâmico inercial elástico. Este modelo tem o potencial de ser utilizado em diversas aplicações, e deve permitir que melhorias sejam implementadas concentrando o trabalho apenas nessas novas potencialidades. Isso resulta, em uma otimização do tempo e esforços necessários para implementar as novas funcionalidades ao sistema.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é o desenvolvimento de uma biblioteca de classes de objetos relacionados a escoamentos em condutos forçados. Essa biblioteca deverá constituir a base de um sistema concebido para ser ampliado, permitindo que novas teorias e idéias envolvendo escoamentos em condutos forçados possam ser rapidamente implementadas com a utilização de objetos especializados, exaustivamente testados, levando a uma otimização de tempo e esforço do pesquisador que vier a introduzir novas potencialidades ao sistema.

Para atingir nossos objetivos delineados, esta Dissertação de Mestrado esta organizado da seguinte forma:

No capítulo 1 é apresentada uma breve introdução e os principais objetivos desta dissertação.

O capítulo 2 contém a revisão bibliográfica da técnica de programação orientada a objetos com ênfase no estudo nas suas principais características.

No capítulo 3 são apresentados os fundamentos conceituais necessários para a sistematização dos procedimentos destinados a elaboração de rotinas de simulação em sistemas de escoamento em conduto forçado com base no MOC.

O capítulo 4 apresenta o desenvolvimento e aplicação da POO, UML.

No capítulo 5 são apresentados os estudos de casos com o modelo desenvolvido.

O capítulo 6 apresenta os resultados e as discussões.

No capítulo 7 são apresentadas as conclusões e finalmente no capítulo 8 são mostradas as principais referências bibliográficas utilizadas nesta dissertação.

1.2.2 Objetivos Específicos e Organização do Trabalho

Os objetivos específicos desta dissertação são:

- Apresentar a base de um sistema computacional baseado na filosofia da linguagem orientada por objetos com emprego do modelo dinâmico inercial elástico fazendo-se uso do Método das Características (MOC) para o tratamento de problemas de escoamento em condutos forçados, em regime permanente e transitório.

- Desenvolver uma biblioteca de classes de objetos relacionados aos sistemas de distribuição de água que permitam uma análise da operação dos sistemas com emprego do MOC.

1.3 Motivação do Estudo

A POO se apresenta como uma proposta atual mais usada no desenvolvimento de software na área da engenharia, uma área especial na ciência da computação. Seus princípios são baseados em abstração, re-usabilidade, encapsulamento e herança, levando os objetos a um conceito real os quais formaram uma integração de partes. Atualmente muitos problemas se devem ao fato das partes não poderem ser integráveis entre si e sendo assim não conseguem trabalhar em conjunto, gerando grande quantidade de "rotinas lixo" (AMELIA DE OLIVEIRA C; FERNANDO COENTE M; GONÇALVES RISO B). Por tanto, a proposta e pensar no problema com outro ponto de vista.

Segundo LUVIZOTTO JR., (1995), existe uma carência de estudos que visem sistematizar o desenvolvimento de modelos computacionais de fenômenos hidráulicos de uma forma geral e em condutos forçados em particular. Esta pesquisa surge devido à ausência de estudos no desenvolvimento de modelos matemáticos computacionais para a solução de fenômenos transitórios como, por exemplo, problemas de escoamento em condutos forçados.

Finalmente, motivados pelo desejo de criar componentes de "*software*" robustos e reutilizáveis, que reduzam os ciclos e o tempo de produção do *software,* escolheu-se a POO, que acompanha uma evolução paralela às técnicas e métodos de engenharia de "*software*".

Capítulo 2

Revisão Bibliográfica

2.1 Introdução

Neste capítulo apresenta-se uma revisão da literatura, de modo a expor o estado da arte no assunto de interesse, servindo para verificar as diversas possibilidades de implementação assim os resultados de outros trabalhos podem ser aproveitados nesta dissertação.

Com a revisão bibliográfica, tem-se, além do ganho conceitual, uma melhor visão do caminho a seguir, em função das experiências relatadas por outros autores. A revisão bibliográfica nesta dissertação consistiu no:

- Estudo da bibliografia específica aos assuntos relativos ao tema do trabalho e,
- Estudo de ferramentas necessárias ao desenvolvimento do trabalho.

A seguir são descritos os itens desta revisão bibliográfica.

2.2 Revisão Bibliográfica Específica

Os principais tópicos necessários para o desenvolvimento desta dissertação são:

- As Ferramentas empregadas
- A Programação Orientada a Objetos (POO)

- UML - *Unified Modeling Language*
- A Solução pelo Método das Características (MOC)

2.3 Ferramentas Empregadas para o Desenvolvimento desta Dissertação

As principais ferramentas necessárias são *softwares*, dentre os quais se destacam o sistema operacional *Windows* e o ambiente *Delphi*. O *Delphi* é uma ferramenta amplamente difundida, e alguns dos recursos que atraíram para sua escolha; o tratamento baseado em formulários orientado a objetos, seu compilador eficiente, seu suporte a banco de dados, sua intima integração com programação em *Windows* e sua tecnologia de componentes. Mas o mais importante foi a linguagem poderosa e familiar *Object Pascal* (mostrando-se mais legível do que *C, C++, Java*), que constitui a base para outros elementos.

2.4 Programação Orientada a Objetos

Essa filosofia da programação difere da programação *procedural*, comum em outros tipos de linguagens científicas, tal como o pascal, por seu comportamento não ser ditado pela seqüência do código e sim pelo comportamento dos objetos componentes do programa.

As principais vantagens da Programação Orientada a Objetos apresentam as seguintes características:

- Encapsulamento
- Herança/especialização
- Polimorfismo

A POO tornou-se muito popular recentemente no desenvolvimento de *software* em diversas áreas (ADELI E YU, 1993; ZIMMERMAN ET AL., 1998; ARCHER ET AL., 1999;

SISTLA ET AL., 2000). Uma destas áreas é a engenharia do *software* que está começando a ter uma maior atenção pelos pesquisadores e programadores. Nos anos anteriores foi dada grande ênfase ao desenvolvimento de algoritmos e de procedimentos para execução de tarefas específicas. Atualmente, quase toda a atenção é dedicada aos aspectos de projeto de *software*, considerando extensão e integração com outros programas e finalmente o uso de uma interface gráfica. Com o crescimento das complexidades nos programas e das novas ferramentas disponíveis, tem-se empregado recentemente método orientado a objetos em lugar da programação *procedural* clássica. De acordo com TABATAI (2002), isto se deve às vantagens do ambiente orientado a objetos em termos de abstração, encapsulamento, modularidade e reuso do código.

O acesso global à estrutura de dados diminui a flexibilidade do software (MACKERLE 2004). Devido à interdependência na arquitetura do programa, sendo oculta e difícil de determinar, torna-se necessário o conhecimento da totalidade do programa. Essas limitações não existem com a programação orientada a objeto, na qual classes, herança e polimorfismo são os principais conceitos. Além disso, o encapsulamento o principio central, que geralmente conduz a programas menores e fornecendo uma melhor gerência de dados.

Uma maneira de se interpretar a POO é mediante a transmissão de uma mensagem a um agente (um objeto) responsável pela ação. A mensagem codifica a petição de uma ação e é acompanhada de qualquer informação adicional ou argumentos necessários para executar a petição. O receptor é o agente ao qual se envia a mensagem. Se o receptor aceita a mensagem, ele aceita também a responsabilidade de executar a ação indicada. Em resposta a uma mensagem, o receptor executará algum método para satisfazer a petição. Temos advertido aqui o princípio da ocultação de informação de acordo com os passos das mensagens. Isto significa que o cliente que envia a petição não precisa conhecer o meio real com que esta será atendida. A ocultação da informação é também um aspecto importante da programação em linguagens convencionais.

ACTOR (1987) descreveu a POO como “animista”; isto é, um processo que cria uma multidão de ajudantes que integram uma comunidade e auxilia o programador na solução do problema. À medida em que os programadores tentavam resolver problemas cada vez mais complexos por meio de um computador, as tarefas ultrapassavam a capacidade dos melhores

programadores pelo seu tamanho. Assim começaram a proliferar equipes de programadores que trabalhavam juntos para empreender grandes tarefas de programação.

A técnica orientada a objeto segue com freqüência o mesmo método aplicado à resolução de problemas na vida diária de acordo com KAY (1977). A mesma tarefa que um programador terminaria em dois meses não poderia ser realizada por dois programadores trabalhando durante um mês. Segundo BROOKS (1975), "ter uma criança dura nove meses, sem importar o número de mulheres que participem". A razão de tal comportamento não-linear é a complexidade devido às interconexões entre os componentes de software, exigindo uma grande quantidade de informações, que tem como objetivo a comunicação entre os diversos membros da equipe de programação.

Durante décadas, têm-se perguntado: "por que a construção de *software* não reflete a elaboração de objetos materiais?". No passado, a reutilização de software foi uma meta muito desejada mas poucas vezes alcançada. As técnicas de POO prevêem um mecanismo para separar com clareza a informação essencial (inserção e recuperação) daquela de pouca importância (formato de registros particulares). Desta maneira, usando as técnicas orientadas a objetos, podemos construir grandes componentes de softwares reutilizáveis.

Devido aos avanços na capacidade de memória e rapidez de execução dos cálculos, bem como na redução dos preços nos computadores, a utilização destes sofreu um impulso nos últimos anos. Esse crescimento, aliado às exigências dos usuários por programas mais complexos e com recursos cada vez maiores, acarretou como conseqüência um desenvolvimento dos campos das linguagens de programação. Um dos maiores desenvolvimentos foi a filosofia da programação orientada a objeto que unem dados e métodos (DIORIO FILHO, 1996). As linguagens tradicionais como *FORTRAN* (nome derivado de "*FORmula TRANslation"*), C ou Pascal interpretam os dados como entidades passivas que podem ser manipulados por "*procedures*" ou "*functions*" a partir de qualquer parte do código (SCHOLZ, 1992).

Segundo DIORIO FILHO (1996), a programação orientada a objeto é diferente da programação *procedural* tradicional porque o programa é formado por um grupo de objetos que se comunicam através do envio de mensagens. Essas mensagens acionam sub-rotinas, chamadas métodos, que agem sobre os dados do objeto responsável pelo envio da mensagem. Sendo a

mensagem a única possibilidade de se acessar os dados do objeto, tal processo fornece vantagens em relação à programação procedural, tais como a menor necessidade de manutenção dos programas e a maior facilidade dessa manutenção quando ela se tornar necessária.

Na evolução das linguagens computacionais surgiu o conceito de estruturação de dados em um único tipo (*record* para Pascal e *struct* para C), permitindo que um conjunto de dados seja definido dentro de uma variável. A orientação a objeto utiliza-se do mesmo conceito de estruturação de dados dentro de um único tipo, ampliando a inclusão de novas sub-rotinas mantendo os dados e as mesmas unidades de modo a transformar esse conceito no conceito de classes.

A POO não consiste simplesmente em umas poucas características novas somadas às linguagens de programação. Ela é uma nova forma de pensar sobre um processo de decomposição de problemas e de desenvolvimento de soluções de programação. A POO considera um programa como uma coleção de agentes amplamente autônomos, chamados objetos. Cada objeto é responsável por cada tarefa especifica.

Um objeto é um encapsulamento do estado (valores de dados) e do comportamento (operações). Assim, um objeto se parece a um módulo ou a um tipo de dado abstrato. O comportamento do objeto fica determinado pela classe do objeto. Cada objeto é um exemplar de alguma classe. Todos os exemplares da mesma classe se comportam de uma forma similar (isto é invocam ao mesmo método) como resposta a uma solicitação similar. A interpretação de uma mensagem (método específico executado) é decidida pelo objeto e pode diferir de uma classe de objetos à outra.

Assim, um objeto, elemento representativo de sua classe, possui os dados e métodos (rotinas ou funções) combinando-os com a finalidade de executar um objetivo para o qual foi criado. Esses dados ficam encapsulados de modo que apenas os métodos do objeto conseguem acessá-los, ou seja, um objeto opera sobre seus próprios dados através de seus próprios métodos de modo que funções externas à classe do objeto não consigam manipular suas variáveis.

2.4.1 Diferenças entre a POO e a Programação Estrutural Tradicional

As linguagens estruturadas são linguagens procedurais onde um programa é dividido em um conjunto de procedimentos e funções. Especificamente, o programador controla a interação entre o código e os dados. O foco principal das linguagens procedurais é a rotina, enquanto que o foco secundário é constituído dos dados que estão sendo manipulados. Nos anos 70, os cientistas das linguagens de computadores voltaram aos conceitos básicos. Eles argumentavam, "E se os dados tornaram-se foco principal? Pois, vivemos num mundo de objetos e não de procedimentos". Observando ao redor uma variedade de objetos cada um tem seu próprio conjunto de características, traços e funcionalidades. Alguns podem ser semelhantes ainda que diferentes, como por exemplo, duas cadeiras feitas de materiais variados. Estas cadeiras compartilham alguns atributos e funcionalidades comuns, e podem ser diferentes. Por exemplo, uma cadeira de escritório é capaz de girar e tem rodas, o que não ocorre com uma cadeira comum.

Podemos observar na Figura 2.1 que a diferença entre a POO e a programação estrutural tradicional ou programação *procedural* está na organização e maneira de agrupar os códigos e os dados.

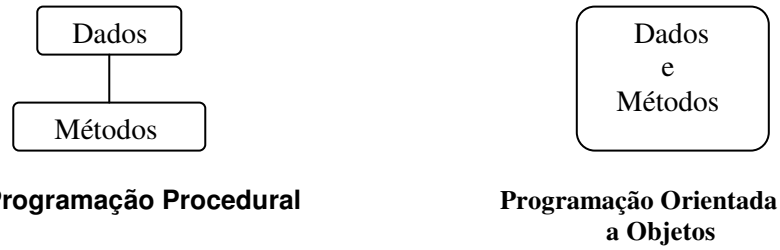

Figura 2.1 - Diferença entre POO e programação tradicional

Segundo DIORIO FILHO (1996), temos então as seguintes diferenças básicas:

- Na programação procedural, as linhas do código são projetadas em torno das funções. Assim sendo, estas são mais importantes neste tipo de programação que na POO, na qual os objetos são mais importantes por serem o núcleo em torno do qual se aglutinam os códigos. Dessa forma, em vez de passar um

objeto para a função, como é realizado na linguagem *procedural*, ele é utilizado para chamá-la de modo que a função fique vinculada ao objeto.

- Na POO pode-se evitar grandes funções que operem sobre casos de múltiplas possibilidades, optando-se por múltiplas classes, normalmente com pequenas funções, que representam os diferentes componentes lógicos de um programa.

2.4.2 Objetos e Classes

CESTA (1996) define uma classe como um tipo determinado pelo usuário que contém o molde, a especificação para os objetos, assim como o tipo inteiro contém o molde para as variáveis declaradas como inteiros. A classe envolve, associa funções e dados, controlando o acesso a estes, defini-la implica em especificar os seus atributos (dados) e suas funções membro (código).

Um programa que utiliza uma interface controladora de um motor elétrico provavelmente definiria a classe motor. Os atributos desta classe seriam: temperatura, velocidade, e tensão aplicada. Estes provavelmente seriam representados na classe por tipos. As funções membro desta classe seriam funções para alterar a velocidade, ler a temperatura, etc.

Os objetos se parecem com os mini-programas pois podem conter tanto dados como métodos, permitindo que possam ser usados para criar objetos mais complexos. Como afirma WEISKAMP ET AL. (1993) "...muito parecido com o uso dado a transistores e redes na montagem de um circuito integrado...".

Os objetos e as classes ampliam o conceito de tipos de dados abstratos ao somar o conceito de herança. As classes podem organizar-se numa árvore de herança hierárquica. As classes que se encontram nos níveis mais baixos na árvore têm acesso e podem usar dados e comportamento associados com classes mais altas da árvore, portanto, tais classes herdam seu comportamento das classes pais.

Ao reduzir a interdependência entre os componentes de software, a POO permite o desenvolvimento do sistema de software reutilizável. Tais componentes podem ser criados e provados como unidades independentes isoladas de outras partes de uma aplicação de software. Os componentes de software reutilizável permitem ao programador trabalhar com problemas num nível mais alto de abstração. Podemos definir e manipular objetos simplesmente em termos das mensagens que os componentes entendem, e de uma descrição das tarefas que eles realizam, passando por alto os detalhes da implantação.

Todo programa de aplicação computacional representa um modelo, no espaço de soluções, de um problema (ou, mais exatamente, da resolução de um problema) do mundo real. A construção de um programa envolve um processo de mapeamento de aspectos de objetos pertencentes ao espaço de problemas para representações abstratas no espaço de soluções, de tal maneira que operações sobre essas representações abstratas correspondam a operações no mundo real. A partir daí, o projetista da aplicação cria algoritmos que, executados em computador, produzirão resultados que podem ser mapeados fisicamente para alguma ação em tempo real, no mundo real ou que serão examinados e mapeados mentalmente por pessoas para resultados no mundo real.

Um modelo computacional baseado em objetos, que trabalham e cooperam entre si, atua através da troca de mensagens para realizar uma tarefa dada. Essas descrições simples podem mascarar algumas características peculiares que devemos agora salientar ou detalhar:

- Uniformidade; nada existe exceto objetos e mensagens. Objetos são a metáfora para processos e dados, enquanto que as mensagens constituem uma metáfora igualmente uniforme para a comunicação entre objetos. Todos os itens em um programa serão objetos, desde somente valores numéricos até as entidades mais complexas abstraídas do mundo real. Por outro lado, todas as operações desejadas serão deflagradas por mensagens de um objeto a outro.

- Processamento; todo processamento ocorre dentro do objeto, e é ativado por mensagens a esse objeto. O objeto reage a uma mensagem solicitando processamento, através do envio de mensagens solicitando processamento a outros objetos, que por sua vez, vão sendo processados até o final. Há um

conjunto de objetos básicos ou primitivos que, ao receberem mensagens, efetuam processamento interno (sem envios de mensagens) e retornam o resultado desse processamento, da mesma forma, como em máquinas comuns. Qualquer operação, independente de sua complexidade termina por se traduzir em operações elementares (ex. soma, subtração, comparação, etc), sobre dados elementares (ex. inteiros, reais, caracteres, etc) e com suporte direto em "*hardware*".

- Paralelismo; um mundo de objetos é inerentemente paralelo. Em um dado instante, um objeto pode estar executando uma ação interna como resposta à solicitação de outro objeto; um terceiro objeto, por outro lado, acaba de enviar uma mensagem a um quarto objeto, que nesse mesmo instante está atendendo a um quinto objeto, e assim por diante.

- Vida de objetos; um objeto nasce ao ser explicitamente criado por um outro objeto. A partir do nascimento, o objeto passa a exibir o comportamento que lhe é peculiar, e interagir na sociedade de objetos, enviando e recebendo mensagens. Em qualquer mundo de objetos, reais ou abstratos, objetos tendem a exibir facetas similares, segundo um outro critério, que permitem agrupá-los (sem perda de individualidade como objetos) em classes. Um primeiro nível de abstração permite identificar classes, de tal forma que cada objeto pertença a uma classe, e dela herde propriedades comuns a todos seus "irmãos" de mesma classe. Temos, então, um universo de objetos e de classes. Um segundo lance de abstração permite constatar o obvio: as classes identificadas mostram propriedades comuns que permitem agrupar (pelo menos algumas dessas) classes em superclasses, de tal forma que se forme uma hierarquia superclasse/classe/objeto. A operação pode ser repetida até ser conveniente, de tal forma que se tenha uma hierarquia como mostrada na Figura 2.2. O desenvolvimento orientado a objetos é centrado na identificação de objetos e na construção de uma hierarquia de classes que resume as propriedades comuns de subclasses e de objetos. De acordo com GOLDBERG (1987), executar um

programa ou sistema é algo tão simples como “criar objetos e disparar uma mensagem”.

Uma outra característica da POO é a herança, um poderoso recurso que lhe permite criar subclasses. Cada subclasse herda as características de sua classe mãe (também conhecida como superclasse). Uma subclasse acrescenta novos atributos de duas ou mais classes mãe.

A Figura 2.2 mostra um exemplo de hierarquia de classes. A subclasse “pressão a montante” é filha da subclasse “controle”. As subclasses “bloqueio e controle” são filhas da classe “válvula”. As quais herdam os atributos e métodos da classe “válvula”. Isto significa que os atributos e métodos da classe “válvula” não precisam ser redefinidos quando se declara as subclasses “bloqueio e controle” apenas os novos atributos e os métodos anulados precisam ser redeclarados.

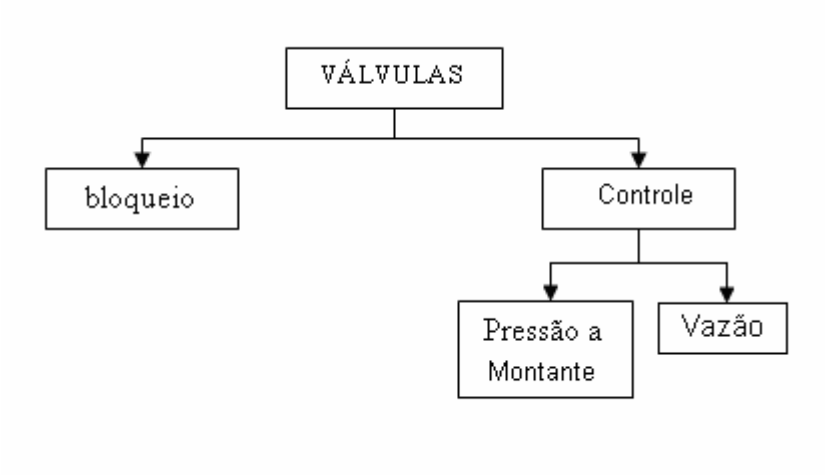

Figura 2.2 - Hierarquia de classes

- Hierarquia de classes; para proceder na tarefa de classificação de objetos e estruturação de hierarquia de classes do particular para o geral, procede-se à observação de objetos, dos quais são abstraídas propriedades para produzir classes.

Generalização e agregação podem então ser empregadas para classificar e estruturar classes em novas classes mais gerais, ou do geral para o particular, principia-se com uma classe geral e, através de especialização, caminhasse para o nível de objetos. Na pratica, é preciso enfatizar, é improvável que uma hierarquia de classes seja projetada estritamente num sentido; tipicamente a compreensão de um universo complexo principiara com uma abordagem do particular para o geral, enquanto o projeto de programas operara predominantemente do geral para o particular.

2.4.3 Métodos

Os métodos podem ser conceituados como as rotinas ou as funções declaradas em uma classe e que atuam sobre os dados dos objetos dessa classe. São as mensagens passadas pelo objeto à sua classe que encontram, entre os métodos nela declarados e implementados, aquele que está associada à mensagem enviada. Assim, o objeto "responde" a mensagem com o seu método.

A mesma mensagem pode ser enviada a objetos diferentes que devido a seus métodos próprios para essa mensagem, responderá de maneira diferente, determinando um comportamento denominado polimorfismo (*do grego* muitas formas). Um exemplo claro está ilustrada na Figura 2.3, que mostra o passo das mensagens através das linhagens hierárquicas. Aqui, operando e operador estão definidos num sentido mais geral. Operando pode significar argumentos atuais de um procedimento e operador o procedimento, operando pode significar um objeto e operador um método, operando pode significar um tipo e operador um objeto deste tipo. Tal comportamento é determinado e é de grande importância além de muito explorado no desenvolvimento deste trabalho.

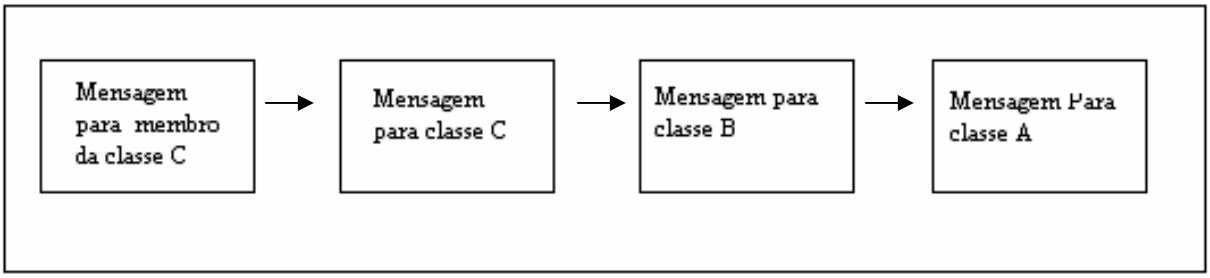

Figura 2.3 - Passando mensagens através da linhagem de classes hierárquicas

Os conceitos de objetos, classes e métodos originam os conceitos de abstração, modularidade e encapsulamento. Abstração é a capacidade de representar entidades como objetos abstratos, suportando sua modelagem de maneira natural. Modularidade significa poder manipular a representação interna de um objeto apenas pelos seus próprios métodos, da sua maneira. Encapsulamento garante que um objeto responda apenas a seus métodos WATSON E CHAN (1991).

Têm-se duas propriedades básicas na POO:

a) Encapsulamento é a propriedade que permite combinar os dados com as operações necessárias para processar os métodos sob um teto de modo que aqueles somente sejam acessados e manipulados por estas. As operações e os dados estarão sempre associados a um objeto da classe. É o encapsulamento que dá aos objetos a aparência de blocos de construção. Sempre que houver uma classe esta propriedade se fará presente. O encapsulamento apresenta dois papeis importantes: i) coloca dados sob um mesmo teto e ii) permite a ocultação de dados.

