UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

ESTUDO DA UTILIZAÇÃO DA LINGUAGEM JAVA NO DESENVOLVIMENTO DE APPLETS E APLICATIVOS PARA ENSINO E PESQUISA DE ENGENHARIA QUÍMICA

Dissertação apresentada ao Curso de Pós-Graduação em Engenharia Química do Centro Tecnológico da Universidade Federal de Santa Catarina, como requisito parcial à obtenção do título de Mestre em Engenharia Química

Orientador: Prof. Luismar Marques Porto, Ph. D.

Rafael Hernandes Ogeda

Florianópolis (SC), fevereiro de 1998

ESTUDO DA UTILIZAÇÃO DA LINGUAGEM JAVA NO DESENVOLVIMENTO DE APPLETS E APLICATIVOS PARA ENSINO E PESQUISA DE ENGENHARIA QUÍMICA

por

Rafael Hernandes Ogeda

Essa dissertação foi julgada para a obtenção do título de

Mestre em Engenharia

Especialidade **Engenharia Química**

Área de Concentração
Desenvolvimento de Processos Químicos e Biotecnológicos

e aprovada em sua forma final pelo curso de Pós-Graduação

Prof. Luismar Marques Porto, Ph. D.
Orientador

Prof. Leonel Teixeira Pinto, D. Sc.
Coordenador do curso de Pós-Graduação em Eng. Química

Banca Examinadora:

Prof. Luismar Marques Porto, Ph. D. (Presidente)

Prof. Álvaro G. Rojas Lezana, D. Eng.

Prof. Ariovaldo Bolzan, D. Sc.

Prof. Klaus Friedrich W. P. Hektor, Dr. rer. nat.

Florianópolis (SC), fevereiro de 1998

AGRADECIMENTOS

O autor deseja agradecer ao Departamento de Engenharia Química e à Universidade Federal de Santa Catarina, pela formação e pela oportunidade de realizar este trabalho.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq – , pelo apoio financeiro.

Ao professor José Carlos Petrus, pelo apoio e amizade.

Ao professor Luismar Marques Porto, pela orientação, amizade e ensinamentos.

Aos amigos Adauto Scalon, Adriano Cancelier, Carlos Claumann, Marcos Mazzuco, Ricardo Machado e tantos outros, pela ajuda, paciência e amizade.

Aos colegas do Curso de Mestrado em Engenharia Química, pelo acompanhamento nesta caminhada.

A meus pais (Luiz e Adelaide), minhas irmãs (Alessandra, Cláudia e Tânia), minha noiva (Cyntia) e família (João César, Marlise, Marlon e Clébio), pelo apoio essencial nas horas mais difíceis.

A todos que contribuíram direta ou indiretamente para a realização deste trabalho.

À Internet e seus colaboradores, pelo despertar dessa tecnologia maravilhosa e cativante.

E um agradecimento especial a Deus, pela iluminação e pelas oportunidades concedidas durante toda minha vida.

ÍNDICE ANALÍTICO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

a	Atividade catalítica, adimensional
A	Área de troca térmica, m^2
c	Concentração do reagente chave, mol/dm^3
C_A	Concentração do componente A, mol/dm^3
C_p	Capacidade calorífica, J/mol.K
Da	Número de Damköhler, adimensional
$D_{z,ef}$	Difusividade efetiva, m^2/s
F	Fluxo volumétrico, dm^3/s
H	Calor de reação, J/mol
k	Constante específica de velocidade de reação, s^{-1}
k_d	Constante específica de velocidade de desativação, $dm^3/mol.s$
K_p	Constante de equilíbrio de pressão parcial, adimensional
L	Comprimento do reator, m
m	Ordem de reação, adimensional
P	Pressão, Pa
Pe	Número de Peclet, adimensional
q	Ordem de decaimento catalítico em relação à concentração de reagente, adimensional
r'_A	Velocidade de reação do componente A, mol/g.s
t	Tempo de reação, s
T	Temperatura, K
U	Coeficiente global de transferência de calor, $J/m^2.min.K$
V	Volume do reator, dm^3
v_z	Velocidade axial do fluido, m/s
W	Massa de catalisador, g
X	Conversão do reagente chave, adimensional
x	Variável espacial, adimensional
y	Concentração, adimensional

Subscritos

0	Condição inicial ou de alimentação
c	Propriedade relativa à camisa de resfriamento

Símbolos gregos

ρ	Densidade, kg/dm^3
μ	Viscosidade, g/m.s

GLOSSÁRIO

API. Abreviatura para Application Programming Interface. Uma Interface de Programação é um conjunto de classes ou componentes que possibilita o desenvolvimento de programas em uma linguagem de programação específica.

Applet. Um programa Java que aparece embebido em um documento Web.

Biblioteca de Linkedição Dinâmica. Uma módulo de biblioteca executável, geralmente no formato binário como um arquivo externo, que será carregado durante o tempo de execução de um programa quando for necessário. Na plataforma Windows, é usualmente chamada de DLL.

Bytecode. Código Java compilado. Bytecodes são instruções portáveis que podem ser executadas por qualquer Máquina Virtual Java.

Cache. Espaço da memória ou do disco rígido local reservado para armazenamento de arquivos transferidos pelo navegador Web.

Class Loader. A parte da máquina virtual que extrai classes do sistema de arquivos cliente ou através da rede.

Cliente. Uma entidade que confia em outra entidade (servidor) para concluir algum objetivo. Clientes podem ser tanto simples classes chamando outras classes, quanto programas clientes chamando programas servidores através da rede.

Compilador. Uma utilitário que traduz um código fonte de linguagem de alto nível para código binário passível de ser interpretado diretamente pelo processador ou máquina virtual.

Conexão. Ligação através de um protocolo de comunicação entre dois computadores de uma rede.

Datagrama. Um tipo de pacote de dados que representa uma comunicação completa. Não são necessários estágios de conexão ou desconexão para a comunicação por datagramas.

Download. Ação de transferir arquivos de um computador da rede para um computador local.

Exceção. Uma condição anormal que altera o fluxo normal de um programa.

Garbage Collection. Uma característica de gerenciamento de memória automático que descarta blocos de memória não utilizados, liberando a memória para novo armazenamento.

Host. Um computador de uma rede.

HTML. Abreviatura para Hypertext Markup Language. A linguagem na qual os documentos Web são escritos. Os applets Java aparecem embebidos em documentos HTML.

HTTP. Abreviatura para Hypertext Transfer Protocol. O protocolo de aplicação utilizado pela World Wide Web para solicitações, transmissões e recebimento de documentos Web.

Interpretador. Um utilitário que lê os comandos em um arquivo e então interpreta e executa cada comando, um por vez.

IP. Abreviatura para Internet Protocol ou Internetwork Protocol. O protocolo central da Internet no qual todos os outros protocolos a nível de aplicação estão sustentados. Algumas das características do IP são a não garantia de entrega de dados e a possibilidade de transferência de pedaços de dados com no máximo 64Kbytes

JDK. Abreviatura para Java Development Kit. A série de ferramentas de desenvolvimento Java distribuídas gratuitamente pela Sun Microsystems. O JDK consiste principalmente das classes da API Java, um compilador Java e o interpretador da Máquina Virtual Java.

JIT. Abreviatura para Compilador Just-In-Time. Compilador que converte os bytecodes verificados para instruções de processador nativas antes da execução e pode aumentar significativamente a desempenho de aplicações Java.

Máquina Virtual Java. O sistema que carrega, verifica e executa bytecodes Java.

Modelo de Aplicação Cliente/Servidor. Um modelo de aplicação comumente empregado em ambientes de rede. O programa de aplicação monolítico é dividido em duas metades; uma metade executando em uma máquina servidora, e outra executando em máquinas clientes. O modelo cliente/servidor é utilizado como uma solução em sistemas multiusuário, onde recursos centrais necessitam ser compartilhados/alterados/consultados por muitos.

Multithread. O meio de realizar múltiplas tarefas independentes umas das outras.

Navegador Web. Um programa visualizador usado por computadores clientes para a exibição de documentos Web.

Pacote. Uma coleção de rotinas (classes) relacionadas.

Pacote de Dados. Uma unidade de comunicação.

Protocolo. A série de regras e as estruturas de pacotes de dados legais que são utilizadas para fornecer algum serviço de comunicação entre dois computadores. Exemplos comuns são TCP/IP, Z-Modem, Kermit, SLIP, PPP, X.25 e IBM SNA.

Servidor. Uma entidade cujo propósito é servir a clientes (seqüencialmente ou em paralelo) através do fornecimento de algum tipo de serviço.

Servidor Web. Um servidor de rede que, sob solicitação, transmite documentos através do protocolo HTTP.

Site Web. Um conjunto de documentos HTML pertencentes a uma determinada organização, assunto ou tema e armazenados em um servidor Web.

Solver. Programa destinado a fornecer a solução numérica de uma ou mais equações matemáticas.

String. Um seqüência de caracteres.

TCP. Abreviatura para Transmission Control Protocol. O protocolo orientado à conexão construído no topo do Protocolo Internet. O TCP garante a entrega de dados e pode manipular quantidades arbitrárias de dados.

Thread. Um simples fluxo de controle em um programa.

UDP. Abreviatura para User Datagram Protocol. Permite o envio de datagramas ao estilo do protocolo IP para uma porta em um computador.

Upload. Transferência de arquivos de um computador local para um computador da rede.

URL. Abreviatura para Uniform Resource Locator. Um string que identifica a localização de um recurso e o protocolo usado para acessá-lo.

Web. Veja *World Wide Web*.

World Wide Web ou **WWW**. Uma coleção de documentos HTML interconectados na Internet.

RESUMO

ESTUDO DA UTILIZAÇÃO DA LINGUAGEM JAVA NO DESENVOLVIMENTO DE APPLETS E APLICATIVOS PARA ENSINO E PESQUISA DE ENGENHARIA QUÍMICA

Palavras-chave: *Applets Java, ensino de engenharia química, monitoramento e controle à distância*

Applets e aplicações em linguagem Java foram desenvolvidos para dar suporte ao ensino e pesquisa em engenharia química. Foram escritos cinco applets simuladores descrevendo problemas típicos de engenharia de reações químicas, incluindo reatores do tipo batelada adiabático, CSTR fluidizado com desativação, CSTR encamisado, conversor industrial de SO_2 e leito fixo com dispersão axial. Um pacote matemático Java de propósito geral – labore.math, foi desenvolvido como uma biblioteca de classes padrão para a resolução de sistemas de equações algébricas e diferenciais ordinárias. Excelente desempenho foi obtido para soluções numéricas utilizando uma ampla faixa de parâmetros. O tempo de transferência dos applets foi considerado aceitável, abaixo de um minuto para um tempo de resposta de 185ms do utilitário ping. Foram também desenvolvidas duas aplicações cliente/servidor: um sistema de monitoramento de experimentos de microgravimetria que permite ao usuário modificar o intervalo médio de amostragem, e outra aplicação cliente/servidor para ajuste de set-points e monitoramento à distância da temperatura, pressão e fluxo permeado de uma unidade de ultrafiltração tangencial. Os resultados mostram a potencialidade dos sistemas para a implementação de estratégias de monitoramento e controle de processos pela rede. O atraso na resposta foi desprezível tanto para a comunicação cliente/servidor quanto para o tempo de amostragem de aquisição de dados para as condições testadas.

ABSTRACT

JAVA LANGUAGE UTILIZATION STUDY IN THE DEVELOPMENT OF EDUCATIONAL AND RESEARCH CHEMICAL ENGINEERING APPLETS AND APPLICATIONS

Keywords: *Java applets, chemical engineering education, remote monitoring and control*

Java Applets and Applications were developed to support chemical engineering education and research. Five simulation applets were written to describe typical chemical reaction engineering problems, including adiabatic batch, deactivating fluidized CSTR, jacked CSTR, SO_2 industrial converter and a fixed bed with axial dispersion reactor. A general purpose Java mathematical package – labore.math, was developed as a standard class library to solve systems of algebraic and ordinary differential equations. Excellent performances were found for the numerical solutions making use a wide range of the parameters. Applets downloading time was acceptable and below one minute for an Internet ping average response time of 185ms. Two client/server applications were developed: a monitoring system to follow microgravimetric experiments, that allows the user to modify the sampling average interval, and another client/server application to adjust set-points and remotely monitor temperature, pressure and permeate flow rate of a tangential ultrafiltration unit. Results showed the potentiality of the systems to implement network monitoring and process control strategies. Response delay was insignificant for both the client/server communication and for the data acquisition sampling time for the conditions tested.

Capítulo 1

INTRODUÇÃO

A evolução apresentada na Internet nos últimos anos tem fomentado um grande avanço na tecnologia da informação. Os efeitos na Engenharia Química se fazem sentir em várias áreas de aplicação, como uma nova e eficiente maneira de comunicação, na obtenção de informações importantes para o rápido progresso na ciência e da indústria, no projeto e controle de instalações químicas e na comercialização de produtos.