Geralmente, o encapsulamento tem três finalidades: proteger dados da exposição excessiva, ocultar os detalhes dos dados armazenados ou implementados. Segundo WEISKAMP ET AL. (1993), "Não interessa saber como é feito. Somente interessa que seja feito..." e facilitar o uso em um outro projeto de código já escrito e testado. Assim, o programador nunca precisa acessar diretamente os campos de dados de um objeto, bastando apenas utilizar seus métodos.

b) Herança; é a propriedade que permite estender as classes que já se possui para abranger novos objetos formando subclasses. Nesse caso a superclasse, também denominada classe raiz (termo mais apropriado para a primeira classe da qual as demais derivam direta ou indiretamente), contém dados e métodos que serão a base das outras classes derivadas, as quais, além desses, podem possuir seus próprios dados e métodos. Pode existir um programa completo, com várias classes, sem que se apresente a utilização desta propriedade, muito embora não seja desejável tal acontecimento visto que várias são as vantagens da sua utilização. A herança permite obter uma nova classe a partir de outra já existente denominado-se a classe obtida da subclasse ou classe derivada. Pode-se incluir dados e códigos (métodos) em uma subclasse sem ter que alterar a classe original, usar um código novamente ou modificar o comportamento da subclasse reescrevendo o

código de alguns de seus métodos. Um objeto pode então herdar dados e métodos de uma classe, evitando-se assim a codificação das mesmas linhas de programa para tratar dois objetos de classes diferentes, porém semelhantes.

A herança permite também que um novo aplicativo seja criado com base nas classes de um outro existente sem, contudo, modificá-las. Isso é conseguido apenas derivando-se subclasses que conterão todas as modificações necessárias e traz, entre outras, as vantagens de não se correr o risco de introduzir erros no aplicativo original e também, de não ter que testá-lo (o original) ao termino das modificações.

A classe filha garante no mínimo o mesmo comportamento, "*behaviour*" da classe pai, podendo acrescentar ou redefinir parte do que foi herdado. Por este motivo, uma variável da classe pai pode receber um objeto da classe filha, o comportamento da classe pai fica garantido e o restante (o que a classe filha acrescentou) é perdido. Já uma variável da classe filha não pode receber um objeto da classe pai, porque os métodos definidos para variáveis desta classe passam a não fazer sentido para o objeto contido nesta variável.

A Figura 2.4 reflete o aspecto das características acrescentadas pela classe filha a classe pai, mas somente o fato de uma variável da classe pai poder receber um elemento da classe filha, isto porque como no desenho o pai é desenhado menor que o filho, o leitor tem a tendência de inferir que o pai cabe no filho o que é justamente o contrário do que acontece em termos de variáveis.

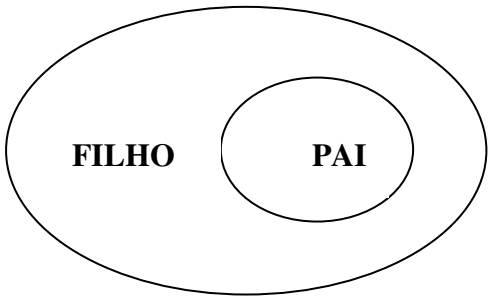

Figura 2.4 - Diagrama das classes

Uma classe filha pode fornecer uma outra implementação para um método herdado, caracterizando uma redefinição "*over riding*" de método. O método deve ter a mesma assinatura (nome, argumentos e valor de retorno), senão não se trata de uma redefinição e sim sobrecarga "*over loading*". A redefinição garante que o método terá o mesmo comportamento que o anterior isto faz com que as subclasses possam ser atribuídas a variáveis da superclasse, pois atendem a todas as operações desta.

2.4.4 Introduzindo Classes e Objetos

A relação entre objeto e classe é a mesma que existe entre variável e tipo. As extensões das linguagens orientadas ao objeto são poucas, mas muito poderosas. A sintaxe geral para declarar uma classe é mostrada aqui:

TYPE
<nome-classe>=OBJECT (<classe mãe>)
<lista de campos de dados>
<lista de cabeçalhos de funções e procedimentos>
END;

2.4.5 Vantagens do uso da POO

Conforme visto na seção 2.4.3, qualquer método que precise ser desenvolvido ou alterado, de modo a adaptar o código que deseja introduzir, pode ser implementado em uma sub-classe. Isso permite a inclusão de qualquer conjunto de classes praticamente sem restrições para o desenvolvimento de ferramentas computacionais direcionadas ao presente trabalho.

No caso de um modelo de simulação de sistemas de abastecimento de água observou-se também que a modularização favorece o desenvolvimento de software por um grupo de pesquisadores, pois cada participante do grupo pode implementar um determinado elemento característico do sistema (válvula, bomba, reservatório, etc) ou um grupo de procedimento concomitante e indiferentemente dos demais. Isso leva uma maior eficiência na execução do produto final.

Na POO, os dados e as rotinas são combinados em objetos. Os objetos contém tanto as características de uma entidade (seus dados) como seu comportamento (seus métodos). Combinando essas características e comportamentos, um objeto sabe tudo o que precisa para fazer seu trabalho.

Para compreender os objetos convém pensar em termos metafóricos. Uma válvula pode ser descrita em termos físicos: o diâmetro, o fator de atrito gerado pelo tipo de material da válvula, e assim por diante.

Mas uma válvula pode também ser descrita em termos funcionais, uma válvula tem a função de abrir e fechar, e assim por diante. Mesmo assim, nem as descrições físicas nem funcionais isoladamente captam a essência do que uma válvula realmente é, o programador precisa das duas.

Na programação tradicional, o programador definiria as características físicas de uma válvula como uma estrutura de dados semelhantes a seguinte:

```
Type   válvula = Record
         diâmetro : Word;
         pressão : Word;
         estado : (aberto , fechado);
End;
```

O programador definiria separadamente os comportamentos da válvula como rotinas e funções, da seguinte maneira:

```
Procedure estadoaberto;
Begin
{...}
End;
Procedure estadofechado;
Begin
{...}
End;
```

Na POO, as características (dados) e os comportamentos (métodos) são combinados em uma única entidade reconhecida como um objeto. Uma válvula definida como um objeto poderia ficar da seguinte forma:

```
Type
        válvula = Object
        diâmetro : Word;
        pressão : Word;
        estado : (aberto , fechado);
 Procedure Init;
 Procedure abertura;
 Procedure fechamento;
End;
```

O objeto que acabamos de apresentar contém declarações tanto para os dados como para os métodos. Na linguagem da POO, as rotinas e funções declaradas em um objeto são conhecidas como métodos. Observe que o objeto define apenas o cabeçalho dos métodos. O verdadeiro código para os métodos é especificado separadamente, da seguinte forma:

```
Procedure valvula.Init;
Begin
        estado := aberto;
        diâmetro := 0;
        pressão := 0;
End;
```

Observe que o método é definido tanto pelo nome do objeto (válvula) como pelo nome do método (aberto), exatamente como se faz referência a um campo contido em um registro. Na realidade, o programador pode pensar em um objeto como um registro que contém tanto campos de dados como declarações e métodos.

Após definir um objeto, você pode declarar variáveis, usando o nome do objeto como é mostrado a seguir.

```
Var
```

A: válvula;

No programa pode-se codificar instruções semelhantes as seguintes:

With A Do
Begin
Init;
abertura;
fechamento;
End;

Agora se podem perceber as vantagens da POO, todas as ações feitas sobre um objeto podem ser feitas por referência ao próprio objeto. Não pode haver confusão sobre qual a estrutura de dados que a rotina abertura usara. Considerando a definição anterior do objeto válvula, o programador pode acessar os campos de dados diretamente. Por exemplo, é perfeitamente válido codificar o seguinte:

A.estado :=aberto;

2.5 UML – *Unified Modeling Language*

A UML é uma linguagem de modelação idealizada para a representação conceitual e física de um sistema, a qual permite especificar quais são as características de um sistema antes mesmo da sua construção. Seus modelos são precisos, completos e podem ser trasladados diretamente a uma variedade de linguagens de programação.

A representação UML é muito útil sob os seguintes aspectos:

- Independe da linguagem de programação a utilizar

- Todo o código pode ser planejado através de uma serie de diagramas que podem descrever o comportamento global de um código, os comportamentos de objetos e a implementação de métodos propriamente dita.

- Induz ao desenvolvimento de documentos previamente à implementação do código, o que em grande parte dos casos conduz à identificação de problemas de implementação antes que estes ocorram.

- Os diagramas UML representam o comportamento de programas orientados a objetos de melhor maneira que os fluxogramas tradicionais.

2.6 Solução pelo Método das Características

O modelo numérico mais utilizado na análise de escoamentos transitórios, é o método das características (MOC) Este modelo transforma as duas equações a derivadas parciais, válidas em todo o plano x-t, em equações a derivadas totais, (C^+ e C^-). Essas equações integradas se expressam na forma de diferenças finitas utilizando-se um intervalo de tempo especificado que permite a solução, partindo-se de um instante conhecido (condição inicial). Tais equações permitem a solução do conhecimento de carga e vazão nos pontos interiores à tubulação. Nos pontos extremos são necessárias, para caracterizar as condições de contorno para caracterizar o desempenho físico de cada elemento da instalação, durante o transitório (ANDRADE, 1994).

Capítulo 3

Metodologia

3.1 Introdução

Para a análise dos escoamentos em condutos forçados devem-se resolver simultaneamente as equações da continuidade (Equação 3.1) e quantidade de movimento (Equação 3.2), equações que fornecem a carga e vazão numa determinada posição da tubulação em função do tempo. Essas equações formam um sistema de equações diferenciais parciais do tipo hiperbólico quase linear cuja solução analítica exata não é disponível, contudo, desprezando ou linearizando os termos não lineares, diversos métodos gráficos, analíticos e numéricos foram desenvolvidos, obtendo-se uma solução aproximada como será vista neste capítulo.

3.2 Modelo Matemático

As equações gerais do escoamento fluvial em conduto forçado podem ser escritas em sua forma geral que permitam análises em condição permanente ou transitória através do emprego das equações de continuidade e da quantidade de movimento.

$$\frac{\partial H}{\partial t} + \frac{a^2}{gA}\frac{\partial Q}{\partial x} = 0 \text{ (equação da continuidade)} \qquad (3.1)$$

$$\frac{\partial Q}{\partial t} + gA\frac{\partial H}{\partial x} + \frac{fQ|Q|}{2DA} = 0 \quad \text{(equação da quantidade de movimento)} \qquad (3.2)$$

Na equação 3.1 H e Q refere-se à carga hidráulica e vazão, respectivamente, numa distância x ao longo do tubo, num instante de tempo t qualquer. Na equação 3.2 "f" é o fator de atrito, da formula universal de perda de carga "A" é a área da seção transversal do tubo, "g"é a aceleração da gravidade e "a"é a celeridade (velocidade de propagação da onda de pressão).

Estas equações permitem descrever as variações de carga e de vazão no tempo ao longo da tubulação, válidas em qualquer ponto do plano (x , t).

As Equações 3.1 e 3.2 formam um sistema de equações de derivadas parciais do tipo hiperbólico, sem solução analítica. No entanto, uma das técnicas numéricas mais utilizadas na solução dessas equações é o método das características (MOC), que permite transformar estas equações em dois pares de equações a derivadas totais, válidas, respectivamente, ao longo das retas C^+ e C^- como indicado a seguir:

$$\frac{gA}{a}\frac{dH}{dt}+\frac{dQ}{dt}+\frac{fQ|Q|}{2DA}=0 \qquad (3.3a)$$

$$\frac{dx}{dt}=+a \qquad (3.3b)$$

$$\frac{-gA}{a}\frac{dH}{dt}+\frac{dQ}{dt}+\frac{fQ|Q|}{2DA}=0 \qquad (3.4a)$$

$$\frac{dx}{dt}=-a \qquad (3.4b)$$

Desta forma torna-se fácil a integração das Equações 3.3a e 3.4a, ao longo das retas características (Equações 3.3b e 3.4b), através da discretização de uma malha no plano (x, t) para uma aproximação de ordem mista do termo de vazão $(Q_P|Q|)$.

Conhecendo-se as condições iniciais de carga e vazão (regime permanente inicial), a cada novo acréscimo de tempo (Δt), encontra-se um novo par de valores de carga e vazão.

3.2.1 Malhas de Cálculo

Para a integração das equações foi testada a malha regular (primeira desenvolvida).

3.2.1.1 Malha Regular

Para o cálculo de vazão e pressão, ao longo de uma tubulação, em vários instantes de tempo diferentes, há a necessidade de uma malha com pontos no plano (x, t), como mostra a Figura 3.1.

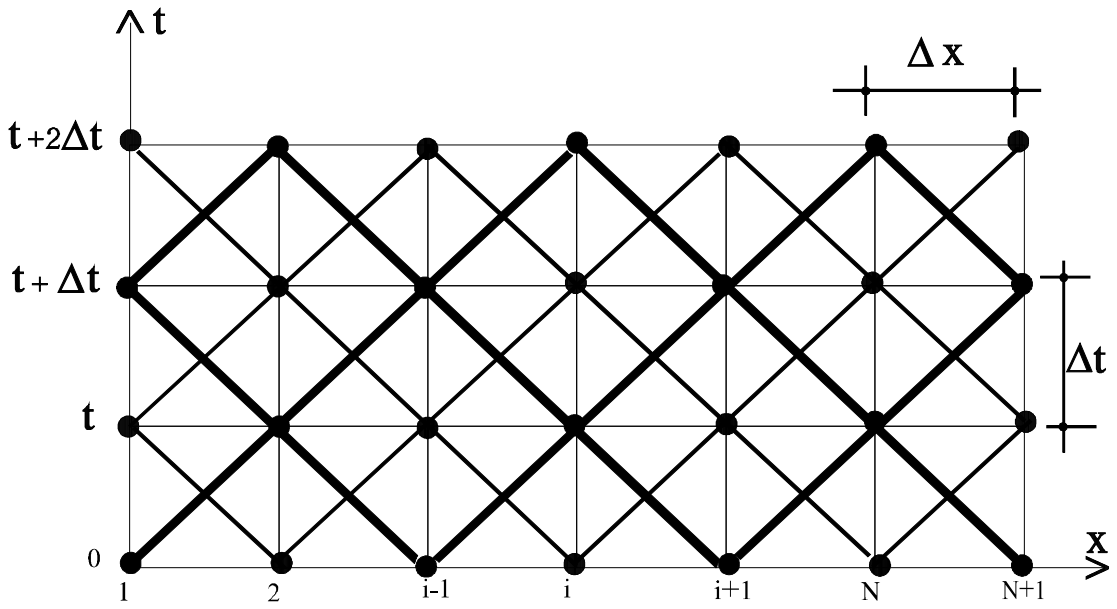

Figura 3.1 - Malha regular

A Figura 3.2 mostra a malha regular que será utilizada para a integração das Equações 3.3a e 3.4a. Obtendo-se o valor de carga e vazão num ponto P, no instante (t+Δt), a partir dos valores conhecidos nos pontos A e B no instante anterior (t) ao longo das retas características C^+ e C^-.

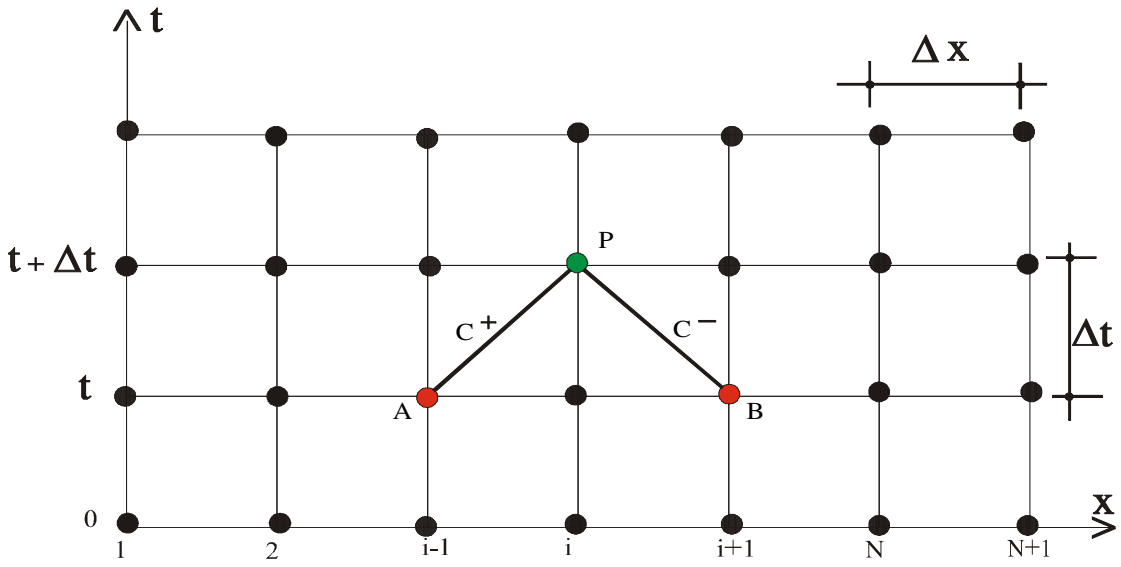

Figura 3.2 - Malha Regular (Método das Características)

A integração dessas equações permite obter os valores de pressão e vazão, apresentados a seguir:

$$C^{+}:\ H_{Pi} = C_A - B_A Q_{Pi} \tag{3.5}$$

$$C^{-}:\ H_{Pi} = C_B + B_B Q_{Pi} \tag{3.6}$$

Nos quais os valores das vazões Q_{Pi} são encontrados igualando-se as Equações 3.5 e 3.6, obtendo-se a seguinte equação:

$$Q_{Pi} = \frac{C_A - C_B}{B_A + B_B} \tag{3.7}$$

Os valores de C_A, C_B e B_A, B_B, constantes referentes aos pontos anteriores conhecidos, são:

$$C_A = \left(H_{i-1} + BQ_{i-1}\right) \tag{3.8}$$

$$C_B = \left(H_{i+1} - BQ_{i+1}\right) \tag{3.9}$$

$$B_A = \left(B + R\left|Q_{i-1}\right|\right) \tag{3.10}$$

$$B_B = \left(B + R\left|Q_{i+1}\right|\right) \tag{3.11}$$

Sendo R o coeficiente de atrito no trecho Δx e B a impedância que são representados, respectivamente, pelas fórmulas:

$$R = \frac{f\Delta x}{2gDA^2} \tag{3.12}$$

$$B = \frac{a}{gA} \tag{3.13}$$

Nota-se que essa discretização resulta em duas malhas distintas como mostrados na Figura 3.1. As variáveis de estado, carga (H) e vazão (Q), são obtidas a cada 2Δt, resultando no dobro de esforço computacional para se obter uma informação num determinado intervalo de tempo, além do fato de que a informação no contorno é calculada a dois intervalos de tempo.

3.3 Modelo Topológico

Com relação à codificação dos elementos que compõe a rede, tem-se um modelo topológico bastante adequado representado por KOELLE (1982), que facilita a identificação desses componentes e simplifica o tratamento das condições de contorno necessários para o equacionamento do modelo elástico.

Nesse modelo, os elementos que compõe a rede (tubos, válvulas, reservatórios, bombas, etc.) são chamados de ENOs. Qualquer elemento que não seja tubo é denominado ENO não-tubo. Cada ENO é interligado entre si por pontos denominados de NÓS, formando-se assim um sistema representado por ENOs e NÓS. Segundo KOELLE (1982) cada NÓ deve estar vinculado no máximo a um ENO não-tubo, para facilitar o equacionamento matemático.

A identificação do ENO é feita por um código de tipo (C), que representa o tipo do elemento; o seu posicionamento na rede, isto é, a sua ordem é representado pelo código (I), um NÓ de montante (N1) e um de jusante (N2), sendo que se atribuí um sentido arbitrário para o escoamento. Desta forma, a identificação completa dos ENOS é feita através de um conjunto de vetores do tipo (C, I, N1, N2). A Figura 3.5 apresenta um exemplo de uma rede hidráulica. .

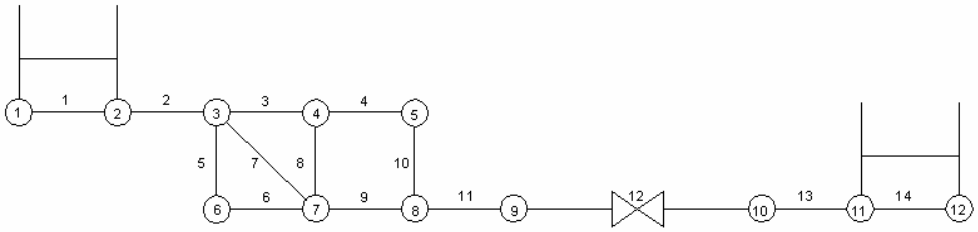

Figura 3.3 - Representação Esquemática de uma Rede

Considerando a codificação dos seus elementos, pela Tabela 3.1, tem-se como representação da rede o vetor (J):

J=	2,1,1,2	1,2,2,3	1,3,3,4	1,4,4,5	1,5,3,6	1,6,6,7	1,7,3,7
	ENO1	ENO2	ENO3	ENO4	ENO5	ENO6	ENO7
J=	1,8,4,7	1,9,7,8	1,10,5,8	1,11,8,9	3,12,9,10	1,13,10,11	2,14,11, 12
	ENO8	ENO9	ENO10	ENO11	ENO12	ENO13	ENO14

Tabela 3.1 - Codificação dos Elementos.

Tipo	Código
Tubos	(1)
Reservatórios	(2)
Válvula	(3)

3.3.1 Equação do NÓ

Numa rede de distribuição de água, num NÓ qualquer, admitindo o regime permanente inicial pode-se identificar, para um dado NÓ, a quantidade de condutos que convergem (MC) ou divergem (MD) suas vazões, dependendo como estas foram arbitradas, inicialmente (Figura 3.4).

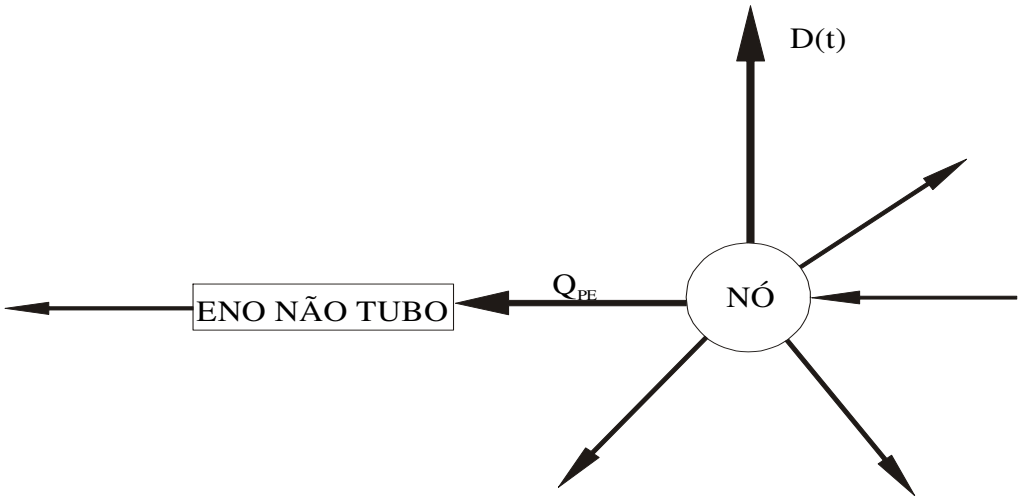

Figura 3.4 - Esquema de um NÓ genérico

A aplicação da equação da continuidade no NÓ fornece:

$$\sum_{j=1}^{MC} Q_P(j) - \sum_{K=1}^{MD} Q_P(k) - Q_{PE} - D(t) = 0 \qquad (3.14)$$

A Figura 3.4 ilustra também um ENO não tubo vinculado ao NÓ, que possui uma vazão Q_{PE} e será positiva quando sair do NÓ e negativa quando ocorrer o inverso. Substituindo-se a equação 3.7 obtém-se a equação do NÓ, como representada a seguir:

$$Q_{PE} = E_{Ni} - B_{Ni} H_{Pi} \qquad (3.15)$$

Sendo que, o índice i refere-se ao NÓ em estudo e E_N e B_N corresponde ao conjunto de "informações" dadas pelas retas características das extremidades dos tubos que convergem e divergem ao NÓ. São obtidas pelas fórmulas:

$$E_{Ni} = \sum_{j=1}^{MC} \frac{C_I(j)}{B_I(j)} + \sum_{k=1}^{MD} \frac{C_{II}(k)}{B_{II}(k)} - D_i(t) \tag{3.16}$$

$$B_{Ni} = \sum_{j=1}^{MC} \frac{1}{B_I(j)} + \sum_{i=1}^{MD} \frac{1}{B_{II}(k)} \tag{3.17}$$

Sendo que os índices j e k referem-se aos condutos vinculados ao NÓ, respectivamente aos que convergem (MC) e aos que divergem (MD), as constantes C_I, C_{II} e B_I, B_{II} são, respectivamente, as constantes das Equações 3.8 a 3.11, no caso da malha regular e $D_i(t)$ refere-se à variação da demanda com o tempo.

Se não houver um ENO não-tubo vinculado ao NÓ, tem-se um caso particular no qual $Q_{PE} = 0$ e o NÓ é apenas um entroncamento de tubos (Figura 3.5).

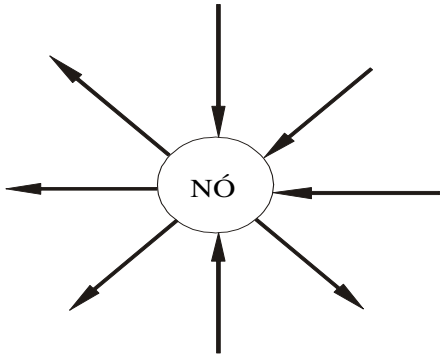

Figura 3.5 - Entroncamento de condutos

Em seguida, tem-se que a equação do NÓ tipo entroncamento fornece diretamente a carga no NÓ:

$$H_{Pi} = \frac{E_{Ni}}{B_{Ni}} \tag{3.18}$$

Uma vez determinada a carga no NÓ, encontram-se as vazões $Q_P(j)$ e $Q_P(k)$ referentes, respectivamente, aos tubos que convergem e divergem a esse NÓ.

$$Q_P(j) = \frac{C_I(j) - H_{Pi}}{B_I(j)} \tag{3.19}$$

$$Q_P(k) = \frac{H_{Pi} - C_{II}(k)}{B_{II}(k)} \tag{3.20}$$

3.3.2 Equação do ENO não-tubo

Para o equacionamento do ENO não-tubo, ele deve estar compreendido entre dois NÓS (Figura 3.6).