Uma das tecnologias mais promissoras surgidas com o advento da Internet é a linguagem Java, que possibilita o desenvolvimento de programas simples, eficientes, modulares, reutilizáveis e portáveis em várias plataformas de hardware e sistemas operacionais. A linguagem Java começou a ser desenvolvida pela Sun Microsystems Corporation em 1991, sendo que sua aplicação com maior intensidade decorre de poucos anos. Java pode ser utilizada tanto para o desenvolvimento de programas carregados a partir da Internet e executados localmente no navegador Web do usuário, quanto para o desenvolvimento de aplicativos mais gerais que não necessitam de um navegador para executar.

A linguagem Java possibilita a criação de aplicativos independentes de plataforma através da utilização de "bytecodes" – instruções independentes de plataforma geradas durante a compilação. A execução de programas Java envolve a execução desses bytecodes em um interpretador. No caso dos applets, o interpretador já é parte integrante de navegadores compatíveis com Java. Para os aplicativos é necessário ter-se um interpretador para a plataforma na qual o mesmo será executado, sendo que a maioria das plataformas importantes possuem um interpretador Java disponível.

O objetivo geral deste trabalho é o estudo e avaliação da utilização desta nova ferramenta, a linguagem Java, no desenvolvimento de applets e aplicativos que forneçam suporte para o ensino e a pesquisa em Engenharia Química. Como objetivos específicos pode-se destacar:

- o desenvolvimento de um pacote matemático de classes Java contendo métodos numéricos para a resolução de sistemas de equações algébricas e diferenciais,

- o desenvolvimento de applets simuladores Java que utilizem o pacote matemático desenvolvido para a resolução de problemas típicos de Engenharia de Reações Químicas,

- o projeto, implementação e avaliação de um sistema cliente/servidor que possibilite a aquisição e o controle de processos químicos à distância,

- a implementação desse sistema em uma unidade de microgravimetria, e

- a automatização e implementação desse sistema em uma unidade de ultrafiltração tangencial.

O estudo na área de ensino em Engenharia envolve o desenvolvimento de cinco applets simuladores para a resolução de problemas típicos de Engenharia de Reações Químicas:

- Simulador de um Reator CSTR Encamisado

- Simulador de um Reator Adiabático em Batelada

- Simulador de um Conversor de SO_2

- Simulador de um Reator CSTR Fluidizado sujeito a Decaimento Catalítico

- Simulador de um Reator de Leito Fixo com Dispersão Axial, utilizando um Modelo Estacionário Isotérmico

Os métodos matemáticos para resolução de sistemas de equações envolvidos nesses simuladores foram modelados como classes Java, e compõem o pacote matemático labore.math, projetado e desenvolvido neste trabalho. O pacote possui uma estrutura hierárquica bem definida e é composto por classes implementando os métodos:

- da decomposição LU, para resolução de sistemas de equações algébricas lineares,

- de Newton-Raphson, para a resolução de sistemas de equações algébricas não-lineares,

- de Runge-Kutta de 4ª ordem, com monitoramento de erro e passo de integração ajustável, para a resolução de sistemas de equações diferenciais ordinárias em problemas de valor final, e

- da Colocação Ortogonal, para a resolução de sistemas de equações diferenciais ordinárias em problemas de valor de contorno.

A implementação e incorporação do pacote labore.math a um programa Java em desenvolvimento é rápida e simples. Poucas etapas são necessárias para a definição da modelagem matemática, resolução do problema e obtenção dos resultados. A maior conclusão sobre esse pacote matemático pode ser verificada com os resultados obtidos pelos applets simuladores na resolução dos mais variados tipos de sistemas de equações. São necessários apenas alguns segundos para a resolução do problema, com quantidades de pontos de simulação e precisão (10^{-8}) considerados adequados.

Um dos fatores mais discutidos quanto à aplicação da linguagem Java, o tempo de transferência dos programas Java pela rede, apresentou resultados positivos. Os simuladores necessitam em média de apenas um minuto para serem transmitidos do host que os armazenam até o host do usuário, para uma boa conexão Internet utilizando um modem de 33,6 Kb/s. Além de rodarem pela rede, os applets simuladores podem ser instalados e executados localmente. Não há diferença quanto ao desempenho dos simuladores em se utilizar o simulador instalado no disco rígido local ou carregado pela rede. Após a transferência completa, os simuladores executam localmente, como se estivessem instalados no disco rígido local. Destaca-se ainda a interface gráfica amigável com o usuário e a possibilidade de acesso aos applets simuladores diuturnamente a partir de qualquer local do mundo.

A utilização da linguagem Java juntamente com a arquitetura cliente/servidor para o desenvolvimento de sistemas de aquisição e controle à distância é outro dos objetivos deste trabalho. O sistema é constituído basicamente de um applet cliente, acessado pela rede e que possibilita conexões remotas ao segundo componente do sistema: o servidor. O aplicativo servidor é responsável fundamentalmente pela aquisição e pelo controle da unidade. Além disso, possibilita a conexão de vários usuários simultaneamente e o envio e recepção de informações. Através da implementação de níveis de acesso, pode-se possibilitar o acompanhamento e controle de unidades de processo industriais ou laboratoriais. Mais do que isso, possibilita que unidades distantes ou localizadas em ambientes insalubres ou perigosos, possam ser acompanhados em tempo real sem a necessidade de presença humana no local.

O sistema foi implementado neste trabalho em duas unidades experimentais do Departamento de Engenharia Química da Universidade Federal de Santa Catarina: Unidade de Microgravimetria do LABORE/UFSC, que se destina principalmente a estudos de envenenamento de catalisadores; e Unidade de Ultrafiltração do LABSEM/UFSC, a qual se destina principalmente a estudos de clarificação de suco de maçã.

Os sistemas se mostraram eficazes tanto para a realização da tarefa de aquisição de dados, quanto para atuarem como servidores de experimentação à distância. O tempo de aquisição apresentou um desvio de apenas alguns centésimos de segundo quando comparado com uma aplicação de aquisição em tempo real ideal. Já o tempo de transferência dos applets foi similar ao obtido para os simuladores Java. O tempo de resposta do servidor, um dos fatores mais importantes na definição da viabilidade de implementação de sistemas de aquisição e controle à distância, apresentou resultados inferiores a um segundo, para uma conexão Internet boa. Caso esse tempo fosse muito alto, a autorização para que um usuário pudesse alterar algum parâmetro do processo poderia causar situações imprevisíveis, visto o atraso envolvido entre o envio do parâmetro e o recebimento integral pelo servidor.