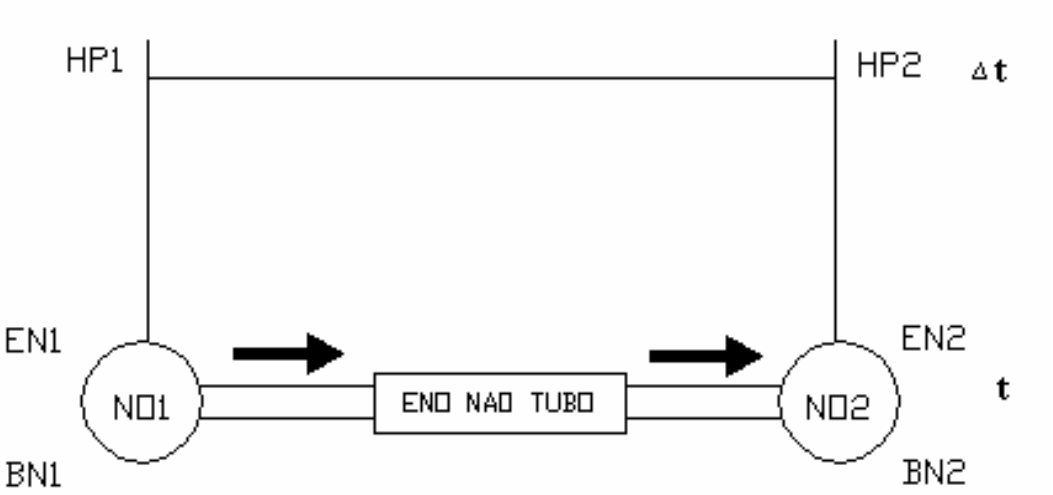

Figura 3.6 - ENO não-tubo entre NÓS

Para cada NÓ, usa-se a equação 3.15. Na Figura 3.6 observa-se que o NÓ de montante a vazão sai do mesmo e para o NÓ de jusante, ocorre o inverso, então, o valor de Q_{PE} será positiva para o primeiro e negativa para o segundo, representada, respectivamente, pelas seguintes fórmulas:

$$Q_{PE} = E_{N1} - B_{N1} H_{P1} \tag{3.21}$$

$$-Q_{PE} = E_{N2} - B_{N2} H_{P2} \tag{3.22}$$

Isolando H_{P1} e H_{P2} das equações acima, obtém-se:

$$H_{P1} = \frac{E_{N1} - Q_{PE}}{B_{N1}} \tag{3.23}$$

$$H_{P2} = \frac{E_{N2} + Q_{PE}}{B_{N2}} \tag{3.24}$$

Em função de $H_{P1} - H_{P2}$, encontra-se a seguinte equação:

$$H_{P1} - H_{P2} = \left(\frac{E_{N1}}{B_{N1}} - \frac{E_{N2}}{B_{N2}}\right) - \left(\frac{1}{B_{N1}} + \frac{1}{B_{N2}}\right) Q_{PE} \tag{3.25}$$

Subtraindo:

$$E_E = \left(\frac{E_{N1}}{B_{N1}} - \frac{E_{N2}}{B_{N2}}\right) \tag{3.26}$$

$$B_E = \left(\frac{1}{B_{N1}} + \frac{1}{B_{N2}}\right) \tag{3.27}$$

Encontra-se a equação geral do ENO não-tubo apresentada a seguir:

$$H_{P1} - H_{P2} = E_E - B_E Q_{PE} \tag{3.28}$$

As equações acima, juntamente, com as características específicas do ENO não-tubo, permitem a determinação das incógnitas H_{P1}, H_{P2}, Q_{PE} e das grandezas que representam o estado do ENO no instante t do transiente.

De acordo com EVANGELISTI (1969), citado por KOELLE (1982), tais expressões podem ser classificadas em dois grupos: i) ENO não dinâmico e, ii) ENO dinâmico.

3.3.2.1 ENO não dinâmico

Representado pelas equações algébricas do tipo:

$$F_1\big(H(t), Q(t), t\big) = 0 \tag{3.29}$$

$$F_1'\big(H(t), Q(t)\big) = 0 \tag{3.30}$$

Na qual Q(t) é a vazão na fronteira, isto é, na extremidade do conduto considerado. Será autônoma se não contiver, explicitamente, a variável t (equação 3.30), e não-autônoma, caso contrário (equação 3.29).

Depreende-se, portanto, que neste tipo de fronteira, a relação entre H e Q não é decorrência da evolução do transiente, isto é, ela é definida, a priori, na concepção da instalação e pelo operador.

São exemplos deste tipo de contorno:

- Variações das propriedades físicas e geométricas dos condutos (mudança do material, espessura ou diâmetro).
- Entroncamento de vários tubos.
- Reservatórios de grandes dimensões que podem ser considerados com nível constante durante o transiente.
- Válvulas não servo-controladas nas quais a manobra é definida pelo operador.
- Orifícios abertos para a atmosfera.

Para os ENOS não dinâmicos a equação algébrica para a aplicação do processamento numérico é, geralmente, do tipo:

$$H_{P1} - H_{P2} = A' + B' Q_{PE} + C' Q_{PE} \left| Q_{PE} \right| \tag{3.31}$$

|Denominada equação particular do ENO, sendo que A', B' e C' são constantes características do elemento não tubo. Combinando-se a equação 3.28 com a 3.31, obtém-se:

$$Q_{PE} \left| Q_{PE} \right| + F Q_{PE} + G = 0 \tag{3.32}$$

sendo:

$$F = \frac{1}{C'} \left(B' + B_E \right) \tag{3.33}$$

$$G = \frac{1}{C'} \left(A' - E_E \right) \tag{3.34}$$

Segundo KOELLE (1982), utilizando uma proposta sugerida por Betâmio de Almeida, as duas equações acima podem ser escritas de forma única pela seguinte equação:

$$Q_{PE} = -\frac{2G}{F + \sqrt{F^2 + 4|G|}} \tag{3.35}$$

Com o valor de Q_{PE} determinado através da equação 3.35, consegue-se calcular todas as incógnitas (H_{P1}, H_{P2}, $Q_P(j)$, $Q_P(k)$) que envolve o ENO não-tubo, os NÓS e os tubos.

No caso particular em que um dos Nós é exterior, isto é, está em contato com a atmosfera, como é o caso de reservatório com nível constante, válvulas de alívio ou orifícios de descarga, a carga no Nó correspondente será definida pela cota geométrica do nível de água, ou do dispositivo, e o equacionamento poderá ser simplificado.

A representação das fórmulas referentes às condições de contorno dos elementos que compõe uma rede hidráulica segue a estrutura proposta por LUVIZOTTO JR.,(1995).

3.3.2.2 ENO dinâmico

Neste tipo de fronteira, a relação entre H e Q é decorrência da evolução do transiente, isto é, a ela esta vinculada, não pode, por tanto, ser previamente fixada.

As expressões matemáticas envolvem k grandezas adicionais, específicas da fronteira, que variam durante o transiente. São necessárias (k+1) equações para representar o comportamento da fronteira durante o transiente. Nas instalações hidráulicas usuais as expressões serão do tipo:

$$G_i = \frac{dG_i}{dt} = F_2(H(t), Q(t), G_i(t), t) \tag{3.36}$$

$$F_2'(H(t), Q(t), G_i(t), t) = 0 \tag{3.37}$$

São exemplos deste tipo de contorno:

- As máquinas hidráulicas de fluxo nas quais a rotação e o torque no eixo são grandezas adicionais que deverão ser determinadas a cada instante.

- O reservatório hidropneumático para o qual o volume de ar no seu interior é variável durante o transiente.

- As chaminés de equilíbrio e os tanques alimentadores para os quais os níveis de água são variáveis durante o transiente.

- As válvulas servo-controladas nas quais a abertura instantânea é definida pela carga na instalação.

3.4 Dispositivos para segurança de instalações

Todo evento que cause uma brusca variação da vazão em escoamento numa instalação de conduto forçado, exige que a massa fluida seja acelerada ou desacelerada a partir de sua velocidade inicial (de regime permanente), para isso, desenvolvem-se forças de superfície, na forma de pressões transitórias, que se somam o se subtraem às pressões anteriormente observadas. Estas pressões podem levar a valores que comprometam a instalação, quer por rompimento do conduto, devido a excessivas pressões internas, que por esmagamento ou por flambagem, devido a pressões internas reduzidas. Outros efeitos também estão associados ao regime transitório e podem causar a inapetência funcional das instalações hidráulicas, alguns destes são:

- Desgaste por cavitação.

- Ruptura da veia líquida em pontos da instalação.

- Fadiga do material, provocada pela variação continua do estado de tensão.

- Ruidos excessivos.

- Ressonância de vibrações.

O estudo de regime transitorio visa localizar possíveis problemas decorrentes do fenômeno (qualificá-los e quantificá-los), de tal forma que possam ser estabelecidos procedimentos adequados de manobras ou projetados dispositivos para a proteção da instalação.

Os dispositivos empregados na atenuação das pressões transitórias são concebidos para um dos seguintes objetivos:

- Controle de pressões mínimas
- Controle de pressões máximas
- Controle de pressões mínimas e máximas

Este controle é exercido por dispositivos de atenuação, que seguem um dos seguintes principios básicos para obter seus objetivos:

- Uso de energia acumulada
- Uso de água armazenada
- Uso de potenciais de energia e de água armazenada
- Uso de ligações temporarias como o exterior, permitindo descargas de água ou entrada de ar

Seguindo um destes princípios encontram-se uma variedade de dispositivos para a proteção de instalações hidráulicas contra os efeitos transitórios. Entre os mais utilizados estão:

- Reservatórios hidro-pneumáticos
- Chaminés de equilíbrio
- Reservatórios ou tanques unidirecionais
- Válvulas de alívio

3.5 Condições de Contorno

As condições de fronteira ou de contorno são expressões matemáticas capazes de representar o comportamento físico do escoamento em ENOS não-tubos, interpretando os resultados das manobras executadas nesses elementos durante a ocorrência do transiente hidráulico.

3.5.1 Válvula

Considerando-se que o ENO não-tubo da Figura 3.6 seja uma válvula, a perda de carga que ocorre entre NÓS corresponde à válvula, e esta dada por:

$$\Delta H = K_D V^2 / 2g \tag{3.38}$$

No qual K_D é o coeficiente de perda de carga associado à válvula. Sabe-se que ΔH é a diferença de carga entre os NÓS 1 e 2. Assim, obtém-se:

$$H_{P1} - H_{P2} = K_D V^2 / 2g = \frac{K_D}{2gA^2} Q_{PE} |Q_{PE}| \tag{3.39}$$

Comparando-se a equação acima com a Equação 3.31, encontram-se os valores das constantes A', B' e C' da válvula, que valem:

$$A' = B' = 0\text{'} \tag{3.40}$$

$$C' = K_D / 2gA^2 \tag{3.41}$$

Com essas constantes, determinam-se F e G, respectivamente:

$$F = -B_E / C' = \frac{2gA^2}{K_D} B_E \tag{3.42}$$

$$G = -E_E / C' = -\frac{2gA^2}{K_D} E_E \qquad (3.43)$$

Assim, consegue-se obter o valor de Q_{PE} através da Equação 3.35 e as cargas nos NÓS de montante e jusante através das Equações 3.23 e 3.24, respectivamente.

3.5.2 Reservatório

Observando novamente a Figura 3.6, neste caso considera-se que o ENO não-tubo é um reservatório de nível constante (caso particular da condição de contorno). Conhecendo-se o seu nível (H_R), encontram-se as cargas nos NÓS:

$$H_{P1} = H_{P2} = H_R \qquad (3.44)$$

Como não existe variação de carga entre NÓS, considera-se que a vazão (Q_{PE}) é nula:

$$Q_{PE} = 0 \qquad (3.45)$$

3.5.3 Bombas

Usando novamente a Figura 3.6 consideramos o ENO não tubo neste caso como uma bomba. A variação de carga produzida entre os NÓS de montante e de jusante, em uma instalação de bombeamento pode ser expressa na forma:

$$H_{P2} - H_{P1} = H_B \qquad (3.46)$$

No qual H_B representa a variação de carga produzida pela bomba, usualmente fornecida em função de vazão através da curva carga x vazão. Esta curva pode ser representada através de um conjunto de pontos discretos, quando usadas em rotinas de simulação. Alternativamente, um polinômio de segunda ordem ajustado a estes pontos pode ser utilizado (FOX, 1977).

$$H_B = a_0 + B_0 Q_B + C_0 Q_B^2 \tag{3.47}$$

A forma polinomial apresentada na Equação 3.47 é bastante adequada para representação da máquina operando em condição normal ($Q_B>0$ e $H_B>0$) para a análise do regime permanente.

Sendo a_o,b_o e c_o coeficientes a serem determinados e Q_B é a vazão bombeada.

A determinação dos coeficientes a_o, b_o e c_o pode ser feita através de três pontos pertencentes á curva da bomba. Se forem tomados os pontos S, R, T (Figura 3.7), relativos à carga de "*Shutt-off*" (S), ao ponto de máximo rendimento (R) e a um ponto qualquer(T), pode-se obter:

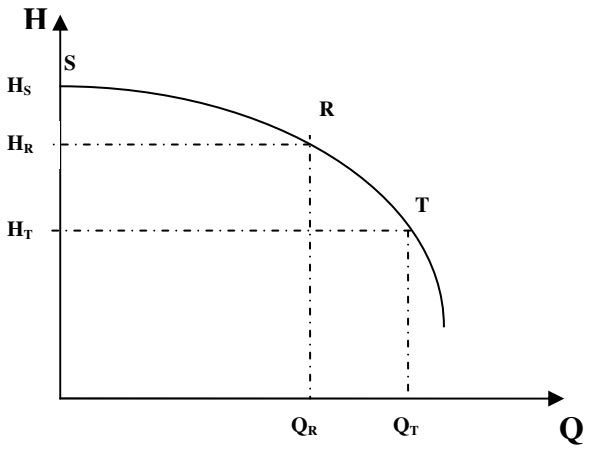

Figura 3.7 – Curva carga x vazão

$$a_0 = H_S \tag{3.48}$$

$$b_0 = \frac{(H_S - H_T)Q_R^2 - (H_S - H_R)Q_T^2}{Q_R Q_T^2 - Q_T Q_R^2} \tag{3.49}$$

$$c_0 = \frac{(H_S - H_R)Q_T - (H_S - H_T)Q_R}{Q_R Q_T^2 - Q_T Q_R^2} \tag{3.50}$$

Das Equações 3.46 e 3.47, com base na Equação 3.31, pode-se escrever:

$$H_{P1} - H_{P2} = -a_0 - b_0 Q_{PE} - c_0 Q_{PE} \mid Q_{PE} \mid \tag{3.51}$$

Da Equação 3.51 se obtém, observando (Equação 3.31), que:

$$\overline{A} = -a_0 \tag{3.52}$$

$$\overline{B} = -b_0 \tag{3.53}$$

$$\overline{C} = -c_0 \tag{3.54}$$

A partir destes valores calculam-se as constantes F e G, respectivamente:

$$F = \frac{1}{\overline{C}}(\overline{B} + B_E) = \frac{1}{c_0}(b_0 - B_E) \tag{3.55}$$

$$G = \frac{1}{\overline{C}}(\overline{A} - E_E) = \frac{1}{c_0}(a_0 - E_E) \tag{3.56}$$

Com estes valores se obtém a vazão pela bomba (Q_{PE}) através da Equação 3.35.

Uma forma analítica mais geral, que permite a caracterização da máquina em suas diversas condições de operação, de fundamental importância na análise de regimes variados, pode ser escrita em termos de séries de Fourier (KOELLE E ANDRADE, 1990; LUVIZOTTO JR., 1992).

3.5.4 Chaminé de Equilíbrio

As chaminés de equilíbrio são tanques abertos para a atmosfera e ligados em derivação ao sistema hidráulico. Em seu funcionamento permite que fluxos de água entrem e saiam de seu interior controlando desta forma variações bruscas de vazão na instalação.

No regime permanente a linha piezométrica tangencia a superficie livre da água no interior do tanque. No regime transitório a superficie livre da água abaixa quando a carga

piezométrica se reduz, provocando uma descarga da água armazenada para a instalação, atenuando as baixas pressões ou elevando-se quando a carga piezométrica cresce, devido ao fluxo da água da instalação para o reservatório, aliviando as pressões internas.

As dimensões de uma chaminé devem ser tais que evitem o transbordamento (ou alternativamente permita o transbordamento através de um vertedor) e a entrada de ar na tubulação, nas condições de nível máximo e mínimo, respectivamente. As seqüências de cálculo das condições de contorno são:

a. Cálculo das constantes:

$$BN = 2A/\Delta t \tag{3.57}$$

$$EN = BN.Z + QE \tag{3.58}$$

$$Ee = EN_1/BN_1 - EN_2/BN_2 \tag{3.59}$$

$$Be = 1/BN_1 + 1/BN_2 \tag{3.60}$$

b. Cálculo de variáveis secundárias:

$$F = 1/Co.(1/BN_1 + \Delta t/2A) \tag{3.61}$$

$$G = -1/Co(EN_1/BN_1 - \Delta t/2A(2ZA/\Delta t + QE)) \tag{3.62}$$

c. Cálculo das variáveis de estado:

$$Zp = z + (QPe + Q)\frac{\Delta t}{2A} \tag{3.63}$$

d. Com estes valores se obtém a vazão nas extremidades da Chaminé (Q_{PE}) através da Equação 3.35.

Com os valores obtidos de Q_{Pe}, H_{Pe},Z_p, faz-se à atualização para o próximo passo.

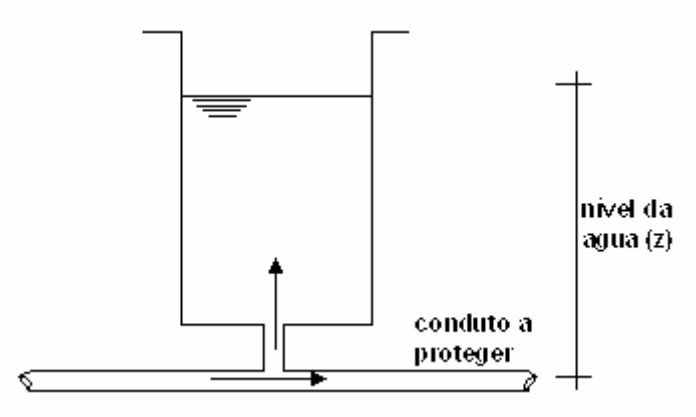

Figura 3.8 – Chaminé de Equilíbrio

3.5.5 Reservatório Hidro-pneumático

Os reservatórios hidropneumáticos são tanques metálicos estanques, preenchidos parcialmente com água e complementados com um gás (normalmente se emprega o ar ou o nitrogênio); o gás pode estar em contato direto com a água, neste caso é utilizado um compressor ou um suprimento de gás, ou ainda, separado por uma membrana flexível ou pistão, evitando o contato direto do líquido como o gás e a conseqüente absorção do gás pela água, mantendo-se assim sua massa praticamente constante.

A pressão no interior do reservatório deve ser tal que equilibre a pressão de serviço da instalação (em regime permanente), no ponto em que está instalado.

No surgimento de uma onda de pressão, tais reservatórios poderão absorver pressões elevadas através da compressão do gás e as quedas de pressão através da descarga do volume de água acumulado em seu interior.

A partir do volume de inicio Δt pode-se calcular a variável secundária C1.

$$H_A = HN_1 - Z + H_{atm} \tag{3.64}$$

$$C_1 = H_A\left(\forall_O + \Delta\forall\right)^n \tag{3.65}$$

Apresenta-se a seguir uma função para o cálculo de $Q_{PE,}$ sendo não linear e solucionada pelo método de Newton-Raphson. A estimativa inicial do processo iterativo esta dada pelas seguintes equações:

$$Q_{PE}{}^{0} = Q_P \tag{3.66}$$

$$f\left(Q_{PE}{}^{O}\right) = Q_{PE}{}^{O} + \frac{\left(BN_{1.}C_1\right)}{\left(\forall_O + \left(Q_{PE}{}^{O} - Q_E\right).\Delta t\right)^n} - EN_1 + BN_1\left(Z - H_{atm}\right) \tag{3.67}$$

$$f'\left(Q_{PE}\right) = 1 - \frac{\left(BN_{1.}C_1.n.\Delta t\right)}{\left(\forall_O + \left(Q_{PE}{}^{O} - Q_E\right).\Delta t\right)^{n+1}} \tag{3.68}$$

$$Q_{PE} = Q_{PE}{}^{0} - \frac{f\left(Q_{PE}{}^{0}\right)}{\frac{df\left(Q_{PE}{}^{O}\right)}{d\left(Q_{PE}\right)}} \tag{3.69}$$

Tendo-se calculado Q_{PE}, pode-se determinar as outras variáveis do processo:

$$\Delta\forall = \frac{\left(Q_{PE} + Q_E\right)}{2}.\Delta t \tag{3.70}$$

$$Z = Z + \frac{\Delta\forall}{A} \tag{3.71}$$

$$\forall_O = \forall_O + \Delta\forall \tag{3.72}$$

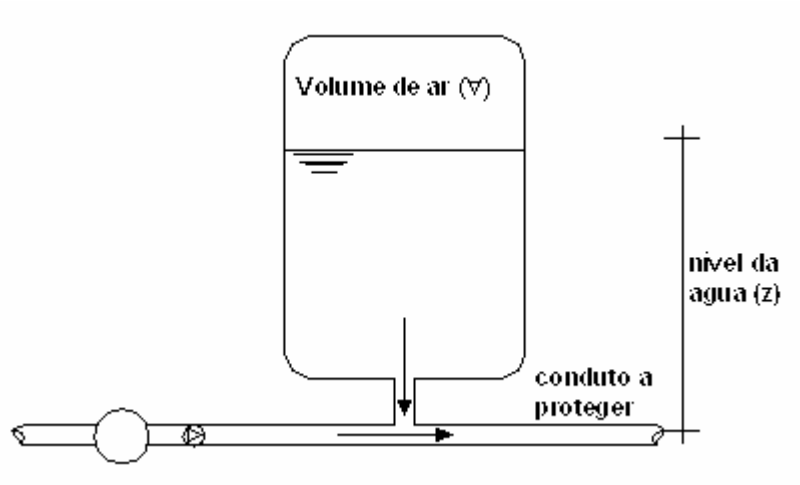

Figura 3.9 – Reservatório Hidro-pneumático

3.5.6 Reservatório ou Tanque Unidirecional

Estes reservatórios unidirecionais são tanques abertos à atmosfera, de altura normalmente bem inferior à das chaminés de equilíbrio. Armazena em seu interior certo volume de água. Sua ligação com a instalação é feita através de uma derivação dotada de válvula de retenção que permite somente fluxo no sentido do reservatório para a instalação.

Em condições de regime permanente a pressão na instalação é superior a do tanque, neste caso a válvula de retenção impede que haja fluxo da instalação para o reservatório. No regime transitório, para o caso de uma onda de depressão, a linha piezométrica cai a níveis inferiores ao nível da água no interior do reservatório, neste caso processa-se a descarga do tanque, o que possibilita um maior tempo para que se anule a vazão em escoamento, atenuando-se assim as depressões excessivas. A re-alimentação dos reservatórios unidirecionais é feita através de uma outra derivação da instalação de pequeno diâmetro, com uma bóia para controle do nível da água.

As seqüências de cálculo das condições de contorno são similares à rotina de cálculo da chaminé de equilíbrio. Os valores EN1 e BN1 são conhecidos para o Nó (1), no instante t em função da configuração da instalação.

Conhecendo como dados de entrada a área do reservatório, coeficiente de orifício, nível máximo e vazão de enchimento calculam-se os valores de BN2 e EN2:

$$BN_2 = 2A/\Delta t \tag{3.73}$$

$$EN_2 = BN_2 Z + Qe \tag{3.74}$$

Com essas constantes, determinam-se F e G, respectivamente:

$$F = 1/C\left(1/BN_1 + 1/BN_2\right) = B_E/C \tag{3.75}$$

$$G = -1/C\left(EN_1/BN_1 - EN_2/BN_2\right) = -E_E/C \tag{3.76}$$

Assim, consegue-se obter o valor de Q_{PE} através da Equação 3.35 e as cargas nos NÓS de montante e jusante através das Equações 3.23 e 3.24, respectivamente.

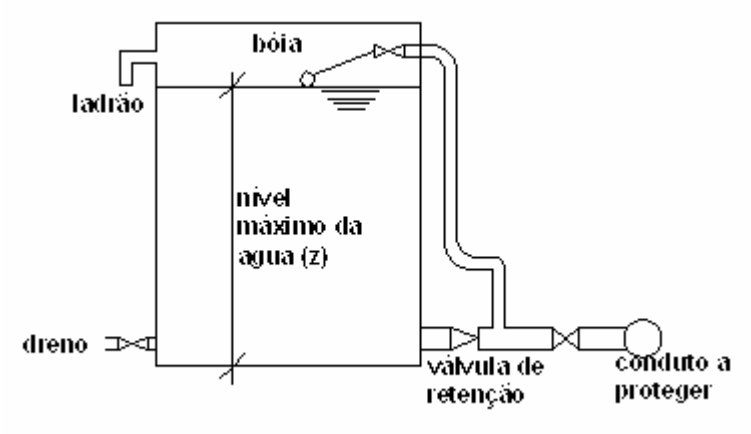

Figura 3.10 – Reservatório ou Tanque Unidirecional

Capítulo 4

Aplicação da Metodologia Orientada a Objetos

4.1 Introdução

A análise e desenho deste trabalho se realizou com a metodologia orientada a objetos, sustentado com a UML, na representação de todas as fases do projeto informático, tanto nos casos de uso como nos desenhos de diagramas e objetos (item 2.5). Dentro da análise orientada a objetos procurou-se identificar e descrever os objetos e conceitos dentro do domínio do problema e também se definiu os objetos lógicos do software que finalmente são implementados na linguagem da POO, *Delphi*.

A orientação a objetos é importante para o desenho de software. O UML incide sobre isso ao permitir gerar modelos de objetos fáceis de usar e compreender para depois convertê-los em software (Item 2.5).

4.2 Modelo de Objetos

4.2.1 Classes

As classes são representadas mediante caixas divididas em três partes: Na parte superior se mostra o nome da classe, na média os atributos e na inferior as operações. A classe é representada de forma esquemática, com os detalhes, no caso atributos e operações.

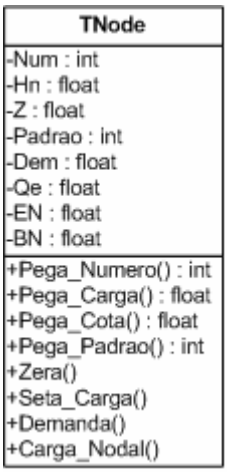

Figura 4.1 - Notação para classe Nó

A Figura 4.1 mostra a representação visual da classe TNode, seguindo nossa abstração. Essa classe, conforme estamos modelando, pode servir de base ao desenvolvimento de aplicações futuras. Por exemplo neste caso a classe TNode encontra-se dividida em três partes: Na parte superior é apresentado o nome da classe, na media são apresentados os atributos e na inferior são apresentadas as operações.

É importante salientar que, no processo de construção de um programa orientado a objetos, deve-se procurar um modelo que melhor represente o mundo real. Assim, pode parecer estranho que na modelagem de uma classe, a qual consiste na abstração de nós do mundo real, apareçam métodos cujo objetivo seja atribuir novo valor a demanda. Tais métodos se justificam para possibilitar alterações desses atributos, em situações nas quais se fizer necessário. Se, por exemplo, em algum ponto de nosso modelo tivermos um objeto da classe TNode, o qual está representando determinado nó do mundo real, e quisermos, por alguma razão de implementação, que esse objeto passe a representar outro nó do mundo real, então tais métodos serão necessários. Métodos que ao serem executados pela instância fazem com que os atributos recebam novos valores, também se fazem necessários quando a instância acabou de ser criada e seus atributos não foram explicitamente inicializados.

Além disso, em muitas situações fora da classe, quando a instância apresenta um atributo que é representado por um valor, faz-se necessário conhecê-lo. Isto só será possível se na

classe for definido algum método que, ao ser executado, faça com que o objeto informe o valor do atributo.

4.2.2 Modelagem de Classes

A construção do modelo de resolução de um problema passou pela identificação de objetos envolvidos neste problema e a modelagem de suas respectivas classes. Clareza e legibilidade foram as características importantes levadas em consideração na modelagem das classes. Além disso, foi fundamental que os identificadores utilizados, tais como nome das classes, nome dos atributos, nome dos métodos, sejam significativos. Assim, o nome de um atributo deve expressar adequadamente o seu conteúdo ou o que representa. Da mesma forma, o nome de um método deve expressar de maneira significativa a tarefa que implementa.

Ao modelar as classes, foram definidas, para cada tarefa distinta, um método distinto. A definição de um método distinto para cada serviço distinto, além de tornar a classe mais legível, facilita a implementação e utilização de objetos dessa classe.

4.2.3 Generalização/Especialização

Por meio da operação de generalização/especialização foi organizada cada uma das classes, em uma estrutura hierárquica, nos quais classes mais genéricas deram origem a classes mais especializadas. Considerando-se por exemplo, a classe que agrega as válvulas de maneira geral. Tal classe pode ser especializada em várias subclasses. Dependendo da abstração realizada, a partir da classe que representa válvulas em geral, podemos, por exemplo, especializar duas outras classes, uma que define as características das válvulas que são de retenção, e outra relativa às válvulas de alívio. Figura 4.2.

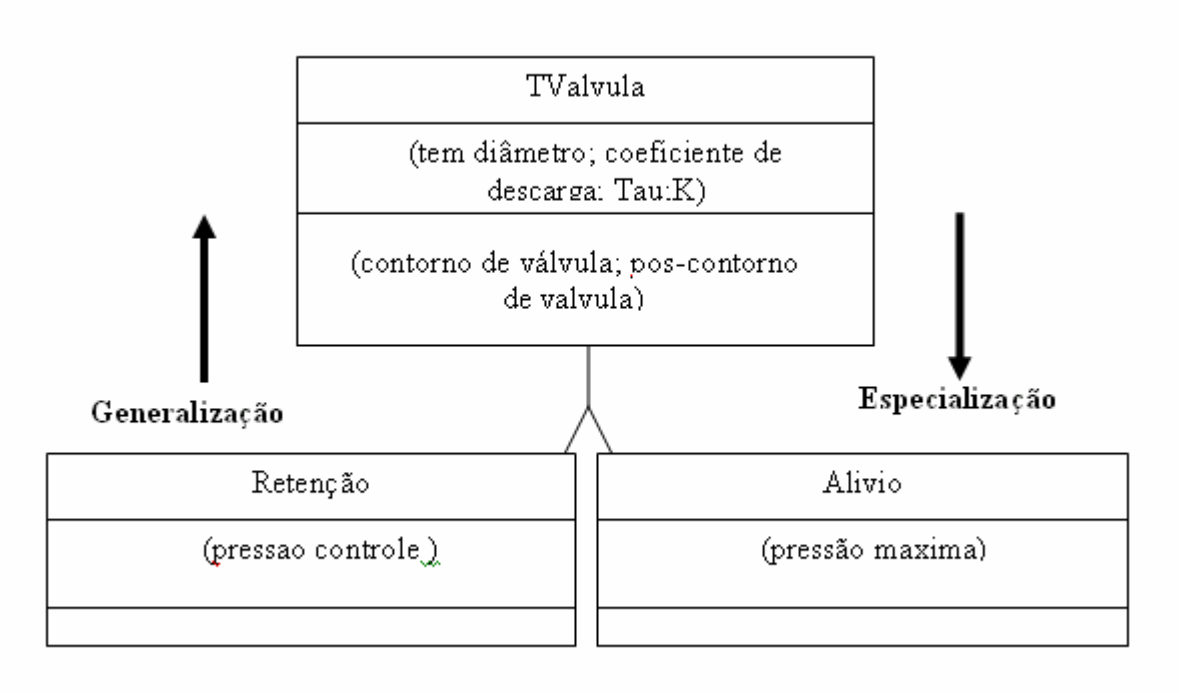

Figura 4.2 – Operação de Generalização/Especialização

No exemplo, representado pela Figura 4.2, uma instância da classe alívio, além dos atributos específicos definidos pela classe alívio, incorporará todos os atributos da classe válvula. Uma classe derivada pode, e deve, se for o caso, ser especializada em outras classes. Assim, para a abstração representada pela Figura 4.3, a classe TEno, considerando-se o regime de trabalho, pode ser especializada em válvula, bomba, reservatório, tubo. Também a classe válvula pode ser especializada no sentido de se definir as características especificas de alívio e retenção.

O processo de especialização foi aplicado às situações onde se tem existência da relação "é um tipo de". Assim válvula de alívio é um tipo de válvula, válvula de retenção é um tipo de válvula. A classe válvula define todas as características relativas a qualquer válvula, enquanto que as classes alívio e retenção definem, respectivamente, as características específicas.

Em relação à classe válvula, as classes alívio e retenção são denominadas subclasses, ou classes derivadas, ou também classes descendentes. Conseqüentemente, a classe válvula é denominada de superclasse, conforme mostra a Figura 4.2.

A característica fundamental do processo de especialização de classes consiste no mecanismo da herança. Por esse mecanismo, a classe mais especializada herda todas as

características da superclasse, ou seja, todos os atributos e métodos definidos na superclasse serão herdados pela subclasse.

4.2.4 Herança

O processo de especialização de classes implica no mecanismo de herança. A existência da herança no processo de especialização, consiste em característica fundamental na programação orientada a objetos, pois possibilita a reutilização de código. Na resolução do problema da Figura 4.3 identificamos a existência de um objeto que representou a Válvula. Como toda Válvula é um Eno, ou seja, toda Válvula "é um tipo de Eno", definimos a classe Válvula como sendo subclasse da classe Eno. Com isto, não precisamos escrever novamente instruções (código) relativas a informações do Eno, as quais o mesmo possui por ser uma Válvula.

A herança é uma característica transitiva, isto é, ocorre através de múltiplos níveis de especialização. Assim, no nosso exemplo, a classe Válvula herdou todas as características definidas na classe TEno, e a classe Retenção herdou todas as características de Válvula e de TEno. Ou seja, ao declararmos uma classe como subclasse de outra, todas as características das classes superiores (também chamadas de classes ancestrais) serão herdadas pela classe derivada.

A reutilização do código é uma das principais características da modelagem orientada a objetos e desenvolvimento de qualquer sistema deve ser feito tendo-se em mente tal característica. Ao definirmos nosso modelo para resolução do problema, modelamos as respectivas classes a partir da definição de classes gerais e, gradativamente especializamos classes, de forma a obter um conjunto de classes hierarquicamente organizadas. Tais classes são agrupadas no sentido de integrar uma biblioteca, podendo ser disponibilizadas aos estudos, quando do desenvolvimento de uma nova aplicação. Obviamente a definição desse primeiro modelo implicou na construção de uma série de classes, as quais são totalmente novas. Entretanto, se ao modelarmos novas aplicações levarmos sempre em consideração a definição de classes segundo uma estrutura hierárquica de generalização/especialização, poderemos gradativamente ampliar a biblioteca. Com uma boa biblioteca de classes, o índice de reutilização de código tende a aumentar a cada nova aplicação desenvolvida, e consequentemente, o custo de desenvolvimento tornar-se-á menor.

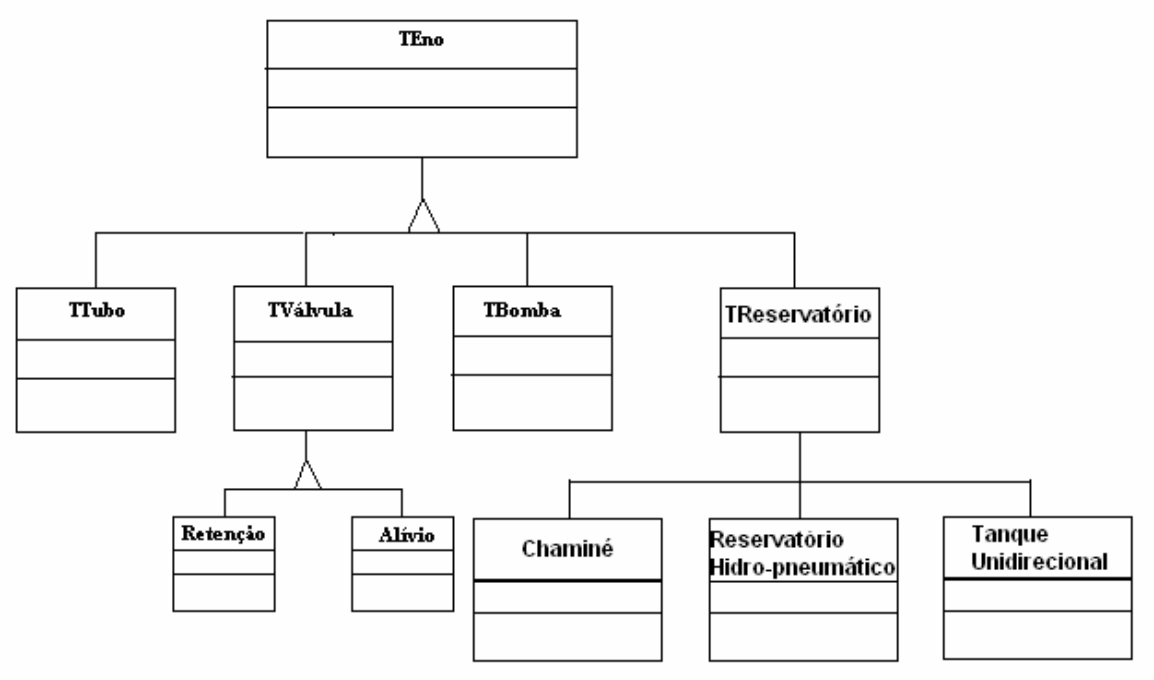

Figura 4.3 – Especialização de classes

Modelando a classe TTubo como subclasse de TEno, todas as características definidas para a classe TEno serão automaticamente herdadas pela classe TTubo. Assim, além dos atributos e métodos definidos especificamente na classe TEno, uma instância dessa classe terá também os atributos de um TTubo (comprimento, diâmetro) e poderá executar qualquer método definido pela classe TEno.

4.2.5 Polimorfismo

Observando a classe TEno na Figura 4.4, nessa classe temos o método Crie_Instancia. A classe TEno foi especializada em subclasses: TTubo, TVálvula, TBomba. Na classe TTubo temos também o método Crie_Instancia, que é implementado de uma forma diferente da implementada na classe TEno. O mesmo ocorre nas classes TVálvula, TBomba, no qual o método Crie_Instancia apresenta outra forma de implementação. Nas quatro classes temos a mesma tarefa: criar uma instância inicializando explicitamente os atributos. Entretanto, essa tarefa é implementada de formas diferentes nas várias classes. A esse conceito denominamos de polimorfismo.

Polimorfismo consiste em ter-se, para a mesma tarefa, varias formas de executá-la.

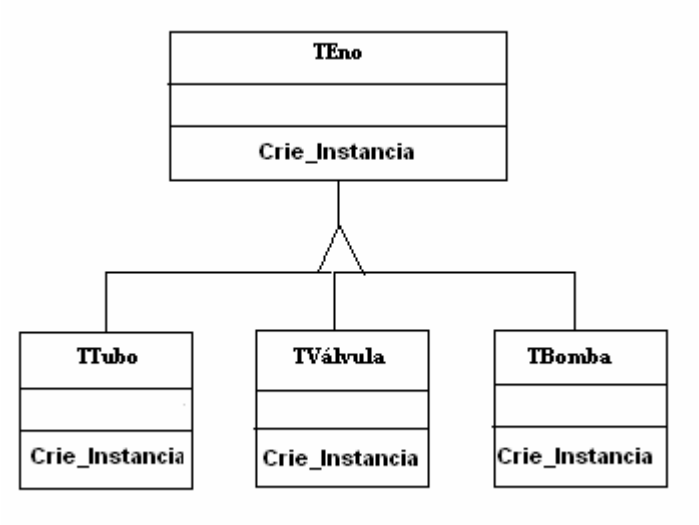

Figura 4.4 – Polimorfismo

4.2.6 Objetos

Os objetos se representam da mesma forma que uma classe. No compartimento superior aparece o nome do objeto junto com o nome da classe sublinhado, de acordo as seguintes sintaxes:

Nome_do_objeto: nome_da_classe

T1: Ttubo

Diâmetro: 0,5m
Comprimento: 1000 m
Nmontante: 1
Njusante: 2
Rugosidade: 0,001

Figura 4.5 - Exemplo do Objeto Tubo

4.2.7 Associações

As associações entre duas classes são representadas mediante uma linha que as une. A linha pode ter uma serie de elementos gráficos que expressam características particulares da associação.

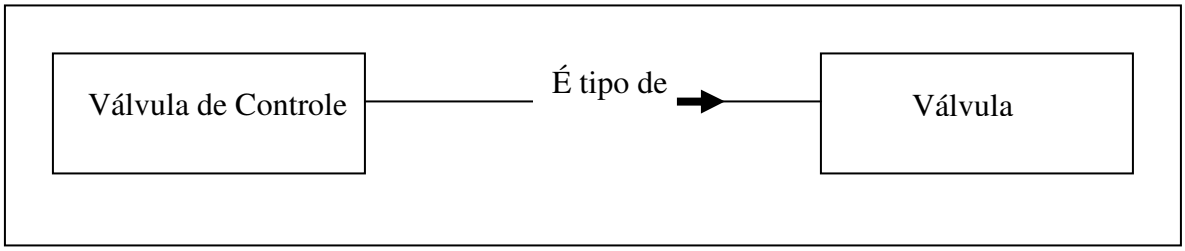

Figura 4.6 - Exemplo de associação com nome e endereço

O nome da associação é opcional e se mostra como um texto que esta próxima da linha, sendo adicionado um pequeno triângulo que indica direção na qual se pode ler "o nome da associação. No exemplo da Figura 4.6 se observa que válvula de controle é um tipo de objeto da classe válvula". Os nomes são incluídos para aumentar a legibilidade. Neste trabalho são suprimidos os nomes das associações consideradas como suficientemente conhecidas. Nas associações de tipo agregação e herança não se acostuma por o nome.

4.2.8 Implementação dos Objetos

Em termos de implementação, a identificação de um objeto constitui-se em uma posição de memória (por tanto uma variável) que armazena o endereço do espaço na memória ocupado pelo objeto. Seja, por exemplo, o conjunto de instruções a seguir:

```
VAR
No_Montante, No_Jusante : Node;
Begin
No_Montante := Node.CrieInstacia;
End;
```

Nesse exemplo, No_Montante e No_Jusante identificam dois objetos da classe Node No_Montante e No_Jusante são, por tanto, posições de memória que armazenarão os endereços da memória ocupada pelos dois objetos. Quando uma variável contém o endereço de um objeto, diz-se que faz referencia ao objeto.

4.3 Implementação de Métodos

A implementação de um serviço denomina-se método. Um método constitui-se em um conjunto de instruções que são executadas pelo computador, representando a ação de um objeto. Seja, por exemplo, a classe resultante da abstração de um tubo do mundo real. Podemos, de maneira simplificada, considerar os seguintes atributos para instâncias dessa classe: diâmetro, comprimento, rugosidade. Consideraremos esses atributos como simples valores, logo cada atributo constitui-se em uma variável. Estamos considerando aqui as características mais comuns de um tubo. Veja que na resolução de um problema teremos sempre um processo de abstração, e aqui não estamos definindo nenhum problema especifico. Para verificar as características de funcionamento da declaração de uma classe, analisaremos a seguir a implementação dos métodos especificados na classe Tubo da aplicação:

O método construtor da classe Tubo foi implementado como:

- Criar uma instância sem inicializar explicitamente seus atributos (construtor).

```
Unit Tubo;
Interface
Constructor CrieInstancia;

Implementation
Constructor TTubo.CrieInstancia;
 begin
 end;
```

- Criar uma instância inicializando explicitamente os atributos (construtor).

```
Unit Tubo:
Interface
Constructor CrieInicialize(Num1,in1,in2:integer;D1,L1,e1,A1,Om:real);

Implementation
Constructor TTubo.CrieInicialize(Num1,in1,in2:integer;D1,L1,e1,A1,Om:real);
 begin
 No_Montante:=in1;
 No_Jusante:=in2;
 Diametro:=D1;
 Comprimento:=L1;
 Rugosidade:=e1;
 Area:=A1;
 Dt:=0.5;
 end;
```

O método destrutor da classe Tubo foi implementado como:

- Destruir uma instância (destrutor).

```
Unit Tubo;
Interface
Destructor Destrua_se;

Implementation
Destructor TTubo.Destrua_se;
begin
end;
```

Em *object Pascal* um método comum pode ser implementado como *procedure* ou como *function* :

- Informar um valor de cada um dos atributos (um método para cada atributo).

```
Unit Tubo:
Interface
Procedure Pega_Diametro (vNum:integer);

Implementation
Procedure TTubo.Pega_Diametro(vNum:integer);
begin
Diametro :=vNum;
end;
```

O restante dos métodos da classe Tubo, podem ser implementados seguindo a mesma sintaxe:

- Atribuir novo valor (modificar) a cada um dos atributos (um método para cada atributo).
- Calcule Área.
- Calcule pontos interiores.
- Calcule os pontos exteriores.

4.4 Representação Gráfica da POO Usando UML

Existem empresas de "*software*" que desenvolveram pacotes que permitem gerar diagramas UML e administrar os mesmos num modelo. Os mais notáveis são *Rational Rose* e *Select Enterprise*. Neste trabalho foi usado o *Microsoft Office Visio Professional* 2003. Os principais modelos que se realizaram para a construção da aplicação são mostrados a seguir na ordem em que foram desenvolvidos.

4.4.1 O Diagrama de Casos de Uso

Representa graficamente os casos de uso que tem o sistema. Definem-se casos e uso como uma interação suposta com o sistema a desenvolver, no qual são apresentados os requisitos funcionais, mostrando o que um sistema tem que fazer. Na Figura 4.7 se apresenta os casos de

uso deste trabalho, em que se apresenta o ator (usuário) e as operações que pode realizar (funções).

Um modelo de casos de uso descreve os requisitos funcionais do sistema, os casos de uso representam a interação típica entre o usuário e um sistema informático, nao necesariamente especificam como se implementam as operações. Os casos de uso descrevem:

- O sistema
- O entorno do sistema
- A relação entre o sistema e seu entorno

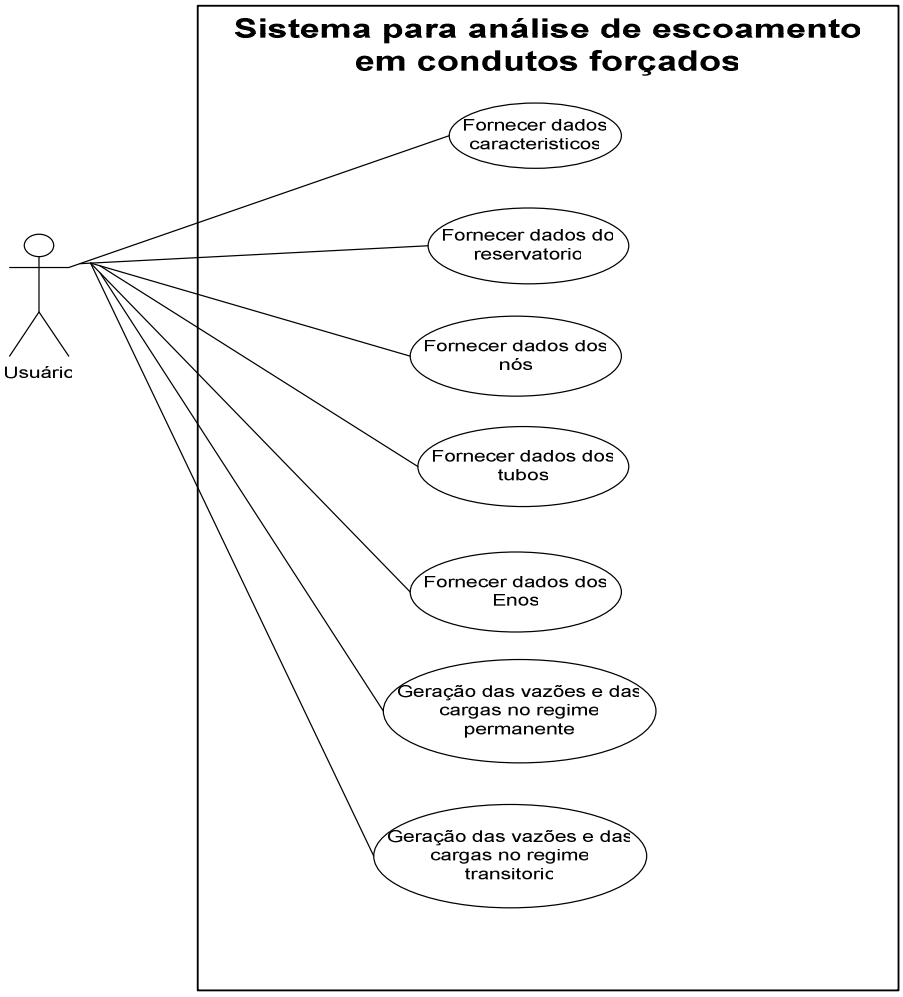

Figura 4.7 - Diagrama de Casos de Uso

4.4.2 O Diagrama de Classes

Apresenta um conjunto de classes, interfaces e suas relações. Sendo considerado o diagrama mais comum na hora de descrever o desenho dos sistemas orientados a objetos. Na Figura 4.8 são mostradas as classes globais, seus atributos e as relações de uma possível solução ao problema de escoamento de água. Os diagramas de classes apresentam um resumo do sistema em termos das suas classes e as relações entre elas. No geral, estes são diagramas estáticos que apresentam as interações.