Capítulo 2

ESTADO DA ARTE E REVISÃO BIBLIOGRÁFICA

2.1. A LINGUAGEM JAVA

A World Wide Web tem sido um meio de distribuir informação passiva a um número cada vez maior de pessoas. Com a adição de formulários e mapa de imagens, as páginas Web começaram a tornar-se interativas. Com o surgimento da linguagem Java, e a capacidade de páginas Web conterem "applets" Java, o potencial para divulgação de informações com interatividade concreta tornou-se uma realidade. Os applets são pequenos programas escritos em linguagem Java que rodam embebidos em páginas Web, e possibilitam que páginas Web praticamente estáticas se tornem verdadeiros programas rodando localmente no navegador ("browser") cliente (Lemay e Perkins, 1996). A discussão a seguir foi baseada em estudos realizados pelo autor e que se encontram parcialmente organizados no trabalho Linguagem Java (Ogeda e Porto, 1996). Sua grande fonte de consulta foram os livros de Lemay e Perkins, publicado pela Sams.net, e de Vanhelsuwé et al., publicado pela Sybex, bem como os manuais on-line sobre a linguagem. A descrição abaixo sobre as características, potencialidades e pontos fracos da linguagem Java tem o intuito de orientar os leitores menos envolvidos com as particularidades da programação em Java.

De fato, quase qualquer coisa que um pequeno programa possa fazer, applets Java também podem. Applets são carregados através da rede a partir de um servidor Web e executados dentro de uma página Web no "host" do usuário por um navegador compatível com a linguagem Java.

Outra grande vantagem que a linguagem Java oferece, e que outras linguagens não proporcionam, é a possibilidade de criar aplicativos que sejam totalmente independentes de plataforma. Pode-se desenvolver um aplicativo Java para resolver determinado problema em certa plataforma, e este aplicativo poderá ser executado tanto nesta como em qualquer outra. Sem necessidade de recodificação, nem de recompilação. Tudo o que se necessita é possuir o interpretador específico para a plataforma desejada. O que possibilita esta característica da linguagem Java são os "bytecodes" – instruções geradas durante a compilação, semelhantes aos códigos de máquina, mas que não são dependentes da plataforma e do processador onde o programa foi compilado.

Os applets, assim como os aplicativos Java, também são independentes de plataforma. Assim como arquivos HTML podem ser lidos em qualquer plataforma, os applets devem poder também ser executados em qualquer tipo de plataforma. No caso dos applets, o interpretador dos bytecodes é parte integrante do próprio navegador que suporta applets em Java.

Um problema com a linguagem Java é a velocidade de transmissão de dados na rede Internet, que ainda em geral é lenta. Dependendo das funcionalidades implementadas em um applet, o que está diretamente relacionado com a quantidade de dados que necessitam ser transferidos pela rede, o mesmo pode dispender desde alguns segundos até vários minutos para transmissão completa pela rede. A qualidade da conexão com a rede é um fator importante nesse tempo de espera.

Note-se que o desenvolvimento de programas com certo grau de complexidade (e consequentemente maior qualidade) em Java, exige familiarização não só com programação, mas também com orientação a objetos, com biblioteca de classes e com peculiaridades da linguagem.

2.1.1. Histórico

Java é uma linguagem de programação que está sendo desenvolvida pela Sun Microsystems Inc. (www.sun.com) desde 1991 como parte de um projeto de pesquisa para desenvolver software para equipamentos eletrônicos de consumo: televisores, videocassetes, torradeiras e outros tipos de máquinas que podem ser compradas em lojas de departamentos.

Modelada com base na linguagem C++, a linguagem Java foi projetada para ser pequena, simples, eficiente, modular, reutilizável, e portável em várias plataformas de hardware e sistemas operacionais, tanto a nível de código-fonte quanto a nível binário. Em Java, no entanto, não existem ponteiros e estruturas, e o gerenciamento de memória é realizado automaticamente. Todas as "strings" e vetores são sempre objetos reais, por exemplo.

A linguagem Java foi usada em vários projetos na Sun, mas não obteve grande atenção comercial até o desenvolvimento do navegador HotJava®. HotJava foi escrito em 1994, tanto para ser um veículo para carregar e executar applets, como um exemplo do tipo de aplicação complexa que pode ser escrita em Java. Atualmente, a Sun distribui a versão 1.1.5 de seu Kit de Desenvolvimento Java (JDK™), o qual pode ser obtido gratuitamente de seu site Web sobre Java (java.sun.com), e que inclui ferramentas para o desenvolvimento de applets e aplicativos Java.

2.1.2. Applets vs. Aplicativos

É importante que seja feita uma distinção entre as duas possibilidades de utilização da linguagem Java: applets e aplicativos. Um simples programa Java pode ser um applet ou um aplicativo, dependendo de como o programa é escrito, e das capacidades que o mesmo utiliza. A linguagem Java foi desenvolvida para ser uma poderosa linguagem de programação na qual pode-se realizar os mesmos tipos de tarefas e resolver os mesmos tipos de problemas que se poderia realizar em outras linguagens de programação, tal como C ou C++.

Os applets são pequenos programas carregados através da World Wide Web e executados por um navegador na máquina do usuário. Os applets dependem de navegadores capazes de interpretar os bytecodes Java gerados pelo compilador para executarem. Alternativamente, o Kit de Desenvolvimento Java oferece uma ferramenta que possibilita a execução de applets Java sem a necessidade de um navegador: o "appletviewer", um visualizador de applets. Aplicativos Java, por outro lado, são programas mais gerais que não necessitam de um navegador para rodar, mas sim de um interpretador específico para a sua plataforma.

O grande atrativo da linguagem Java é a possibilidade de se poder desenvolver applets que rodam em uma página Web no sistema do usuário que carrega o documento. Os applets aparecem em uma página Web da mesma forma que as figuras, mas ao contrário de figuras, applets são dinâmicos e interativos. Applets podem ser usados para criar animações, figuras ou áreas que respondem a eventos gerados pelo usuário, jogos e outros efeitos interativos, na mesma página onde são apresentados textos e figuras. O primeiro navegador capaz de suportar applets Java foi o HotJava, uma aplicação inteiramente desenvolvida pela Sun em linguagem Java. Atualmente este suporte está tornando-se disponível na maioria dos navegadores comerciais.

Para criar um applet, o programador escreve o código-fonte em linguagem Java, compila-o utilizando um compilador Java e o referencia em suas páginas Web escritas em HTML. Colocam-se os arquivos HTML e Java resultantes em um servidor Web da mesma forma que seria feito com arquivos HTML simples. Quando um usuário utilizando um navegador capaz solicitar a URL de uma página com um applet nela embutido, o navegador do usuário carregará o applet para o sistema local e o executará, possibilitando ao usuário visualizar e interagir com o applet.