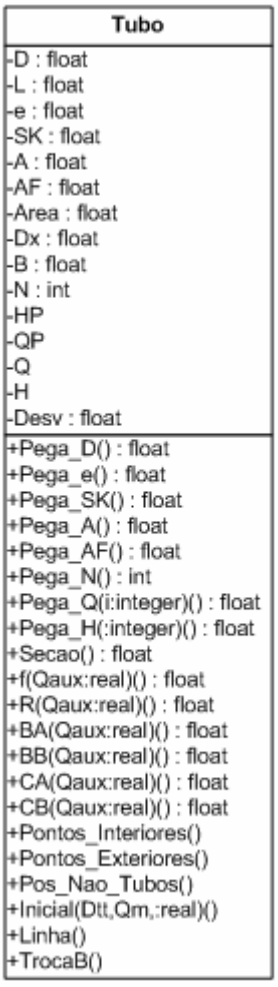

Figura 4.8 - Notação para Classe Tubo

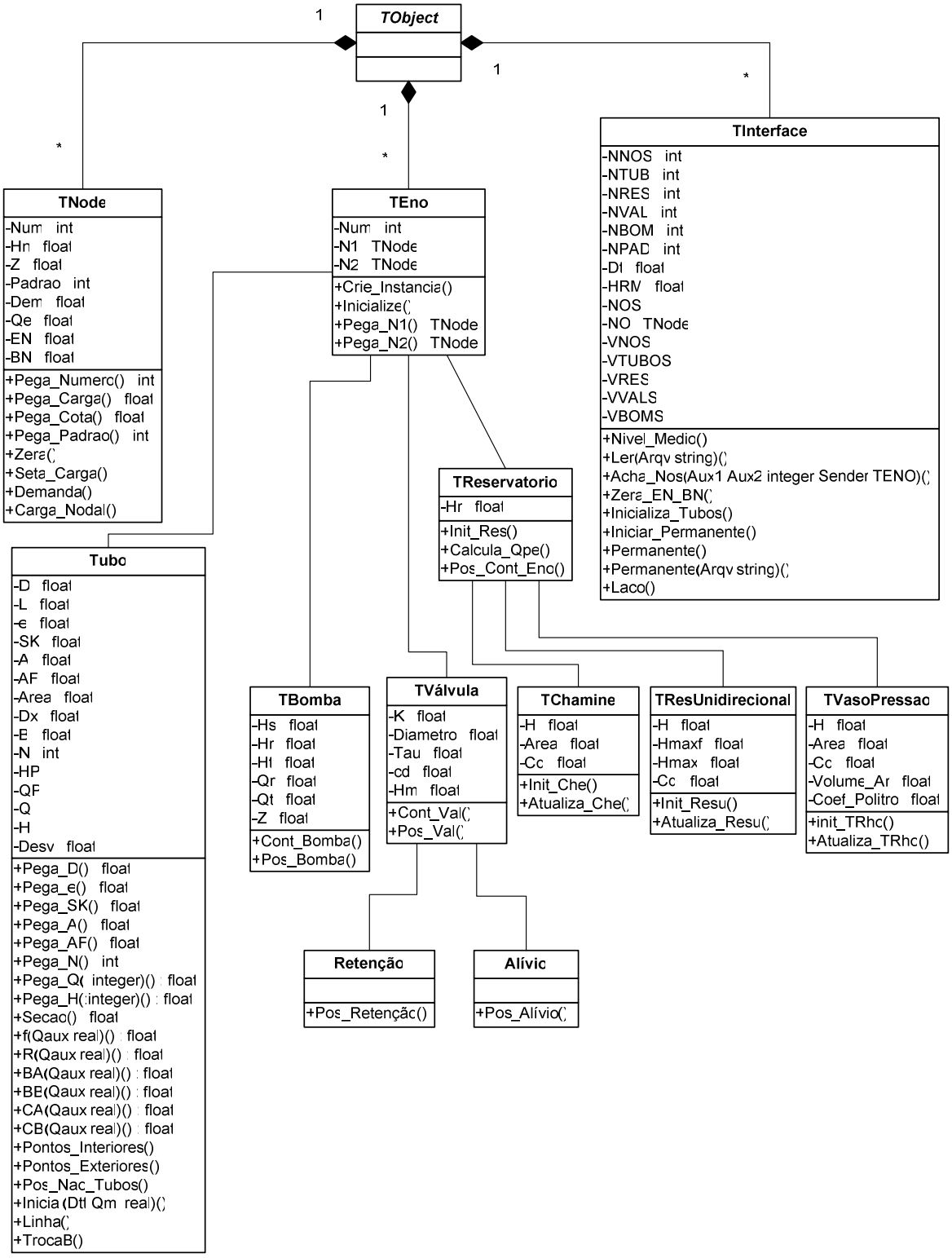

Figura 4.9 - Diagrama de Classes

4.4.3 O Diagrama de Seqüência

O diagrama de seqüência apresenta a interação dos objetos que compõem um sistema de forma temporal como mostrado na Figura 4.10.

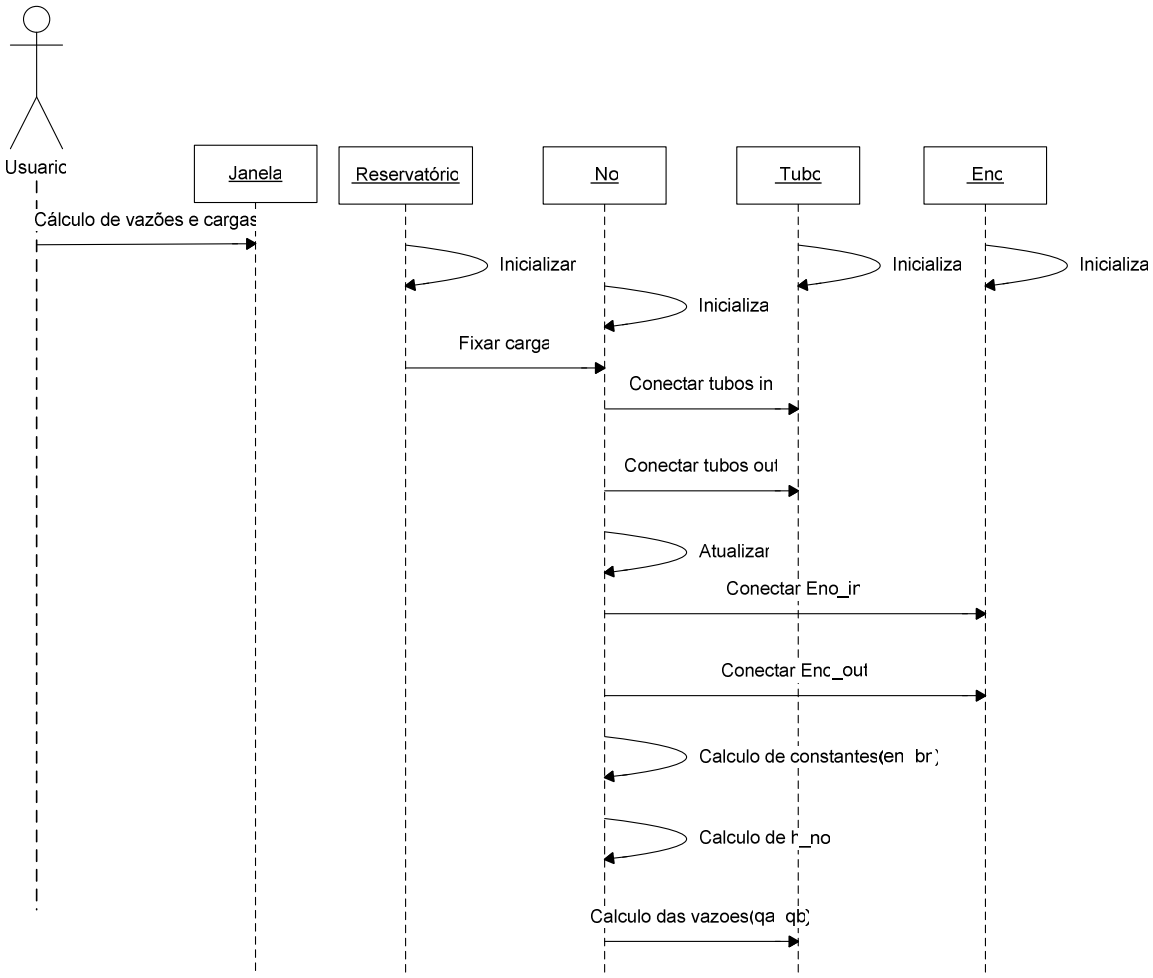

Figura 4.10 - Diagrama de Seqüência

4.4.4 O Diagrama de Estados

Os diagramas de estados apresentam os possíveis estados em que poderia encontrar-se um objeto e as transições que podem causar. O estado de um objeto depende da atividade que está executando ou de alguma outra condição.

As transições são linhas que unem os diferentes estados. Num estado em que o objeto esta pendente de algum tipo de validação que dependa de um processo em curso, não é necessário evento externo algum para que se produza a transição, esta acontecera quando terminar o processo, em função do resultado deste.

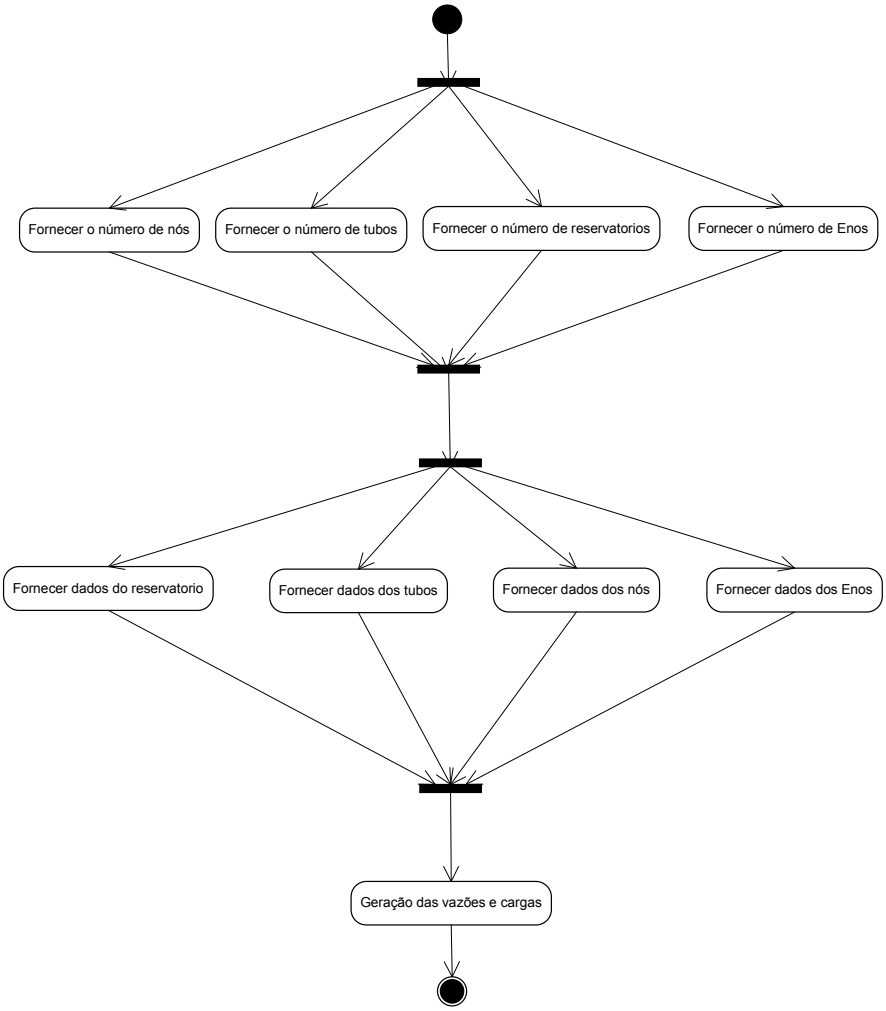

Figura 4.11 - Diagrama de Estados

4.5 Representação de Classes em *Object Pascal*

A construção do modelo para resolução de um problema em computador, bem como sua representação através de um programa, envolve uma série de informações.

Essas informações estão relacionadas tanto a conceitos de modelagem orientada a objetos quanto aos aspectos relativos à linguagem de programação utilizada na representação final do modelo. Analisaremos a seguir a declaração de uma classe verificando as características de funcionamento da declaração.

Unit Elemen**;**	***Unit*** básica construída pelo programador. Começo do bloco de declarações
interface	
uses Classes,math;	***Units*** que contém bibliotecas *Delphi* usadas.
const G = 9.81;	***const*** indica que os parâmetros são referências a variáveis não modificáveis definidas no entorno que invoque ao método {aceleração da gravidade}.
type PTNODE = ^TNode;	Começo de declarações de tipos de dados {ponteiro para objeto tipo *Node*}.
VETFloat = array of real;	
TNODE = class (TObject) **private**	Começo de declarações de tipos de classes Se define a nova classe, filha de *TObject*, classe raiz de todas as classes em *Delphi*. O nível de proteção *private* estabelece que só a classe e seus descendentes tenham acesso à lista protegida.
Num : integer; {Numero do Nó} HN : real; {Carga do Nó} Z : real; {Cota do Nó } Padrao: integer; {Número da curva padrão} Dem : real; {Demanda média do nó} Qe : real; {vazão de elemento não tubo} EN : real; {EN para o MOC} BN : real; {BN para o MOC}	
public	O restante do código tem acesso público (sem restrição)

function Pega_Numero : integer; function Pega_Carga : real; function Pega_Cota : real; function Pega_Padrao : integer;	***Function*** pressupõe o retorno de um resultado (valor ou objeto) no nome da função. Assim, o nome de uma função é declarado como sendo de um determinado tipo ou classe, pois nesse nome deverá retornar uma resposta resultante da execução do respectivo método.
Procedure Zera; procedure Seta_Carga(HM : real); procedure Demanda; procedure Carga_Nodal;	Tal qual uma *function*, um método implementado como *procedure*, apresenta parâmetros. Entretanto, o nome de uma função retornará um resultado, o que não acontece com um *procedure*.
end;	Fim da declaração da classe.

4.6 Aporte da POO na Solução do Problema Hidráulico

Sendo a característica fundamental do computador a de ser uma maquina programável que pode ser usada na resolução dos problemas hidráulicos. O computador resolve determinado problema através da execução de um programa, denominado software. Assim a construção de um programa para resolução de um problema hidráulico implicara na definição de um modelo de resolução.

A construção desse modelo pode ser feita sob vários paradigmas. Nesta dissertação o processo de modelagem foi feito tendo-se como objetivo fundamental a POO abordando a resolução do problema hidráulico através de uma construção que represente como as coisas acontecem no mundo real, constituído por unidades chamadas de objetos.

A POO mostrou-se mais amigável no desenvolvimento de modelos que envolvem grupos de pesquisa, sendo que cada pesquisador pode desenvolver os objetos por separado (tubos, válvulas, reservatórios, etc.), não precisando torná-los dependentes os uns aos outros.

A POO facilita o entendimento do problema hidráulico, sendo que o modelo e a representação de objetos que interagem entre si, mostrando, da melhor forma possível, como as coisas acontecem no mundo real.

As vantagens da POO sobre outros paradigmas computacionais tais como, re-uso de código, herança, polimorfismo, encapsulamento, as quais são mostradas no texto anterior.

Capítulo 5

Estudo de Casos

5.1 Introdução

Neste capítulo são apresentados alguns dos exemplos que foram analisados com o programa desenvolvido usando a programação orientada a objetos (POO) e o método das características (MOC). Visando ilustrar a aplicação dos conceitos e esquemas de cálculo apresentados nos capítulos anteriores, os exemplos mencionados a seguir foram também analisados pelo programa EPANET (em regime permanente), para que os resultados obtidos nesta dissertação de Mestrado tivessem um parâmetro de comparação e, a partir dessas comparações, se necessário, efetuar correções no programa proposto. Alguns exemplos típicos de instalações hidráulicas serão analisados e discutidos no decorrer deste capítulo, considerando as manobras significativas que acarretam as condições transientes.

5.2 Exemplos de Aplicação

5.2.1 Exemplo 1: Fechamento Instantâneo da Válvula num Sistema de Distribuição de Água

O exemplo 1 proposto visa verificar o comportamento do programa desenvolvido neste trabalho, resolvendo o regime não permanente da instalação esquematizada na Figura 5.1 que foi apresentado por KARNEY e MCINNIS (1990).

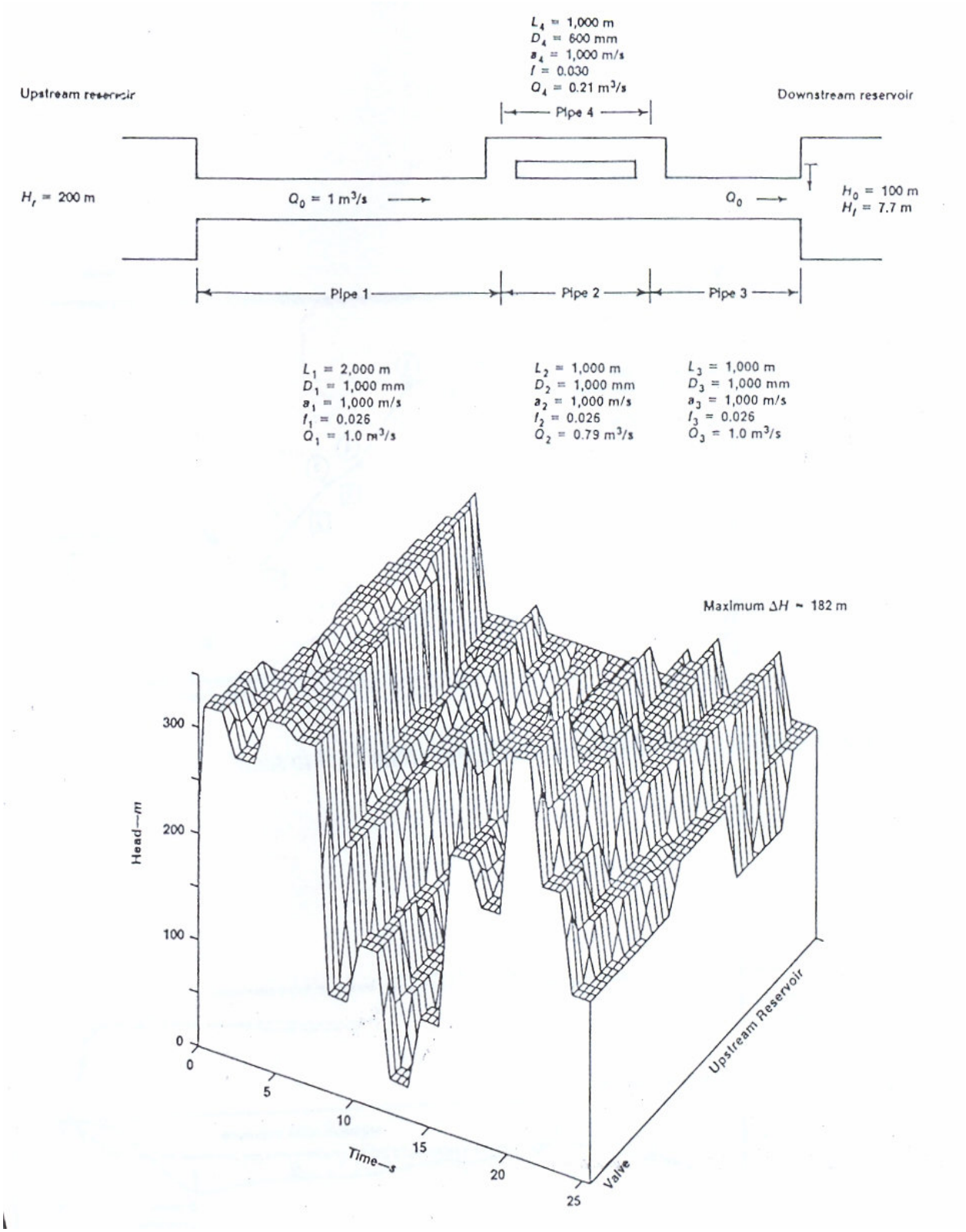

Figura 5.1 – Exemplo1 - Esquema Físico e Resposta a um Fechamento Instantâneo da Válvula

A Figura 5.1 apresenta um esquema da instalação. O sistema é composto por quatro tubulações e quatro nós, com dois reservatórios de nível constante e uma válvula a jusante do tubo três, no qual uma manobra de fechamento instantâneo da válvula é efetuada.

O gráfico da Figura 5.1 é o resultado obtido por KARNEY e MCINNIS (1990), decorrente da manobra da válvula na seção a jusante do tubo três. Na Figura 5.2 apresenta o resultado obtido com o programa desenvolvido usando os conceitos de POO descritos neste texto.

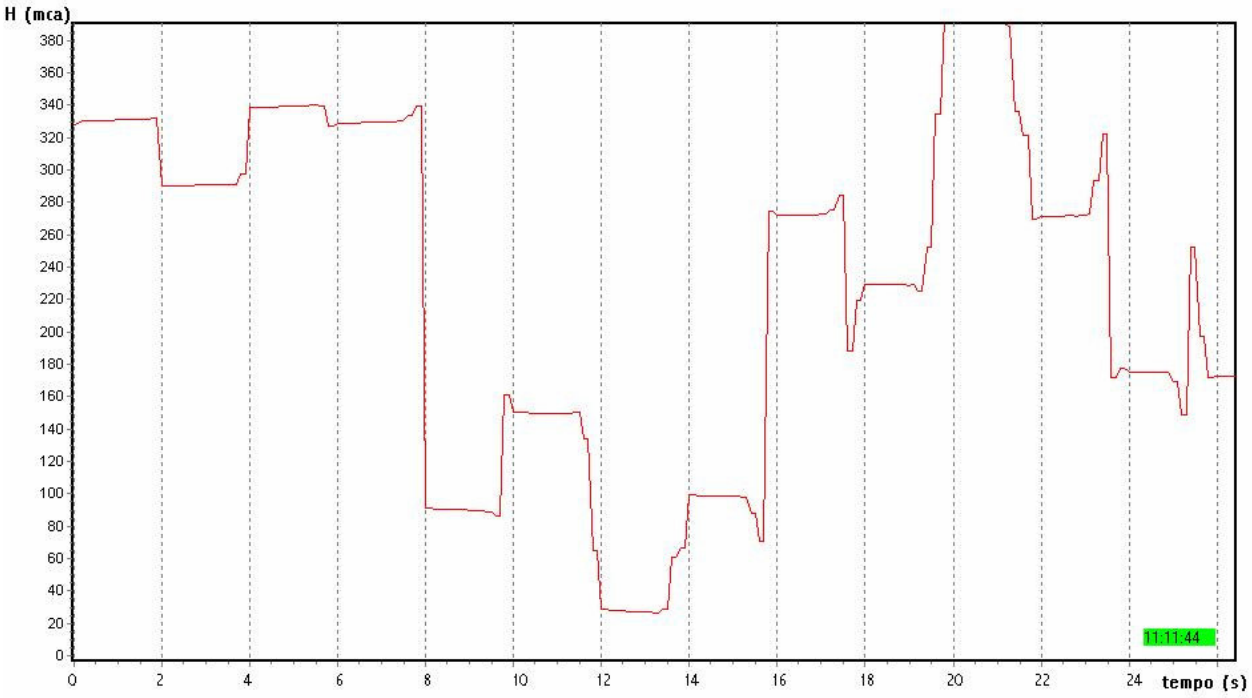

Figura 5.2 – Exemplo1 - Resposta a um Fechamento Instantâneo da Válvula

Observa-se da comparação das Figuras (5.1) e (5.2) que as respostas obtidas utilizando o programa desenvolvido neste trabalho (Figura 5.2), apresenta picos de onda os quais são devidos à topologia adotada neste trabalho, sendo que foi adicionado um trecho de tubulação a jusante da válvula o mesmo que gera a deflexão de onda.

Observa-se, também, que as respostas obtidas pelo programa desenvolvido são aceitáveis.

5.2.2 Exemplo 2: Regime Transitório Aplicado na Chaminé de Equilíbrio

O esquema físico da instalação em regime transitório aplicada na chaminé de equilíbrio proposta no exemplo 2 encontra-se indicado na Figura 5.3 e a codificação correspondente mostrada na Figura 5.4. Cabe-se mencionar que na topologia não são consideradas demandas nodais, enquanto que as características das tubulações são dadas conforme a Tabela 5.1.

Tabela 5.1 – Características das Tubulações

Tubo N°	**Comprimento (m)**	**Diâmetro (m)**	**Rugosidade (mm)**	**Celeridade (m/s^2)**	**Número de Seções**
1	500	0,5	0.001	1000	6
2	500	0,5	0.001	1000	6
3	1000	0,5	0.001	1000	11

As características da válvula empregada são:

- Diâmetro (m) = 0,3
- Kv (adi.) = 1
- Δt (adi.) = 0,1

A chaminé de equilíbrio instalada tem as seguintes características:

- Cota da Chaminé (m) = 95
- Área da Chaminé (m^2) = 3,14

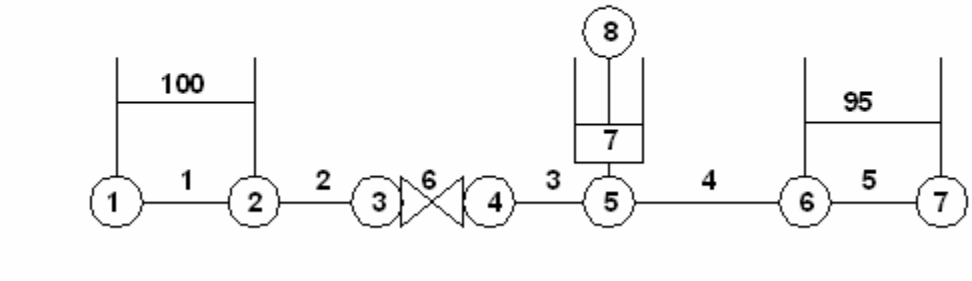

Figura 5.3 – Exemplo 2 - Esquema Físico

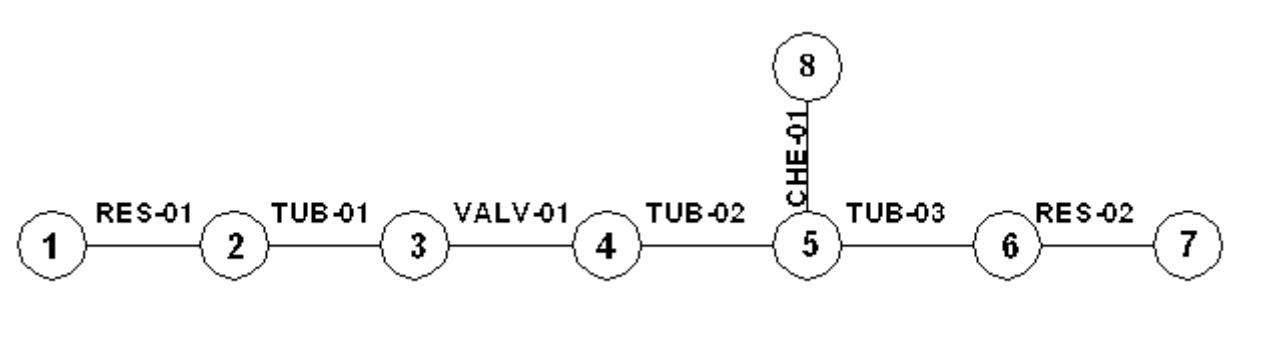

Figura 5.4 – Exemplo 2 – Modelagem e Codificação

A manobra simulada para a análise dos transientes é admitida supondo que há abertura instantânea da válvula seguida de um fechamento da mesma. Os resultados do transiente são mostrados nas Figuras 5.5 a 5.8. Na Figura 5.5 observa-se que após alcançar 200 segundos com uma pressão de 95 mca há uma elevação abrupta da pressão devido a manobra de abertura da válvula, equilibrando em 97.40 mca e diminuindo a pressão ate 93.8 mca devido a fechamento de válvula, seguida pela oscilação ate alcançar o equilíbrio em 95 mca após o qual houve um incremento da pressão por abertura da válvula que finalmente estabilizara num regime permanente 97.40 mca.

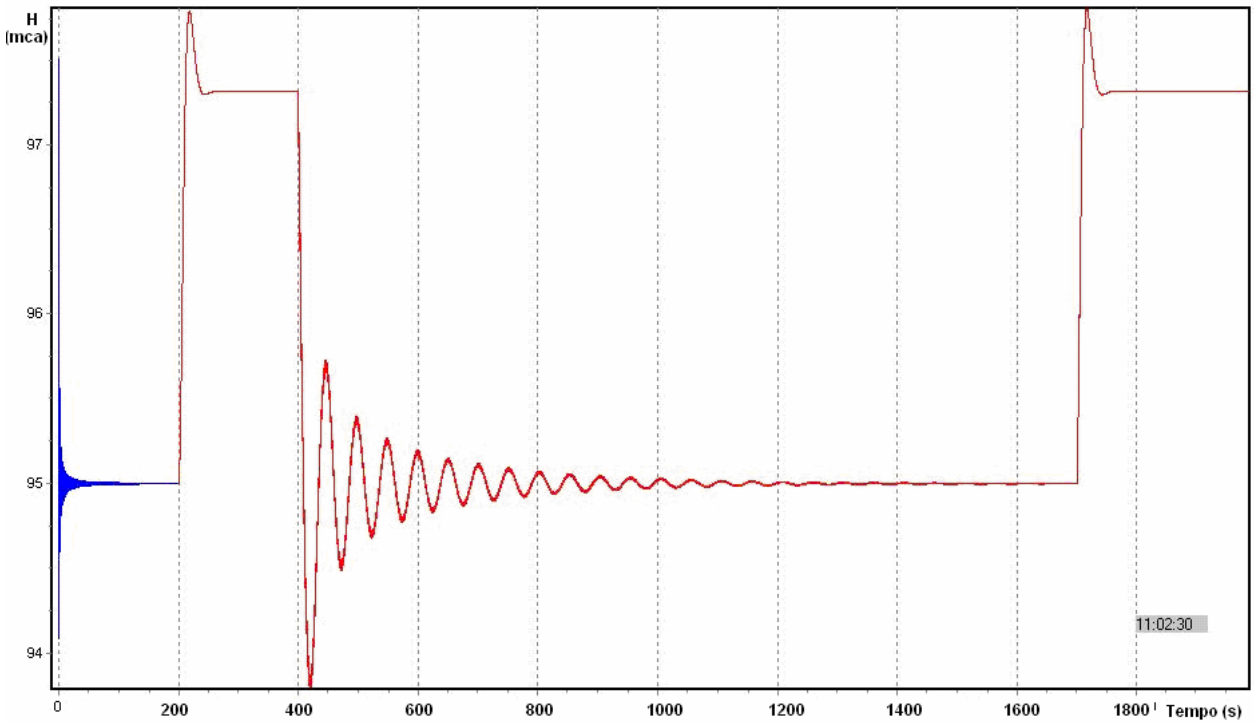

Figura 5.5 – Exemplo 2 – Transitório na Chaminé de Equilíbrio

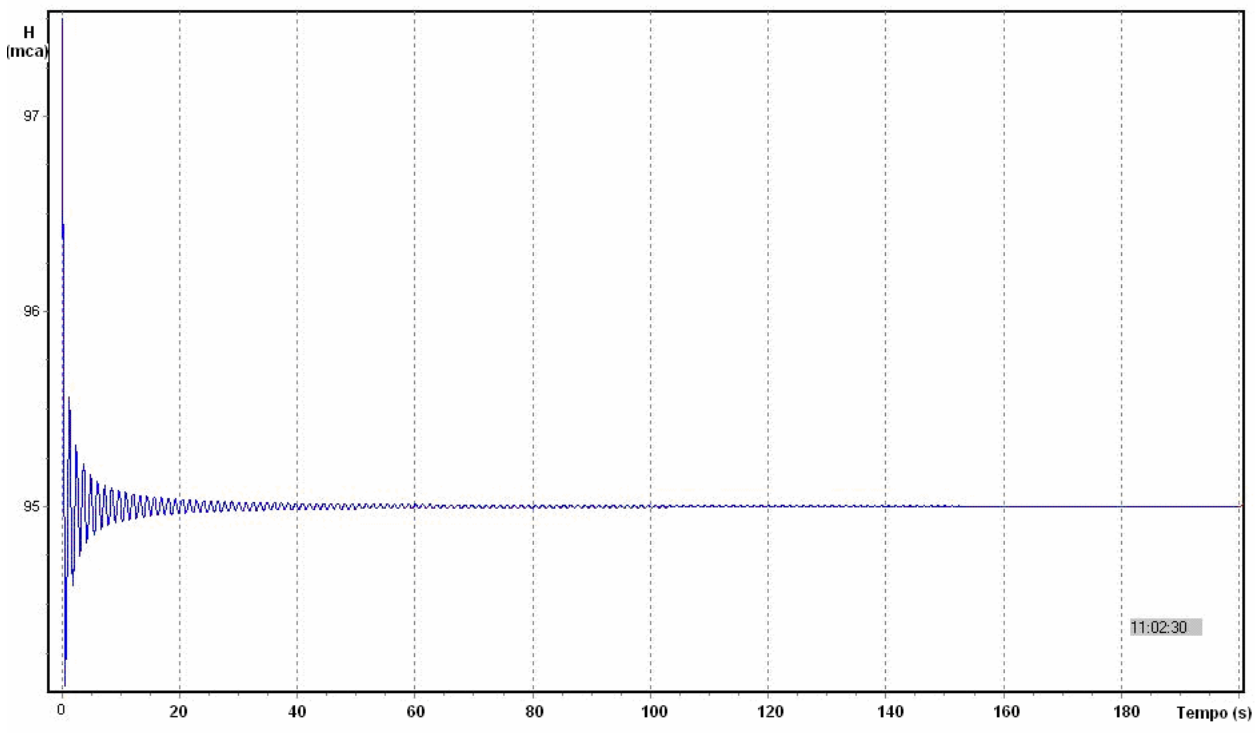

Figura 5.6 – Exemplo 2 – Análise do Permanente

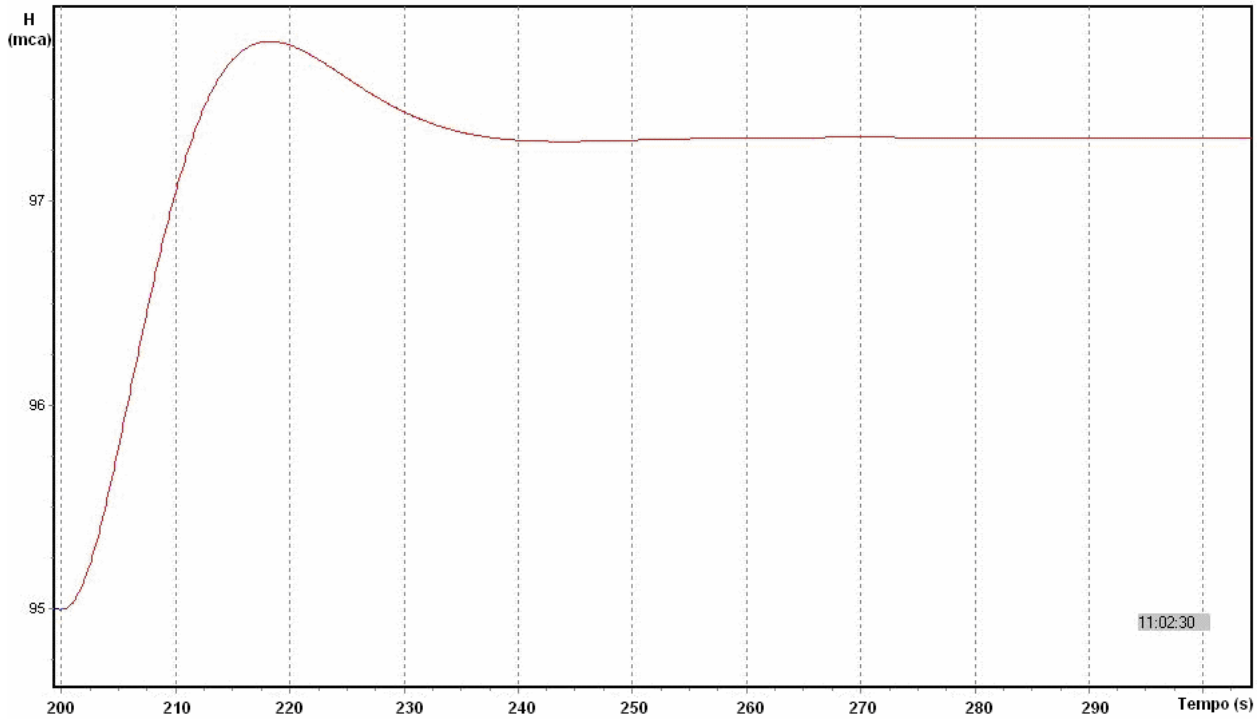

Figura 5.7 – Exemplo 2 – Abertura Instantânea da Válvula

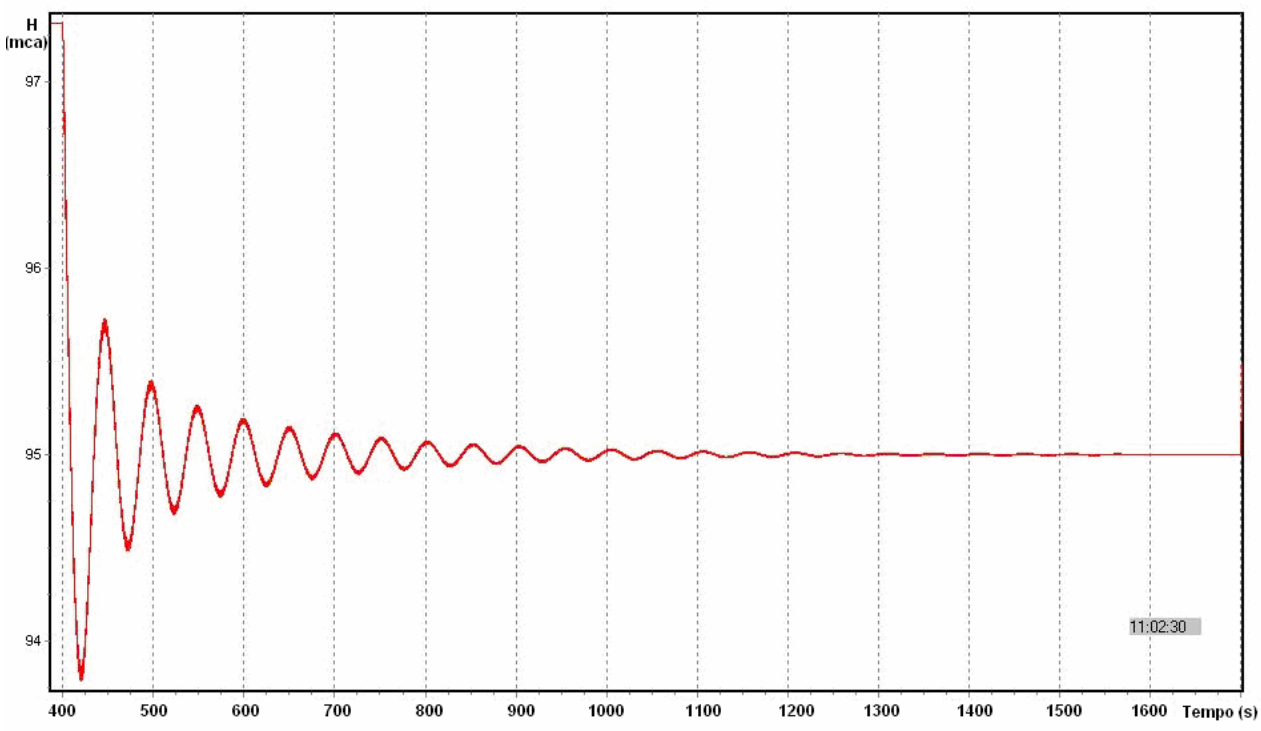

Figura 5.8 – Exemplo 2 – Fechamento Instantâneo da Válvula

5.2.2 Exemplo 3: Regime Transitório no Reservatório Unidirecional

Este exemplo é análogo ao Exemplo 2 sendo que, o dispositivo de controle usado neste caso é um reservatório unidirecional.

Considerando-se as mesmas características da válvula, as características do Reservatório Unidirecional são mencionadas a seguir:

- Cota do Reservatório Unidirecional (m) = 95
- Área do Reservatório Unidirecional (m^2) = 3,14
- Δt (adi.) = 0,1

As Figuras 5.9 e 5.10 mostram o esquema da instalação a modelagem e codificação, respectivamente. Comparando com as Figuras 5.3 e 5.4, somente se modificou o dispositivo de segurança neste caso se usou o Reservatório Unidirecional.

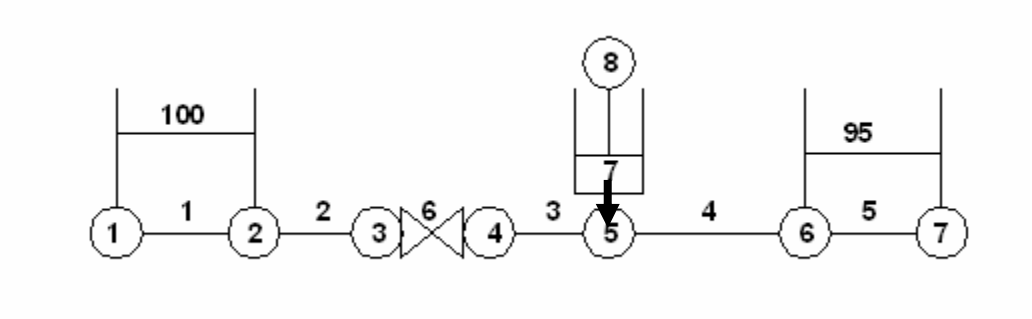

Figura 5.9 – Exemplo 3 - Esquema Físico

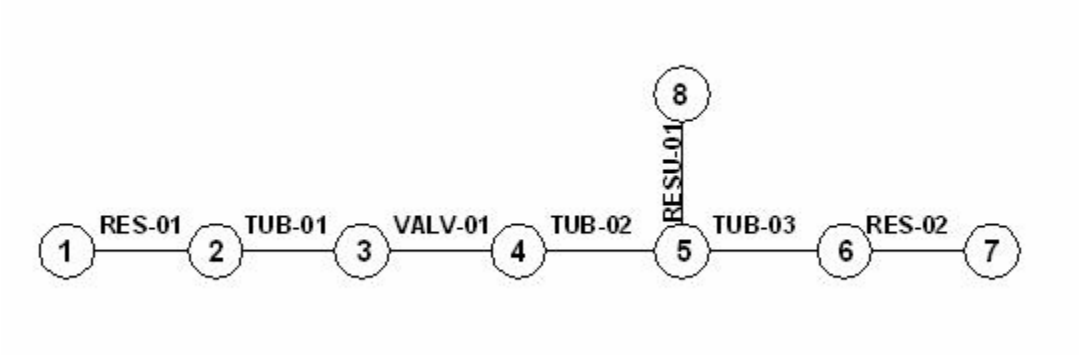

Figura 5.10 – Exemplo 3 – Modelagem e Codificação

Sendo as manobras simuladas análogas ao exemplo 2, as Figuras 5.11 a 5.14 apresentam os resultados do transiente.

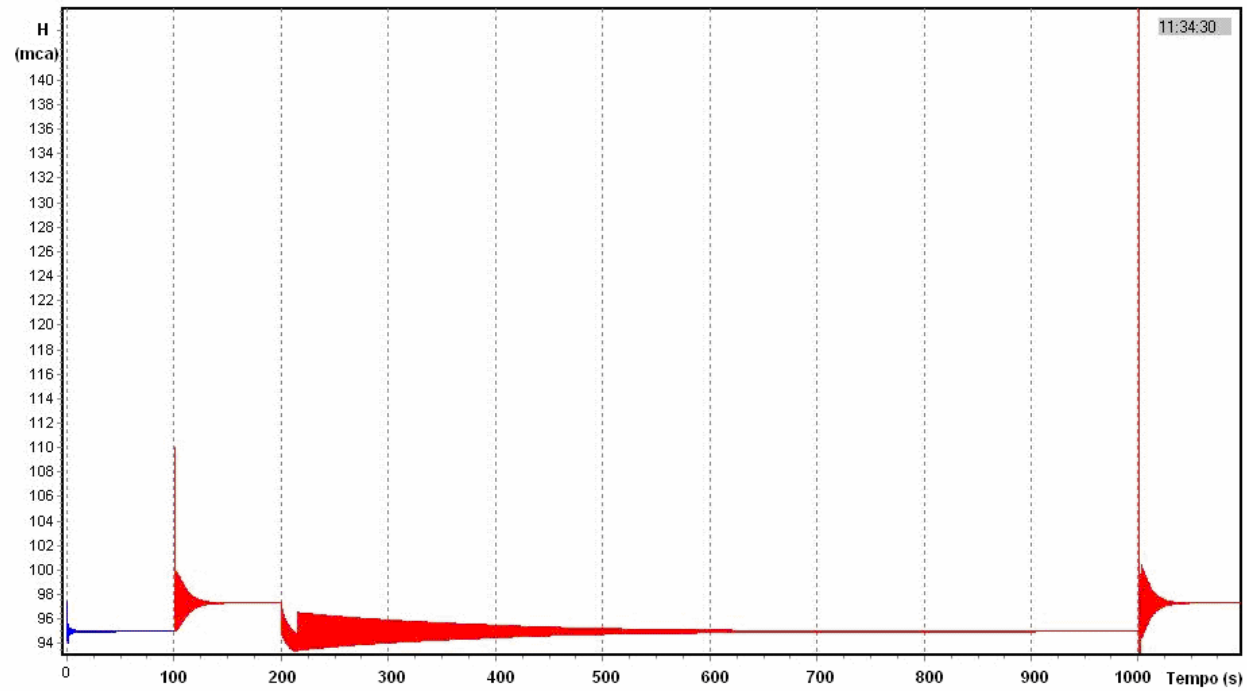

Figura 5.11 – Exemplo 3 – Transitório no Reservatório Unidirecional

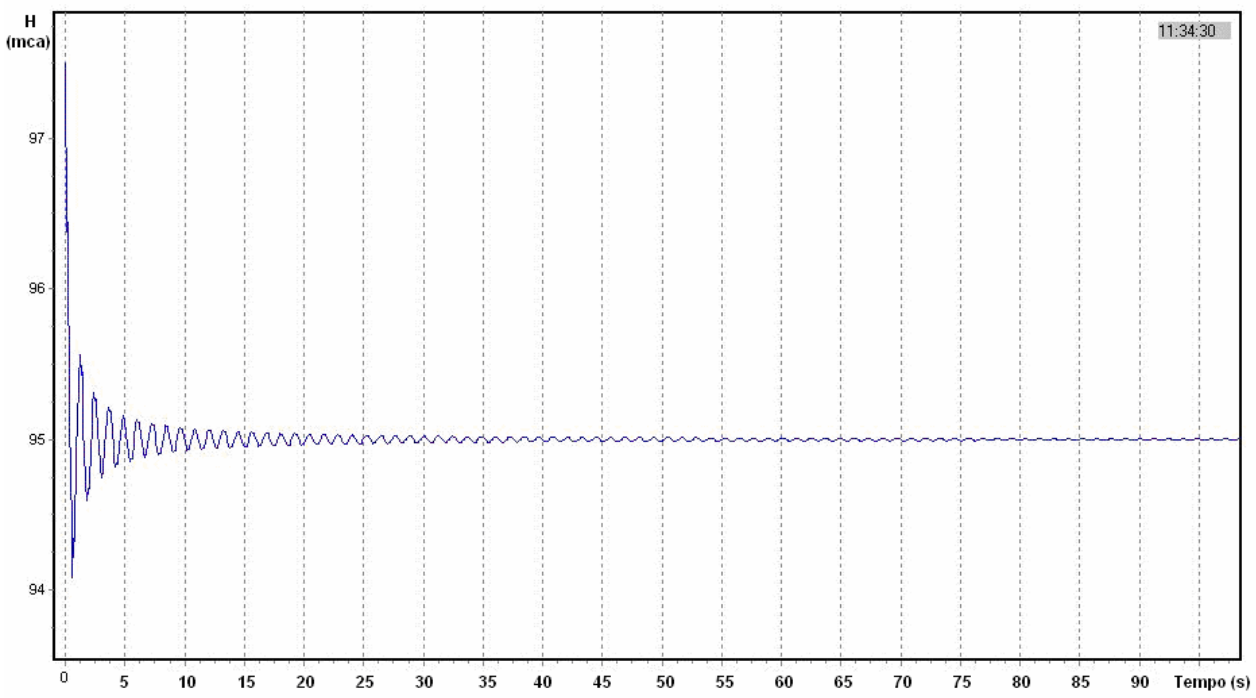

Figura 5.12 – Exemplo 3 – Análise do Permanente

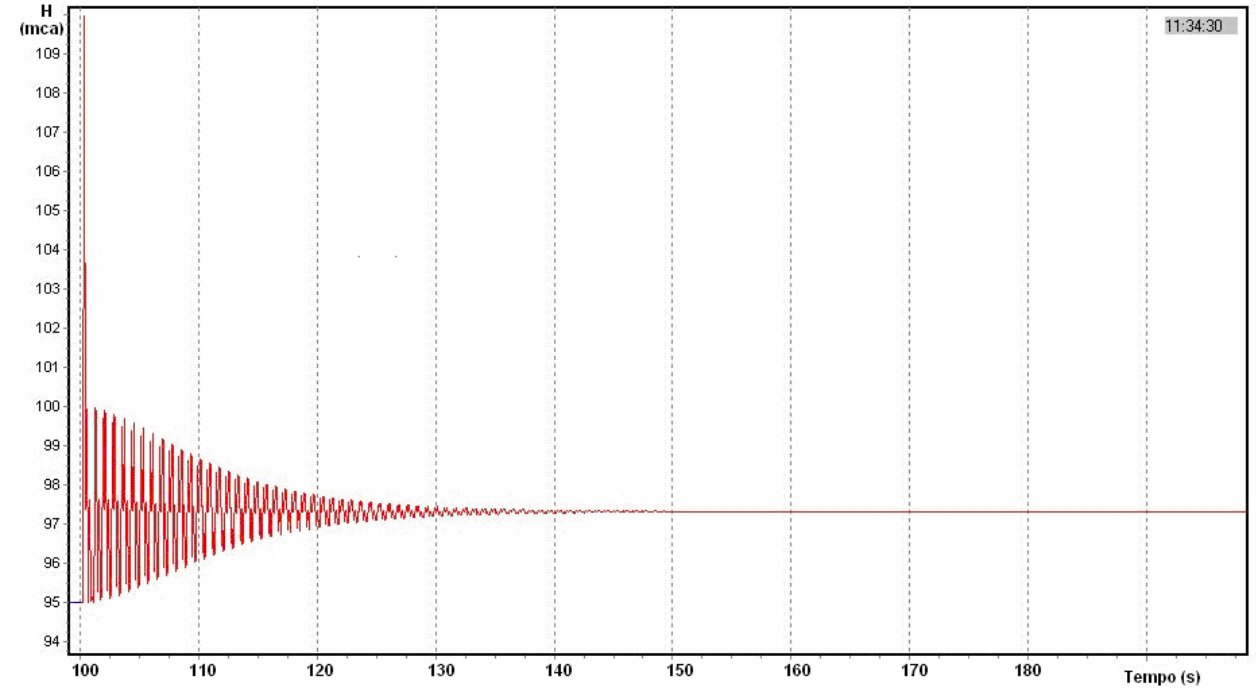

Figura 5.13 – Exemplo 3 – Abertura Instantânea da Válvula

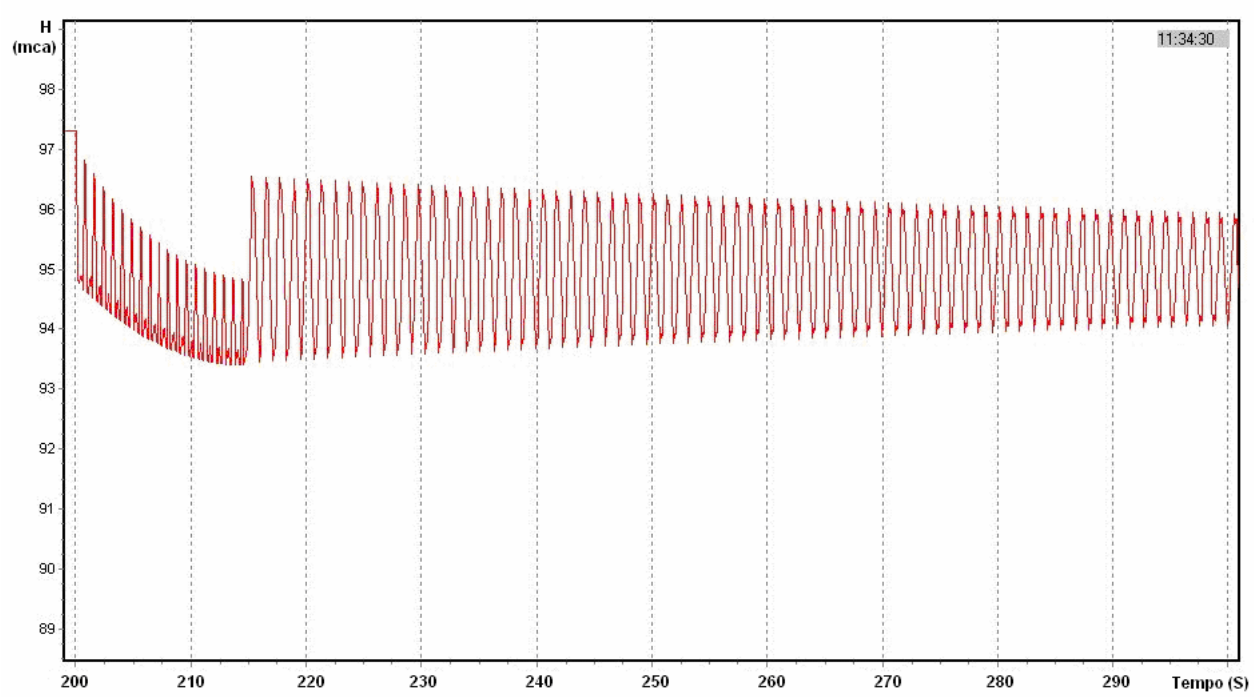

Figura 5.14 – Exemplo 3 – Fechamento Instantâneo da Válvula

5.2.3 Exemplo 4: Transitório no Vaso de Pressão

Este exemplo e análogo aos Exemplos anteriores sendo que, o dispositivo de controle usado neste caso é um Vaso de Pressão.