2.1.3. Independência de Plataforma

A grande vantagem da linguagem Java em relação a outras linguagens é que os programas desenvolvidos em Java são independentes de plataforma, ou seja, os programas Java podem ser movidos facilmente de um computador para outro. Uma vez construído um programa Java com êxito, este programa poderá ser executado em qualquer tipo de plataforma sem a necessidade de reprogramação ou recompilação.

A nível de código, os tipos de dados primitivos em Java têm tamanhos consistentes através de todas as plataformas desenvolvidas. A nível binário, os arquivos Java também são independentes de plataforma e podem rodar em múltiplas plataformas sem a necessidade de recompilação do código. Isto ocorre porque os arquivos binários em Java estão em um formato chamado "bytecode", que são séries de instruções que se parecem bastante com alguns códigos de máquina, mas que não são específicas a nenhum processador.

Normalmente, quando um programa é compilado em C ou em outra linguagem de programação, o compilar traduz o programa para códigos de máquina ou instruções de processador. Estas instruções são específicas para o processador utilizado na compilação, e o programa compilado resultante somente rodará em um sistema similar. Se for necessário usar o mesmo programa em outro sistema, será preciso retomar ao código original, obter um compilador para o sistema desejado e recompilar o código. Desta forma, obtém-se vários programas executáveis para vários sistemas. Em Java é diferente: tem-se o compilador Java e o interpretador Java. O compilador atua sobre o código-fonte Java mas ao invés de gerar códigos de máquina a partir de seus arquivos, ele o traduz em código para uma Máquina Virtual especificamente projetada para suportar as características da linguagem.

Esse código traduzido é chamado bytecode. Para a execução de um programa Java, portanto, é necessário a execução de um programa interpretador, o qual executa o código traduzido. Não há necessidade de "linkagem", sendo que o interpretador conecta dinamicamente classes adicionais conforme a demanda do programa. Apesar de não ter de recompilar o programa Java para cada plataforma, há a necessidade de um interpretador específico para cada plataforma na qual deseja-se utilizar o programa. No caso de applets Java, o próprio navegador que suporta Java possui um interpretador de bytecodes inserido no mesmo. A Figura 2.1 mostra o processo de desenvolvimento de programas em linguagem Java.

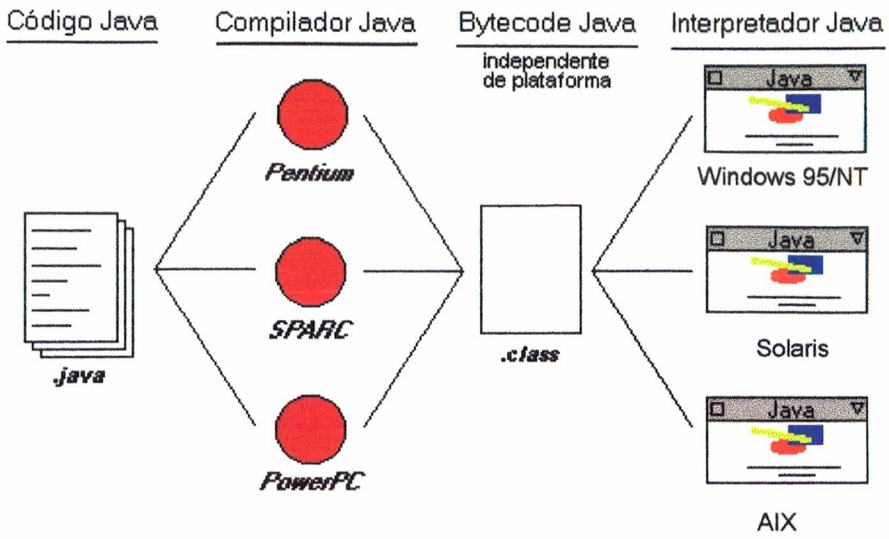


Figura 2.1 – Processo de desenvolvimento de programas em linguagem Java

Uma desvantagem de se usar bytecodes está na velocidade de execução. Programas específicos de um sistema rodam diretamente no hardware para o qual foram compilados, enquanto que bytecodes Java necessitam ser interpretados pelo interpretador. Para a maioria dos programas Java, a velocidade de execução pode não ser problema. Se for, tem-se algumas opções, que infelizmente podem custar a perda de portabilidade, como conectar códigos nativos em um programa Java ou usar ferramentas para converter os bytecodes Java em código nativo.

2.1.4. Orientação a Objetos

Outra vantagem é que a linguagem Java é uma linguagem totalmente orientada a objetos. A filosofia da orientação a objetos é que cada componente em um sistema se constitui em um objeto real, diferenciado dos demais. Cada texto, figura, botão, janela, etc., se constitui em um objeto distinto. E cada objeto é responsável por si próprio, por alterar seu estado e executar ações específicas relacionadas a ele. Na verdade cada objeto é geralmente composto por variáveis, as quais definem seu estado e sua diferenciação em relação a outros objetos semelhantes, e por métodos, os quais definem as ações realizadas pelo objeto.

A estrutura dos componentes com relação às variáveis e métodos é declarada em um representação abstrata do componente, denominada "classe". O objeto em si, também chamado "instância de classe", é a representação real de um determinado componente no sistema, através do qual pode-se manipular suas variáveis e seus métodos.

A orientação a objetos possibilita características vantajosas em um programa, como ser flexível, modular e reutilizável. Muitos dos conceitos de orientação a objetos em Java foram herdados da linguagem C++, mas também foram mesclados muitos conceitos de outras linguagens orientadas a objetos.

Como muitas linguagens orientadas a objetos, Java inclui uma série de bibliotecas de classe que oferecem tipos de dados básicos, capacidades de entrada e saída do sistema, e outras funções de utilização geral. Essas classes básicas são parte do Kit de Desenvolvimento Java (JDK), que também inclui classes para suporte de rede, protocolos Internet comuns, e funções para interface gráfica com o usuário. Assim como todos os programas Java, essas bibliotecas escritas em Java também são portáveis entre plataformas.

Maiores detalhes sobre a Programação Orientada a Objetos pode ser obtida no Apêndice A – Programação Orientada a Objetos em Java.