Considerando-se as mesmas características da válvula, o Vaso de Pressão apresenta as seguintes características:

- Nível da água (mca) = 94
- Área do Vaso de Pressão (m^2) = 3,14
- Volume de ar (m^3) = 3,14
- Coeficiente poli tropico (adi.) = 0,15
- Δt (adi.) = 0,1

As Figuras 5.15 e 5.16 mostram o esquema físico e a modelagem e codificação, respectivamente. Comparando com as Figuras 5.3 e 5.4, somente se modificou o dispositivo de segurança neste caso se usou um Vaso de Pressão.

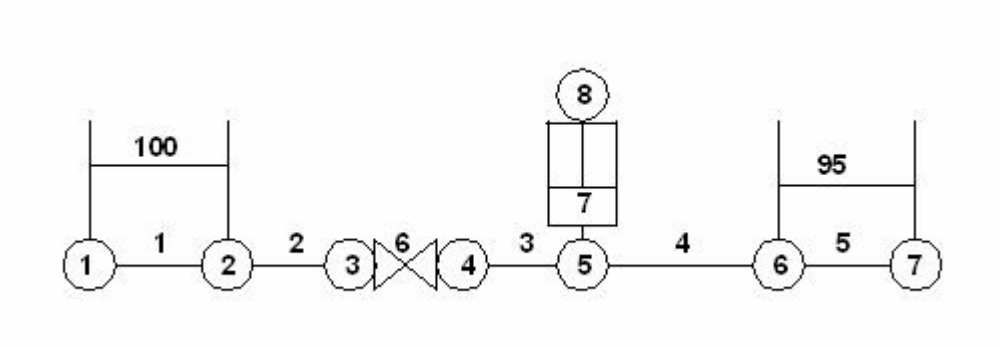

Figura 5.15 – Exemplo 4 - Esquema Físico

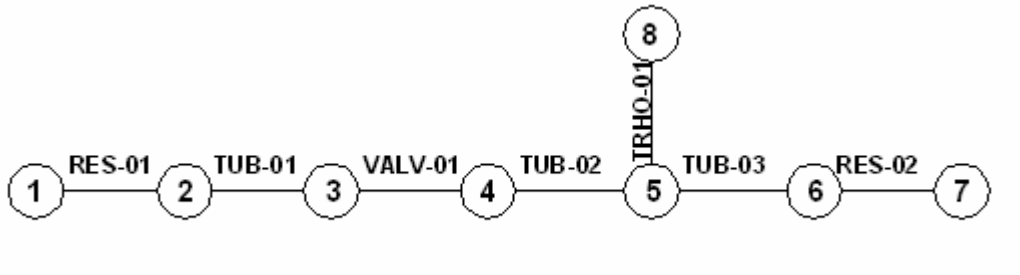

Figura 5.16 – Exemplo 4 – Modelagem e Codificação

Sendo as manobras simuladas análoga aos exemplos anteriormente apresentados, os resultados do transiente no vaso de pressão são mostrados nas Figuras 5.17 a 5.20.

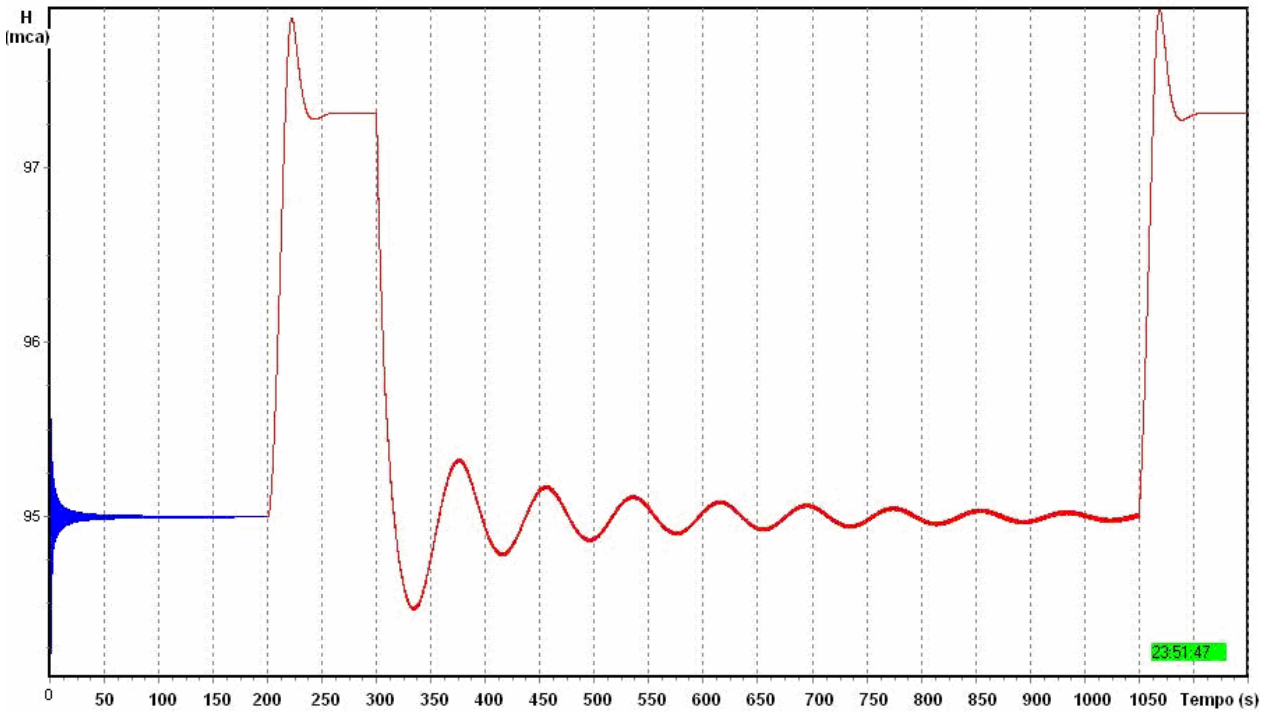

Figura 5.17 – Exemplo 4 – Transitório no Vaso de Pressão

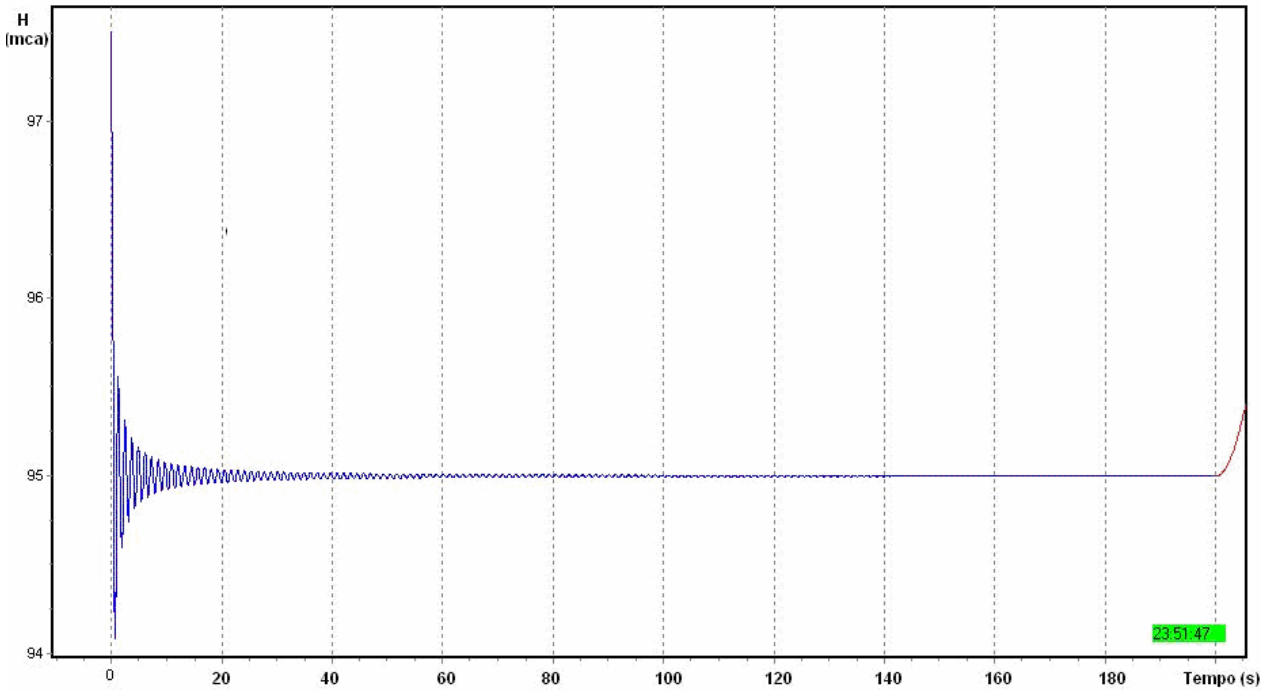

Figura 5.18 – Exemplo 4 – Análise do Permanente

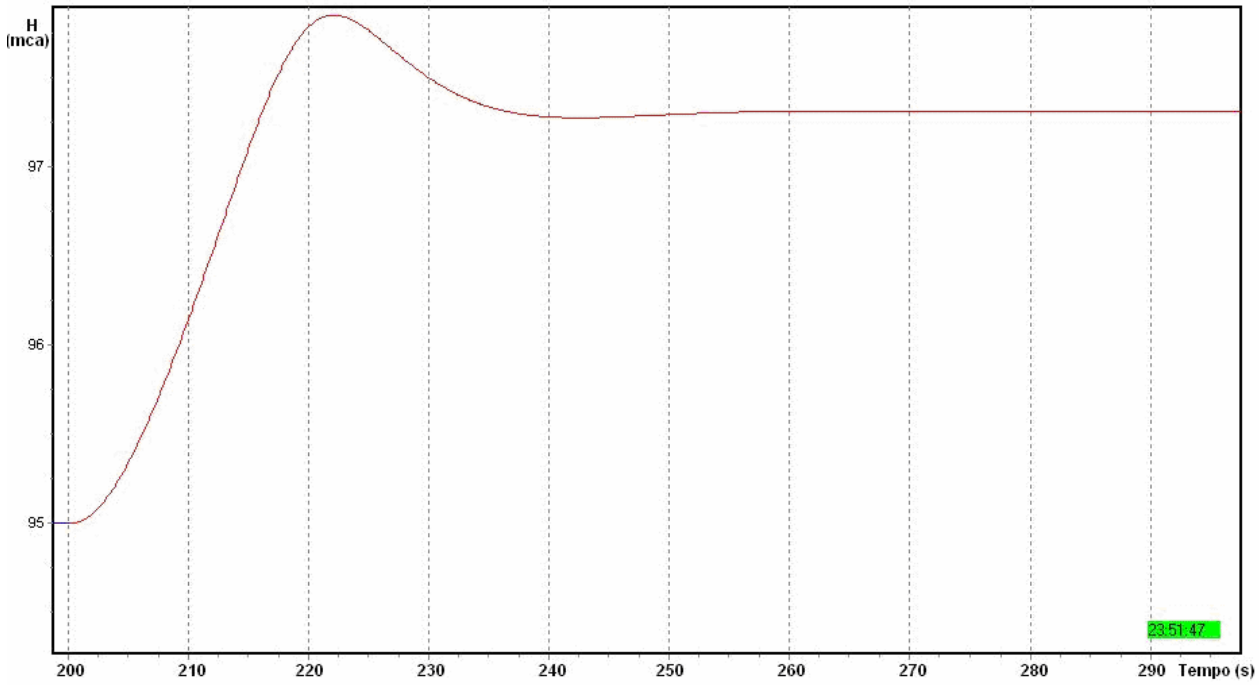

Figura 5.19 – Exemplo 4 – Abertura Instantânea da Válvula

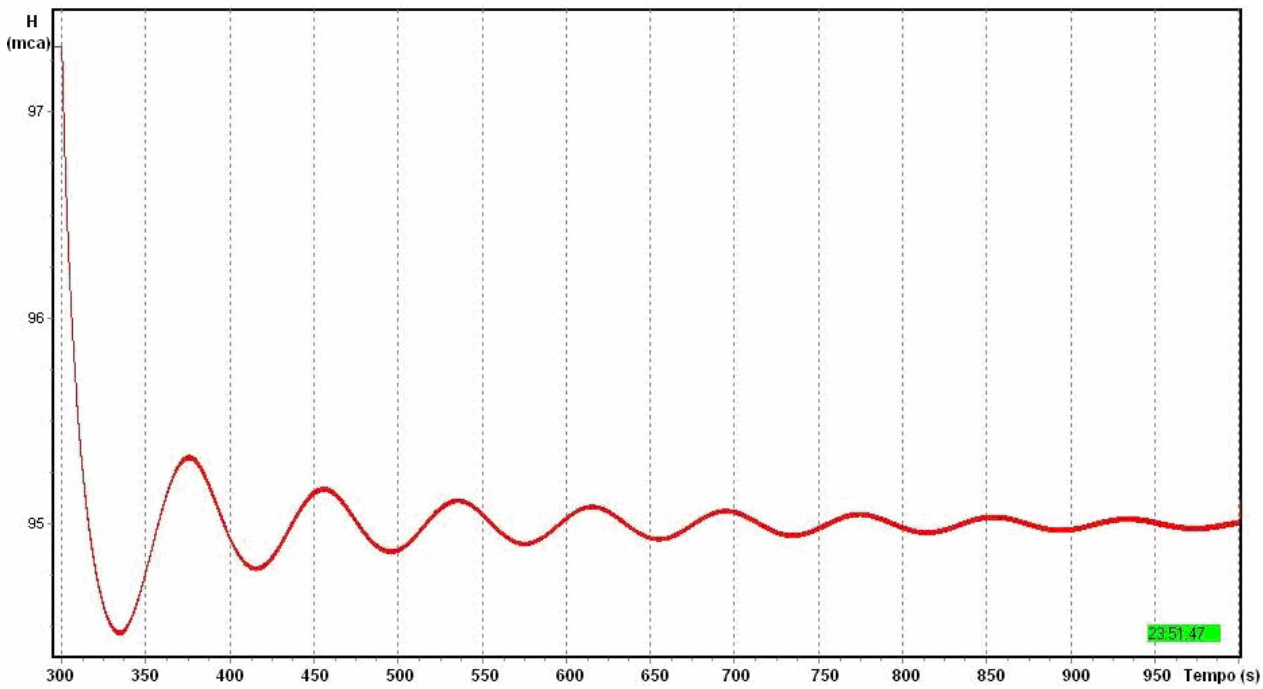

Figura 5.20 – Exemplo 4 – Fechamento Instantâneo da Válvula

5.2.4 Comparação dos Dispositivos de Controle Simulados neste Trabalho

Para discutir os efeitos dos dispositivos de controle utilizados para simulação de transitórios nesta dissertação é apresentada a Figura 5.21, sendo que os dispositivos de controle encontram-se sob as mesmas condições.

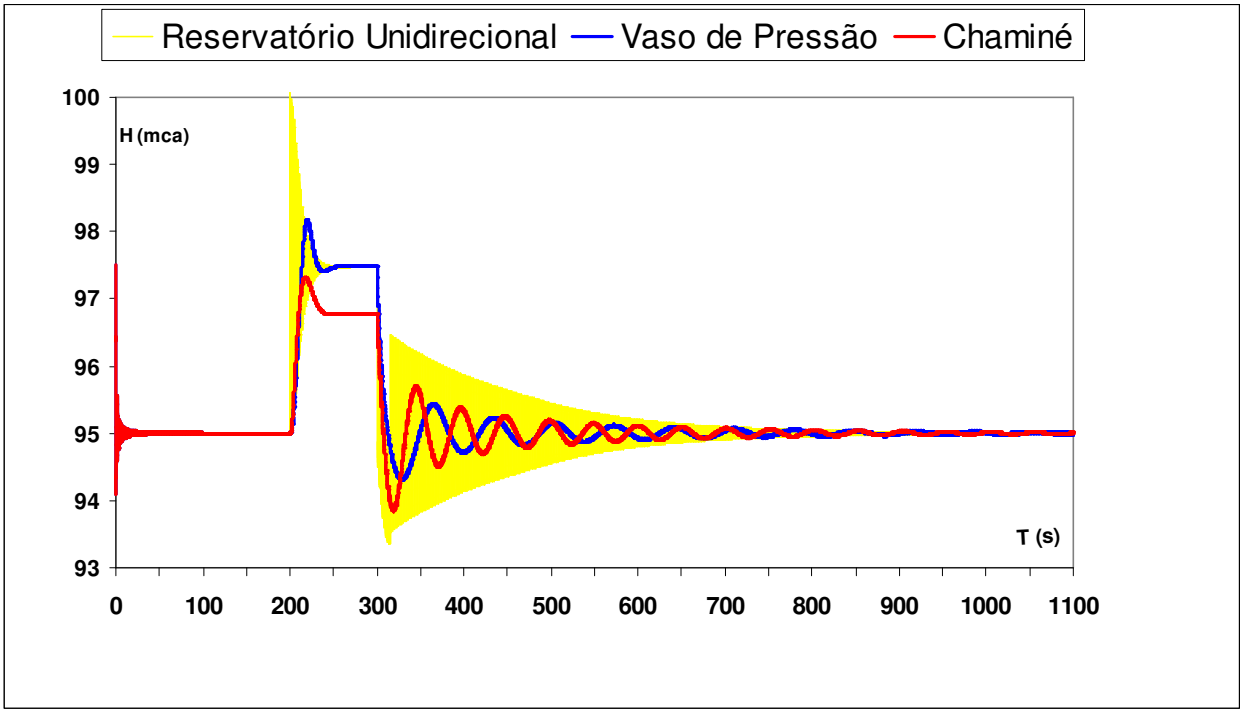

Figura 5.21 – Comparação do Comportamento dos Dispositivos de Controle

Na Figura 5.21 apresenta-se os resultados da análise das simulações tanto em regime permanente como em regime transitório usando os dispositivos de controle, sendo analisadas as cargas nos Nós de montante dos dispositivos: reservatório unidirecional, chaminé de equilíbrio e vaso de pressão.

A Figura 5.21 mostra que nos primeiros 200 segundos é atingido o regime permanente em 95 m.c.a, sendo que as condições iniciais são as mesmas para os dispositivos de controle simulados o regime permanente é o mesmo para os três casos. Após esse intervalo de tempo é simulada uma manobra de abertura instantânea da válvula mostrando que o reservatório unidirecional atinge o maior pico de carga, a seguir vemos que nos 300 segundos é atingido um novo valor de regime permanente. Seguidamente é simulada uma manobra de fechamento instantâneo de válvula voltando às condições iniciais do problema, atingindo o valor inicial de carga 95 m.c.a.

5.2.5 Exemplo 5: Transitório com manobra na Bomba

Na topologia não são consideradas as demandas nodais. As características das tubulações são dadas conforme a Tabela 5.2.

Tabela 5.2 – Características das Tubulações

Tubo Nº	Comprimento (m)	Diâmetro (m)	Rugosidade (mm)	Celeridade (m/s^2)	Número de Seções
1	4000	0,5	0,001	1000	41
2	1000	0,3	0,001	1000	11
3	1000	0,3	0,001	1000	11
4	1000	0,3	0,001	1000	11

As características da válvula empregada são semelhantes aos exemplos anteriores. Sendo as características da Bomba instalada as seguintes:

- Inércia da Bomba ($kg.m^2$) = 18
- Rotação da Bomba (r.p.m) = 140
- Rendimento da máquina (adi.) = 0.80
- Δt (adi.) = 0,1

A curva da bomba adotada é mostrada na Figura 5.22.

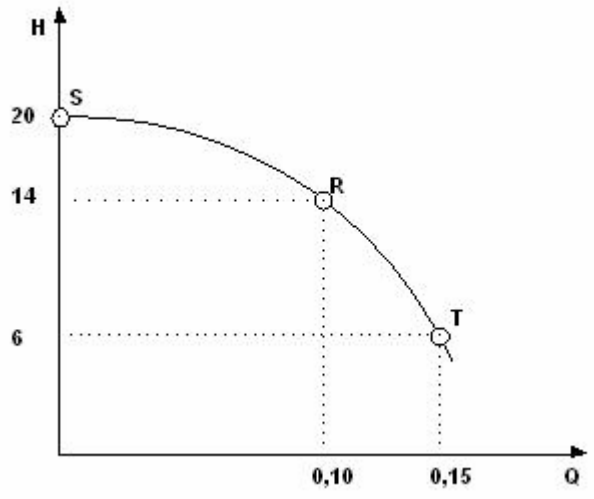

Figura 5.22 – Exemplo 5 – Curva Carga (H) x Vazão (Q).

O esquema físico da instalação proposta para o exemplo 5 esta indicado na Figura 5.23 e a codificação correspondente é mostrada na Figura 5.24.