2.1.5. Kit de Desenvolvimento Java (JDK)

Atualmente, o JDK™ distribuído pela Sun se encontra na versão 1.1.5 e está disponível para plataformas Sun Solaris 2.4, 2.5 SPARC, Sun Solaris 2.5 x86 e Microsoft Windows 95, NT 4.0. O Kit de Desenvolvimento Java contém várias ferramentas e arquivos necessários para o desenvolvimento de programas em Java, como:

javac – compilador Java

java – interpretador Java

appletviewer – visualizador de applets, para testar e executar applets

jdb – protótipo de depurador em linha de comando que usa a API

javap – ferramenta para construção de um índice com declarações de classes

javadoc – gerador de documentos HTML com os comentários dos arquivos de código

classes.zip – arquivo com os arquivos compilados que compõem a API

src.zip – arquivo que contém o código-fonte das classes que compõem a API

A interface de programação Java (API) consiste de vários pacotes, específicos para cada tipo de funcionalidade que se deseje implementar. Os principais pacotes disponíveis na API Java são:

java.lang – define o núcleo da linguagem Java

java.util – oferece várias utilidades gerais, como datas e números randômicos

java.io – suporte para fluxo de dados (entrada/saída)

java.net – suporte para operações de rede

java.awt – compõem a interface gráfica do usuário

java.applet - procedimentos específicos para applets Java

2.1.6. Java vs. C++

Em Java tudo que diz respeito a uma classe fica contido em um único arquivo. A declaração de uma classe ou método é feita uma única vez no código e sua implementação deve ser feita simultaneamente à sua declaração. Não existe, como em C++, um arquivo exclusivo para a declaração da classe ("header") e outro arquivo exclusivo para sua implementação.

Um problema em grandes projetos em C++ é a "poluição" de nomes de classes. Java contorna este problema usando o conceito de "pacotes" ("packages"). Os pacotes são utilizados tanto para categorizar quanto para agrupar classes relacionadas. Os pacotes Java são arranjados em uma hierarquia, que além de melhor organizar classes relacionadas, permite que programadores Java possam criar e distribuir seus projetos através da rede.

Java não possui ponteiros. Entender o conceito de ponteiros é um dos aspectos mais difíceis da programação em C/C++, sendo que os ponteiros são também uma das maiores fontes de confusão e erro na programação. Em Java, objetos são passados diretamente como argumentos.

Os vetores em Java também são objetos reais, e deve-se manipulá-los utilizando-se referência explícita através de índices. C++ acessa vetores através de aritmética de ponteiros e suporta vetores multidimensionais, enquanto Java não os suporta. Em Java, existem vetores dentro de vetores.

Em C++ pode-se fazer chamadas a procedimentos existentes em C, incluindo muitas chamadas de sistema, simplesmente declarando o procedimento C como externo, através da sintaxe `extern "C"`. Em Java, há um método de importação similar, mas não tão simples de implementar. Deve-se definir um "método nativo", cujo propósito é realizar a interface com uma função C e então providenciar sua conexão. A interface com classes em C++ é ainda mais complexa, envolvendo a interface a classes C e os problemas normais de envolver funções C++ e funções C.

Em C++ exceções e tratamento de exceções são quase desconhecidas. Exceções são condições de erro que não são esperadas em processamento normal. Consequentemente, elas não são retornadas por um método como argumentos ou valores de retorno. No entanto, não podem ser ignoradas. Normalmente, a ocorrência de uma exceção não esperada causa o término anormal do programa. Em Java, exceções são parte essencial da linguagem. A declaração das funções-membro inclui informações sobre exceções, e o processador da linguagem obriga um estilo de programação na qual, ao se chamar um método que pode retornar uma exceção, deve-se certificar-se de que qualquer uma das exceções previsíveis ocorreu, e tratá-la. Quase todo programador Java se deparará com exceções. Trabalhar com exceções não é uma tarefa difícil, mas é uma coisa sobre a qual deve-se estar bem informado.

Java inclui um objeto String, que é constante. Um String em Java não é o mesmo que um vetor de caracteres, embora seja fácil de construir um objeto String a partir de um vetor de caracteres. Deve-se usar objetos String sempre que possível, uma vez que eles não podem ser sobrescritos com valores diferentes de forma não intencional.

Java gerencia automaticamente a alocação e liberação de memória, através do chamado "garbage collector". Em uma linguagem deste tipo, o dispositivo em tempo de execução memoriza quais espaços da memória estão em uso e quais não estão. Quando um espaço de memória não é necessitado por longo tempo, o sistema automaticamente requisita a memória. O sistema conhece quais objetos estão sendo utilizados, e quais possivelmente não serão utilizados novamente porque não há referências que apontem para ele. Como resultado, não há necessidade de preocupação sobre a destruição de objetos ou a liberação de memória. Muitos erros em C++ são causados pela deleção de objetos em uso ou por esvaziamento de memória causado pela não deleção de objetos alocados. O fato da linguagem Java ser garbage collected significa que não há necessidade de se escrever destrutores para todas as classes, não significando, no entanto, que nunca será necessário utilizá-los. Por exemplo, um objeto que abre uma conexão de rede necessita fechar a conexão quando ele é destruído. Em Java, um destrutor é conhecido como "método de finalização".

Java não implementa herança múltipla. Em qualquer sistema complexo orientado a objetos, implementar uma classe que necessita herdar a funcionalidade de mais de uma classe é um problema freqüente. A herança múltipla usada em C++ permite a uma classe ser derivada de mais de uma classe base. Ao invés disto, você pode declarar "interfaces", que descrevem a interface de programação usada para obter determinada funcionalidade. Uma classe pode então implementar uma ou mais interfaces, como também implementar sua funcionalidade própria. Classes diferentes, não relacionadas, podem implementar a mesma interface. Parâmetros formais para métodos podem ser declarados como classes ou interfaces. Se são interfaces, objetos de diferentes classes que implementam a interface podem ser passados como argumento para o método.

Em C++ utiliza-se "templates" para implementar tipos parametrizados, usados para descrever uma implementação para vários tipos similares de argumentos. Como exemplo, pode-se ter uma função que aceite parâmetros do tipo inteiro ou de ponto flutuante. Java não tem templates equivalentes aos de C++.

2.1.7. Multitarefa (*Multithread*)

Java foi projetada para suportar aplicações multitarefa desde a sua criação. A utilização de multitarefa permite escrever um programa que possa realizar duas ou mais operações ao mesmo tempo. Um programa multitarefa é dividido em diferentes linhas de execução ou tarefas ("threads"), para as quais são alocados recursos próprios. No entanto, as tarefas compartilham uma série de outros recursos do sistema, como o espaço na memória e arquivos abertos. O uso de multitarefa possibilita uma melhor utilização dos recursos computacionais (incluindo a CPU) e uma abordagem mais eficiente para a resolução de uma série de problemas. Para funcionar corretamente, um programa precisa ter controle sobre como as diferentes tarefas manipulam dados compartilhados ou fazem decisões baseadas em dados comuns a mais de uma tarefa. Alguns conceitos novos advêm da utilização de tarefas, sendo essenciais para uma programação multitarefa eficiente (Vanhelsuwé et al., 1996):

– A sincronização de tarefas é o método utilizado para evitar danos que podem ser causados pelo acesso concomitante aos mesmos. Como todas as tarefas em um programa compartilham o mesmo espaço da memória, é possível que duas tarefas tentem acessar a mesma variável ou executar o mesmo método de um mesmo objeto ao mesmo tempo. Através da sincronização é possível bloquear o acesso a determinado dado, caso o mesmo esteja sendo utilizado por uma tarefa, aumentando a estabilidade e robustez do programa.

– A comunicação intertarefa possibilita que as tarefas possam se comunicar ou aguardar umas pelas outras, através da utilização de métodos apropriados.

– Para cada tarefa criada em Java é atribuída uma prioridade, através da qual pode-se garantir que tarefas importantes ou que tenham forte dependência temporal sejam executas freqüentemente ou imediatamente. Quando mais de uma tarefa estiver competindo pelo tempo de CPU, a tarefa com o maior valor de prioridade terá a preferência. Os valores de prioridades que podem ser atribuídos às tarefas variam de um valor mínimo "um" até um valor máximo "dez". As aplicações desenvolvidas por usuários são normalmente executadas com o valor de prioridade 5.

– A programação ("scheduling") de tarefas é a atividade utilizada para determinar a ordem de execução de múltiplas tarefas.

– Tarefas "deamon" são tarefas normalmente executando em laços infinitos com a finalidade de fornecer serviços a outras tarefas.

2.1.8. Métodos Nativos

Os métodos nativos são utilizados em programas Java quando há a necessidade de utilização de uma capacidade especial de determinado computador ou sistema operacional. Tais capacidades incluem a interface com um novo equipamento periférico ou cartões "plug-in", o acesso a um tipo diferente de rede (Java suporta apenas TCP/IP), ou a utilização de uma característica única e valiosa de um sistema operacional em particular. A aplicação de métodos nativos em Java envolve o desenvolvimento de módulos otimizados específicos de plataforma – compiladores "Just-In-Time" (JIT), gerenciadores de segurança especializados e "class loaders" mais rápidos – para o suporte da linguagem Java em novos equipamentos tais como telefones celulares inteligentes e dispositivos de rede e para o controle de hardware especializado.

Para implementar um método nativo deve-se declarar um método Java e implementar sua funcionalidade utilizando a linguagem C. Utiliza-se o código em C e um compilador/"linker" para produzir uma biblioteca de linkedição dinâmica (DLL), a qual é carregada pelo sistema quando a classe contento o método nativo é instanciada. Quando uma chamada é realizada ao método nativo, o controle do programa é passado para a função correspondente na DLL. Após a função completar sua execução, o controle é retornado ao programa Java. Vale a pena lembrar que o uso de métodos nativos torna o programa Java não portável e dependente de plataforma.

2.1.9. Comunicação em Rede

A programação em rede em Java é realizada através de uma série de classes oferecendo suporte para o conjunto de protocolos de comunicação de dados TCP/IP. O protocolo TCP/IP (Transmission Control Protocol/Internet Protocol) implementa os dois principais protocolos de comunicação de dados de sustentação da Internet atual. A linguagem Java fornece suporte para dois protocolos de transferência de dados: TCP e UDP. O protocolo TCP apresenta um serviço de fluxo de dados orientado à conexão. Para que um computador possa enviar dados via TCP para outro computador, deve inicialmente ser estabelecida uma conexão entre os dois; somente então os dados podem ser transferidos. O protocolo TCP possibilita transmissões de quantidades ilimitadas de dados, com garantia de entrega ao destinatário e livres de erro. O protocolo UDP (User Datagram Protocol) é um protocolo TCP/IP de mais baixo nível orientado a datagramas, e que utiliza o protocolo IP para alcançar sua funcionalidade. Em termos de complexidade e facilidade de utilização, o protocolo UDP se situa entre o protocolo TCP e o protocolo IP. O protocolo UDP, cujos pacotes de dados são transmitidos individualmente com tamanho máximo de 64Kbytes, é útil principalmente quando informações de pouco valor necessitam ser difundidas ou transmitidas com freqüência, situações nas quais a perda de alguma parte da informação não afetaria o serviço. Para a maioria das necessidades de comunicação, todavia, existe a necessidade de garantir a entrega dos dados. Essa é a função do protocolo TCP.

2.1.10. Segurança

A linguagem Java oferece uma proteção concreta contra códigos nocivos através de uma série de defesas interligadas, que formam uma verdadeira barreira contra ataques à segurança do sistema. Na realidade, Java não introduz qualquer ameaça em relação a muitas atividades comuns na rede. Pode-se obter resultados desastrosos, por exemplo, carregando-se arquivos binários e executando-os, carregando macros auto-executáveis ou lendo-se mensagens de "e-mail" com programas executáveis embutidos. Além disso, a tendência é que a linguagem Java se torne ainda mais segura com o tempo, na medida em que se aprimora o seu desenvolvimento (Lemay e Perkins, 1996).

Os mecanismos de segurança implementados em Java agem em quatro níveis de arquitetura do sistema:

– A linguagem e o compilador são a primeira linha de defesa. A não utilização de ponteiros em Java, elimina duas falhas de segurança fatais observadas em linguagens como C++: com ponteiros, objetos

não podem ser protegidos de modificações externas ou duplicação e o uso de ponteiros é mais susceptível a erros graves de execução que comprometem a segurança do sistema, modificando algum outro objeto na memória.

– Antes da execução dos bytecodes, o interpretador sujeita-os a uma rigorosa série de testes de segurança, o que exige que eles estejam de acordo com as regras de segurança. Esses testes garantem que os bytecodes não forjem ponteiros, não violem restrições de acesso, não acessem objetos diferentes deles, não chamem métodos com algum argumento inapropriado, nem sobrecarreguem a memória. Na verdade, os bytecodes carregam consigo mais informações de tipo do que é estritamente necessário ao interpretador, permitindo a uma parte do sistema de interpretação, chamada "verificador", saber a qualquer momento todos os tipos existentes nos elementos da memória e em variáveis locais. Desta forma, o sistema pode saber de antemão esse conjunto completo de informações de tipo, verificar que todos os tipos de argumentos, parâmetros e resultados são corretos, e garantir que objetos Java e dados não são manipulados ilegalmente. Essa é parte crucial da segurança Java, e depende somente do seu sistema interpretador estar corretamente implementado, sem erros.