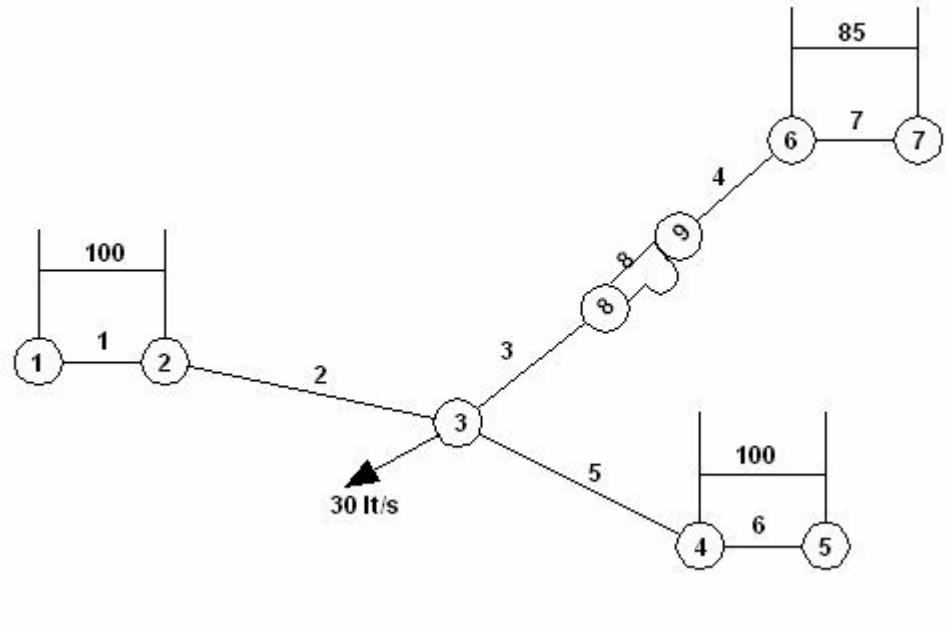

Figura 5.23 – Exemplo 5 - Esquema Físico

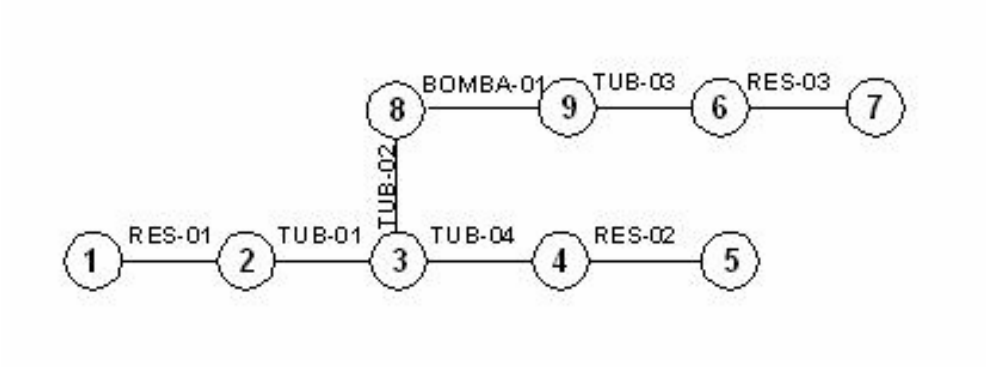

Figura 5.24 – Exemplo 5 – Modelagem e Codificação

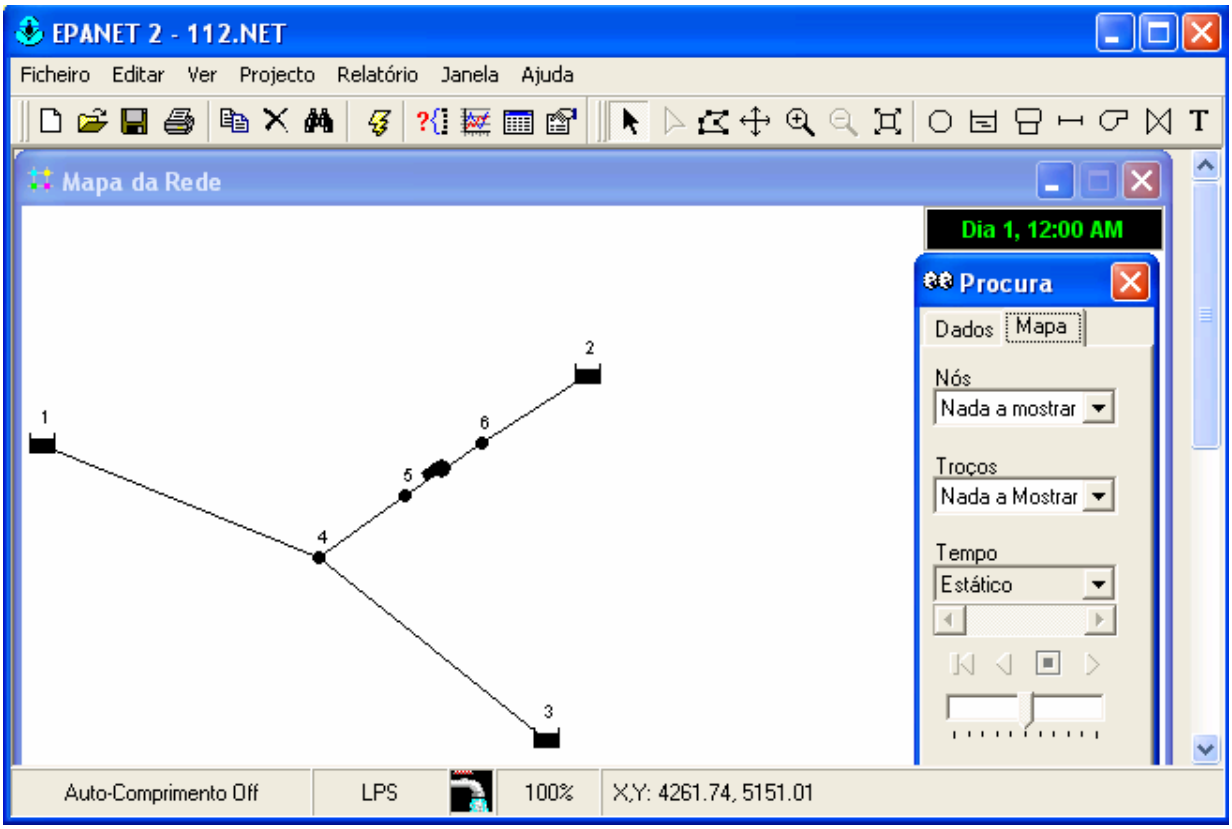

Figura 5.25 – Exemplo 5 – Representação da rede no EPANET

Tabela da Rede - Nós

ID do Nó	Consumo LPS	Carga Hidráulica m	Pressão m	Qualidade
Nó 4	30.00	100.03	100.03	0.00
Nó 5	0.00	101.95	101.95	0.00
Nó 6	0.00	83.08	83.08	0.00
RNF 1	9.82	100.00	0.00	0.00
RNF 2	-45.05	85.00	0.00	0.00
RNF 3	5.23	100.00	0.00	0.00

Figura 5.26 – Exemplo 5 – Resultados do EPANET nos NÓS da Rede

Observa-se na Figura 5.26 que as respostas obtidas utilizando o EPANET, para análise do regime permanente, mostram resultados que se aproximam bastante. No caso da Figura 5.25 é

analisado o comportamento do Nó 4, que foi considerado como Nó 3 na Figura 5.23, o mesmo que atinge o regime permanente até os 20 segundos iniciais.

A manobra simulada para a análise dos transientes é admitida supondo que há uma parada da bomba. Os resultados do transiente após ter atingido um estado permanente são mostrados nas Figuras a seguir:

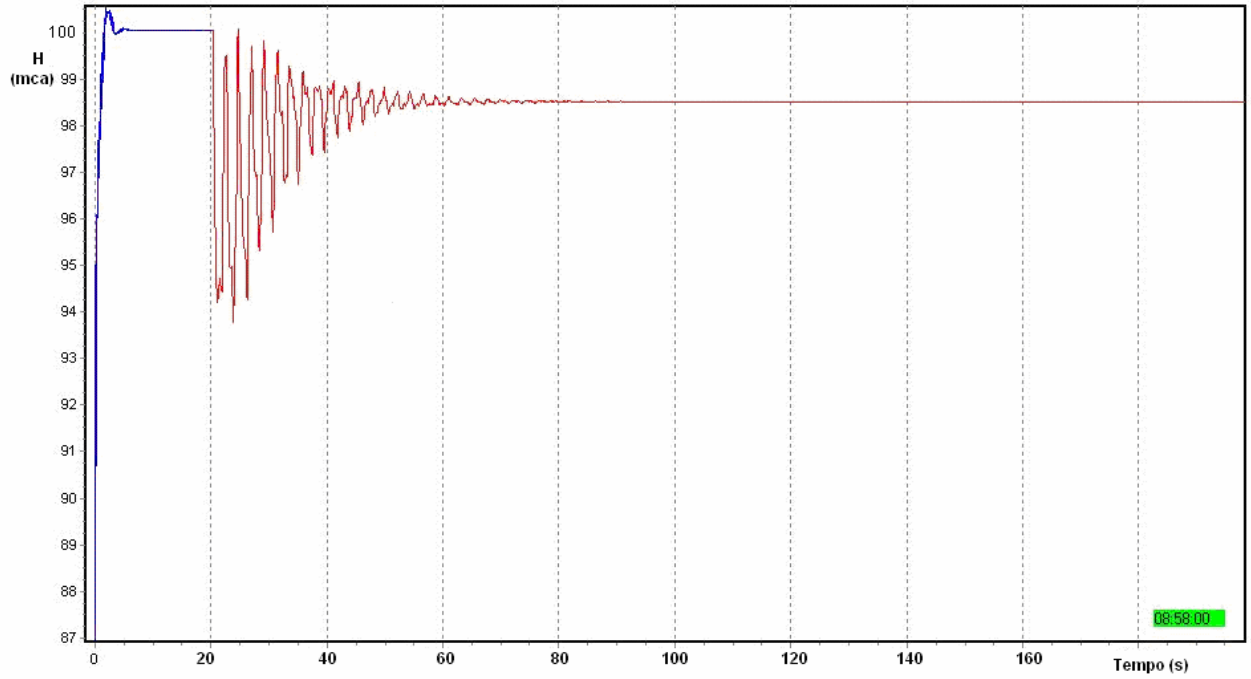

Figura 5.27 – Exemplo 5 – Transitório na Parada de Bomba

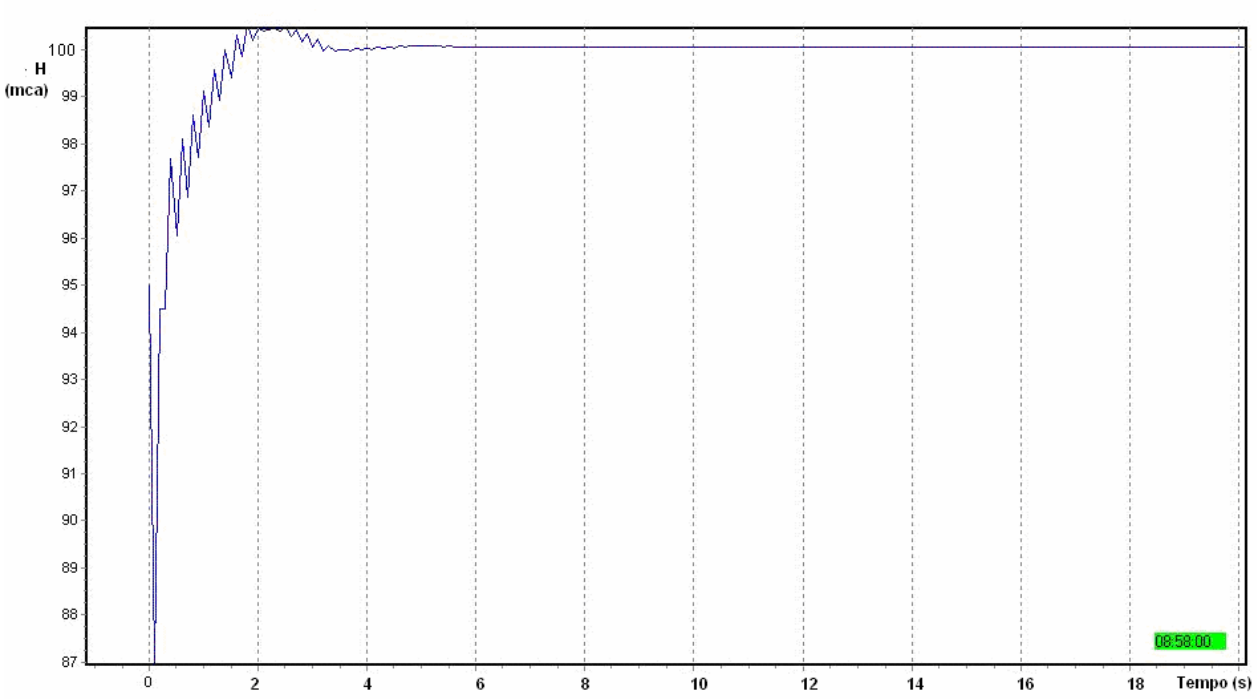

Figura 5.28 – Exemplo 5 – Análise do Permanente

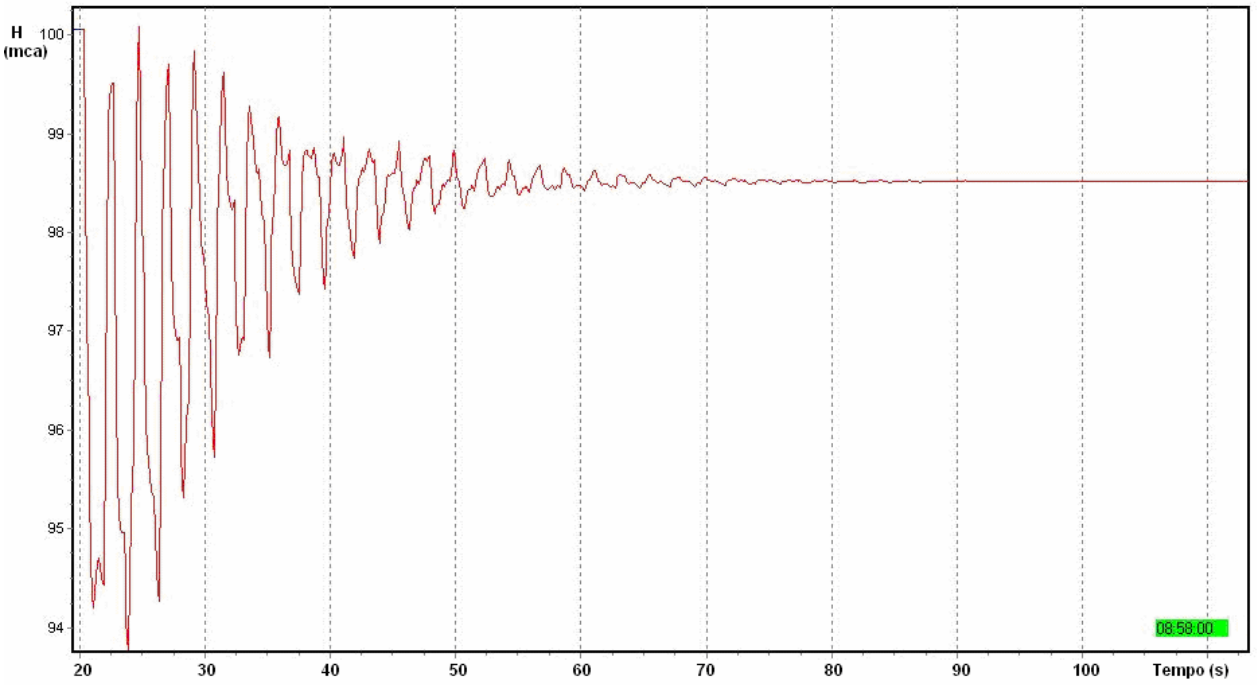

Figura 5.29 – Exemplo 5 – Parada de Bomba

Capítulo 6

Resultados e Discussões

Nesta dissertação foram lançadas as bases e foi desenvolvido um sistema computacional baseado na filosofia da linguagem orientada por objetos. Este sistema é composto por uma biblioteca de classes de objetos relacionados a escoamento em regime permanente e transitório com uma modelação baseada no Método Inercial Elástico e solução numérica através do Método das Características (MOC).

A pesquisa teve como objetivo o desenvolvimento de uma biblioteca de classes de objetos relacionados a escoamentos em condutos forçados. Essa biblioteca deverá constituir a base de um sistema que poderá crescer indefinidamente, permitindo que novas teorias e idéias envolvendo escoamentos em condutos forçados possam ser rapidamente implementadas com a utilização de objetos especializados, exaustivamente testados, levando a uma otimização de tempo e esforço do pesquisador que vier a introduzir novas potencialidades ao sistema. Essas potencialidades a serem introduzidas posteriormente poderão incluir análise dinâmica, otimização envolvendo diferentes materiais, combinação com diferentes tipos de elementos e eventualmente detalhamento do sistema de distribuição de água.

No exemplo 1 do estudo de caso foi analisado o comportamento de um sistema hidráulico apresentado na literatura, a fim de se verificar o comportamento do programa desenvolvido neste trabalho. Na Figura 5.1 é apresentado o resultado obtido por KARNEY e MCINNIS e na Figura 5.2 são apresentados os resultados obtidos com o programa descrito neste trabalho.

No exemplo 2 é analisado o comportamento da chaminé de equilíbrio. Durante o fenômeno transitório utilizou-se o sistema ilustrado na Figura 5.3 que está composta de 8 nós, 2 reservatórios, 3 tubos, 1 chaminé e 1 válvula. O regime permanente é calculado inicialmente considerando que a válvula encontra-se fechada até os 200 segundos (Figura 5.6). Após esse intervalo é simulada uma manobra de abertura instantânea da válvula para análise dos transientes.

A chaminé de equilíbrio provoca uma manobra lenta na instalação e o nível tende asintoticamente para um valor que corresponderá a um regime permanente final em 300 segundos (Figura 5.7), a seguir é simulada uma manobra de fechamento instantâneo da válvula mostrando se a atuação da chaminé de equilíbrio na atenuação das ondas de pressão até um regime permanente final nos 1600 segundos (Figura 5.8).

O exemplo 3 ilustra o comportamento do reservatório unidirecional durante o fenômeno transitório como ilustrado nas Figuras 5.9 e 5.10. O regime permanente é calculado inicialmente considerando que a válvula encontra-se fechada até os 100 segundos (Figura 5.12). Após esse intervalo é simulada uma manobra de abertura instantânea da válvula para análise dos transientes (Figura 5.13). O reservatório unidirecional em condições de regime permanente impede que haja fluxo da instalação para o reservatório. No transitório para uma onda de depressão processa-se a descarga do tanque, atenuando-se assim as depressões excessivas, seguidamente é simulada uma manobra de fechamento instantâneo da válvula (Figura 5.14) mostrando se a atenuação das ondas de pressão até um regime permanente final nos 1000 segundos.

No exemplo 4 é analisado o comportamento de vaso de pressão. Durante o fenômeno transitório utilizou-se o sistema ilustrado nas Figuras 5.15 e 5.16, análogo aos casos anteriores. O regime permanente foi calculado inicialmente considerando que a válvula encontra-se fechada até os 200 segundos (Figura 5.18). Após esse intervalo foi simulada uma manobra de abertura instantânea da válvula para análise dos transientes (Figura 5.19). O vaso de pressão apresenta no seu interior uma pressão tal que equilibra a pressão de serviço da instalação em regime permanente, no ponto em que este instalado, o vaso de pressão consegue absorver pressões elevadas através da compressão do ar no seu interior e as quedas de pressão através da descarga de volume de água no seu interior. Obtendo um regime permanente final nos 300 segundos (Figura 5.19), seguidamente foi simulada uma manobra de fechamento instantâneo da válvula,

mostrando se a atuação do vaso de pressão na atenuação das ondas de pressão até um regime permanente final nos 1000 segundos, mostrando inicialmente um período de excitação para em seguida entrar na fase de oscilação (Figura 5.20).

O exemplo 5 ilustra os resultados obtidos no NÓ 3 (Figura 5.23), que analisa o comportamento numa parada de bomba. Durante o fenômeno transitório utilizou-se a modelagem ilustrada na Figura 5.24 que está composta de 9 nós, 3 reservatórios, 4 tubos, 1 bomba. O regime permanente é calculado, inicialmente utilizando EPANET (Figura 5.25), considerando que a bomba encontra-se ativa até os 20 segundos (Figura 5.28). Após esse intervalo foi simulada uma manobra de parada de bomba para análise dos transientes. É mostrado o comportamento das pressões no sistema após a parada da bomba que corresponderá a um regime permanente final em 100 segundos (Figura 5.29).

Capítulo 7

Conclusões

Pelos resultados obtidos nesta dissertação, estudos de casos com o modelo desenvolvido, podemos concluir o seguinte:

a) O programa orientado por objetos apresentado neste trabalho para análise de escoamentos em condutos forçados em regime permanente e transitório pode ser empregado com sucesso na simulação de sistemas hidráulicos em substituição às metodologias de programação convencionais, como ficou demonstrado através dos diversos exemplos ilustrativos no Capítulo 5.

b) O presente trabalho mostrou que com o atual avanço nas técnicas de computação, a filosofia orientada a objetos pode ser usada como uma ferramenta de apoio para a simulação de sistemas hidráulicos.

c) Como experiência pessoal o programa desenvolvido foi receptivo às necessidades de manipular o código já existente, tornando o código mais claro e fácil de gerenciar. A codificação e a metodologia propostas para a representação hidráulica favoreceram a idealização gráfica dos principais modelos que foram gerados pela UML.

d) A modelagem apresentada neste trabalho deixou em evidência a facilidade de aplicação do paradigma orientado a objetos na resolução dos problemas hidráulicos através de uma construção que represente, da melhor forma possível, como as coisas acontecem no mundo real.

e) A filosofia orientada a objetos se mostrou superior à programação procedural tradicional sob os seguinte aspecto:

- Os códigos fonte podem ser reutilizados no desenvolvimento de aplicações e na ampliação de bibliotecas geradas por terceiros.

f) Como resultado da comparação do programa desenvolvido nesta dissertação com o programa EPANET para análise de regime permanente e o trabalho publicado por KARNEY e MCINNIS (1990) resolvendo o regime não permanente conclui-se que o presente trabalho pode ser aceito como a base de um sistema computacional baseado na filosofia da linguagem orientada por objetos com emprego do modelo dinâmico inercial elástico fazendo-se uso do Método das Características (MOC) para o tratamento de problemas de escoamento em condutos forçados, em regime permanente e transitório.

g) Sendo que as equações de continuidade e quantidade de movimento regem o modelo dinâmico inercial elástico são gerais e englobam os demais modelos como casos particulares, com base neste fato podemos usar este modelo para a simulação de diversas formas de escoamentos.

h) Finalmente como sugestão para pesquisas futuras propõe-se ampliar objetos e métodos com a finalidade de cobrir um maior número de dispositivos num sistema hidráulico.

Capítulo 8

Referências Bibliográficas

ANDRADE, J.G.P. (1994), Análise e otimização da operação de usinas hidroelétricas, Tese de livre docência, Universidade Estadual de Campinas.

ADELI, H. AND YU, G. An object-oriented data management model for numerical analysis in computer aided engineering. Microcomputer Civil Eng., 8(3): 199–209, 1993.

ANJO, L.F.R.S. (2003), Análise do comportamento teórico e experimental de um sistema piloto de abastecimento de água, Tese de Mestrado, Universidade Estadual de Campinas, 73f.

ARCHER, G.C., FENVES, G. AND THEWALT, C. A new object-oriented finite element analysis program architecture. Computers & Structures, 70(1): 63–75, 1999.

BOOCH, G. Object oriented analysis and design with applications, Benjamin/Cummings Publishing Company Inc., 2nd ed., Redwood city, California, USA, 1994, 589 pp.

BOOCH, G., RUMBAUGH, J. AND JACOBSON, I. The unified modeling language user guide, Reading M.A.: Addison-Wesley Publishing Company, Massachusetts, USA, 2003, 482 pp.

BROOKS, F.P. The mythical man-month, Essays on software engineering reading, Addison Wesley Publishing Company, Massachusetts, USA, 1975.

BUDD, T. Introducción a la programación orientada a objetos, Addison Wesley Iberoamericana, Oregon state university, USA, 1991, 409 p.

CESTA, A.A. "C++ como uma linguagem de programação orientada a objetos", Instituto de Computação, Unicamp, SP, Brasil, 1996.

.

DIORIO, F.A, (1996), Um ambiente computacional orientado por objetos para análise de estruturas aporticadas tridimensionais, Tese de Mestrado, Universidade Estadual de Campinas, 185 pp.

FOX, J.A. Hydraulic analysis of unsteady flow in pipe networks, The Mcmillam Press. Ltd., 1977.

GOLDBERG, Oriented language on the programming environment, in: D.R. BARSTOW, H.E. SHROBE AND E. SANDEWALL (editors), Interactive Programming Environments, McGraw Hill, New York, USA, 1984, 609 pp.

INGALLS, D.H. Design principles behind Smalltalk, Byte, 6(8), p. 286-298, Actor language manual, The Whitewater Group Inc., Evanston, Illinois, USA, 1987.

JACOBSON, I., CHRISTERSON, M., JONSSON, P., ÖVERGAARD, G. Object oriented software engineering. Addison Wesley Publishing Company, 1995, 528 pp.

KARNEY, B.W. AND MCINNIS, D. "Transient Analysis of Water Distribution Systems.", Journal AWWA, 82(7), 62-70.

KAY, A. Microelectronics and the personal computer, Scientific American, 237(3): 230-244, 1977.

KOELLE, E., ANDRADE, J.G.P., (1990), Análise da operação transitória da usina hidroelétrica, A representação das curvas características das máquinas hidráulicas, Conferência Internacional Small Medium, p 281-290, São Paulo.

KOELLE, E. (1982), Transientes hidráulicos em instalações e condutos forçados, Tese de Livre Docência, Escola Politécnica da Universidade de São Paulo.

KOELLE, E., (2001), Apostila de notas de aula do curso de transiente hidráulico, UNICAMP, Campinas, Brasil.

LESSA, R, C. (1984), Transientes hidráulicos em sistemas complexos de adução de água, Tese de Mestrado, Escola de Engenharia de São Carlos, Universidade de São Paulo, 117 pp.

LUVIZOTTO, JR. E. (1992), A representação analítica das curvas características das máquinas hidráulicas para uso em rotinas de simulações computacionais, Dissertação de Mestrado, Escola Politécnica da USP.

LUVIZOTTO, JR. E. (1995), Controle operacional de redes de abastecimento de água auxiliado por computador, Tese de Doutorado, Escola Politécnica da USP, 143 pp.

MACKERLE, J. Object oriented programming in FEM and BEM: a bibliography (1900-2003), Advances in Engineering Software, 35(6): 325–336, 2004.

MACKIE, R. I. Object oriented programming of the finite element method, International Journal for Numerical Methods in Engineering, 35: 425-436, 1992.

OLIVEIRA, A.C., COENTE F.M., GONÇALVES RISO B, Aspects on teaching learning with object oriented programming for entry level courses of engineering, Universidade Federal de Santa Catarina, Departamento de informática e estatística.

PAPPAS, C.H. AND MURRAY, W.H. Turbo C^{++} completo e total. Makron Books do Brasil Editora Ltda., São Paulo, Brasil, 1991, p. 771.

RIGHETTO, A.M. (1977), Desenvolvimento de modelos de simulação para o dimensionamento de redes de distribuição de água, Tese de Doutorado, Escola de Engenharia de São Carlos, Universidade de São Paulo.

RUMBAUGH, J., BOOCH, G. AND JACOBSON, I. Modelagem e projetos baseados em objetos, Editora Campus Ltda., 1994, 652 pp.

SCHOLZ, S.P. Elements of an object oriented FEM^{++} program in C^{++}. Computers & Structures, 43(3): 517-529, 1992.

SISTLA, R., DOVI, A.R., SU, P. A distributed, heterogeneous computing environment for multidisciplinary design and analysis of aerospace vehicles. Adv. Eng. Software, 31(8/9): 707–716, 2000.

STREETER, V.L. AND WYLIE, E.B. Fluid transients in systems, McGraw Hill International Book Co., New York, USA, 1993, 384 pp.

SWAN, T. Aprendendo C^{++}. Editora Campus Ltda., Rio de Janeiro, Brasil, 1993, p. 675.

TABATABAI, S.M.R. Object-oriented finite element-based design and progressive steel weight minimization, Finite Elements in Analysis and Design, 39(1): 55–76, 2002.

TAKAHASHI, T. Introdução a programação orientada a objetos, Edição EBAI, Curitiba, Brasil, 1988, 148 pp.

WATSON, A.S. AND CHAN, S.H. Aprolog-based object oriented engineering DBMS. Computers & Structures. 40(1): 11-21, 1991.

WEISKAMP, K., HENRY, L. AND FLAMING, B. Programação orientada para objeto com turbo C^{++}. Makron Books do Brasil Editora Ltda., São Paulo, Brasil, 1993, p. 474.

ZIMMERMANN, T., BOMME, P., EYHERAMENDY, D., VERNIER, L., COMMEND, S. Aspects of an object-oriented finite element environment. Computers & Structures, 68(1/3): 1–16, 1998.