– Quando uma classe é carregada no sistema, entra em ação o "class loader", que é o dispositivo de segurança por excelência. O class loader essencialmente particiona o conjunto das classes Java em pequenos grupos protegidos, sobre os quais se pode seguramente fazer suposições que serão sempre verdadeiras. Este tipo de predição é a chave para programas bem-comportados e seguros. O class loader trata diferentemente os vários níveis de segurança ("campos") de onde pode ser carregada uma nova classe no sistema: seu computador local, a rede local na qual seu computador está conectado – protegida por um "firewall" –, e a Internet (rede global). O class loader nunca permite que uma classe de um campo menos protegido substitua uma classe de um campo mais protegido. Os tipos primitivos de entrada e saída do sistema de arquivos, sobre os quais poderia, e deveria, haver grande preocupação, são todos definidos em uma classe Java local, o que significa que estão todos localizados no campo do computador local. Assim, nenhuma classe de fora do seu computador pode obter a localização dessas classes e enganar o código Java usando versões destrutivas desses tipos primitivos. Além disso, classes em um campo não podem chamar métodos de classes em outros campos, a menos que essas classes tenham explicitamente declarado aqueles métodos públicos. Isto implica que classes de outro computador não podem nunca "ver" os métodos de entrada e saída do sistema de arquivos de seu computador local, muito menos chamá-los, a menos que o usuário ou o sistema queira fazê-lo. Como programador, pode-se criar um class loader personalizado e tê-lo sob controle, podendo-se

implementar quantos níveis de segurança forem necessários. No navegador, o usuário tem a possibilidade de selecionar um dos níveis de segurança padrão oferecidos.

– O último nível de segurança é a própria biblioteca de classes Java, que possui várias classes cuidadosamente planejadas e interfaces de programação que adicionam os toques finais à segurança do sistema. Uma classe abstrata, `SecurityManager`, foi adicionada ao sistema Java para colecionar, em um local, todas as decisões de estratégia de segurança que o sistema realiza quando os bytecodes executam. Como dito anteriormente, pode-se criar um class loader próprio, mas, é muito mais fácil derivar uma classe de `SecurityManager` para realizar muito do mesmo comportamento. Quando os bytecodes executam, o sistema utiliza o "gerenciador de segurança", que é sempre uma instância de alguma subclasse de `SecurityManager`, para controlar uma série de métodos pré-definidos, que têm permissão para serem chamados por determinada classe. Ele leva os campos mencionados anteriormente em consideração, a origem da classe e o tipo da classe (aplicativo ou applet). Cada um desses pode ser configurado separadamente para produzir o efeito desejado. Pode-se, por exemplo, especificar diferentes grupos de usuários, permitindo-se adicionar privilégios quando applets daquele grupo forem carregados. A série de métodos protegidos é composta por entrada e saída em arquivos, onde os applets podem abrir, ler ou escrever arquivos somente com autorização expressa do usuário, somente em diretórios restritos, criação e uso de conexões de rede para entrada e saída de dados, e permissão a tarefas para acessar, controlar e manipular outras tarefas.

2.2. A INTERNET E A ENGENHARIA QUÍMICA

Ao longo da última década, o computador tem se tornado uma ferramenta essencial para os profissionais das indústrias de processos químicos. Publicações impressas e contatos pessoais são ainda importantes fontes de informação, mas fontes de dados "online" estão se tornando mais importantes a cada ano. Os engenheiros atualmente utilizam cada vez mais a Internet para suas necessidades de informação (Shank e Kim, 1996).

Nos últimos anos, a Internet tem invadido cada vez mais as indústrias de processos químicos, alterando definitivamente a maneira de comunicação, busca de informações, projeto e controle de instalações químicas e comercialização de produtos (Fouhy et al., 1997).

2.2.1. Principais Serviços da Rede Internet

Entre os principais serviços disponibilizados pela Internet pode-se destacar:

- O correio eletrônico (e-mail), um sistema que possibilita a usuários enviar mensagens a qualquer outro usuário com um endereço de e-mail. Apesar do correio eletrônico possuir um formato de texto informal, ele permite ao remetente a inclusão de arquivos de qualquer tipo, sendo possível a inclusão de arquivos de várias páginas, documentos complexos, figuras CAD ou instruções de controle numérico de máquinas.

– Os "Usenet newsgroups", fonte de onde pode-se ler mensagens que outras pessoas tenham escrito sobre um determinado assunto, e escrever suas próprias mensagens. Por exemplo, o newsgroup sci.engr.chem pode conter muitas postagens sobre vários aspectos da engenharia química, sendo possível responder ou realizar uma pergunta a alguém participando do grupo.

– Um site "ftp" (file transfer protocol) funciona como um repositório de arquivos. Além de poder-se acessar o diretório principal e transferir seus arquivos, os quais podem ser longos documentos de texto, programas de computador ou arquivos gráficos, este serviço é utilizado como um caminho efetivo para cientistas, engenheiros, etc. trocarem dados.

– A "World Wide Web" é um sistema de computadores interconectados executando programas compatíveis que permitem a "navegação" hipertexto de um computador para outro. Com a utilização de programas chamados "navegadores" pode-se "saltar" de uma página localizada em determinado computador para outra selecionando conexões, ou "links", com o mouse, ou digitando um endereço URL. O acesso à Web é fornecido através de um servidor Web, que é usualmente instalado em um computador servidor localizado dentro de grandes organizações ou em provedores prestadores de serviços.

Durante os primeiros momentos da Internet atual, muitas companhias de produtos químicos utilizaram habitualmente apenas o correio eletrônico para se comunicar a baixos custos com escritórios distantes. Entre os pioneiros das indústrias de produtos químicos a lançarem seus produtos e serviços na rede pode-se destacar a National Intruments, American Institute of Chemical Engineering, GE Corp., Monsanto Co., DuPont Co. (www.dupont.com) e a Eli Lilly & Co. (Chowdhury, 1995).

2.2.2. Desenvolvimentos Internet Recentes

Após uma aceleração gradual a partir de março de 1995, os desenvolvimentos relacionados com a Internet têm experimentado um grande crescimento e, segundo observadores, muito mais mudanças podem ser esperadas nos próximos anos (Basta e Kim, 1996).

Os desenvolvimentos Internet recentes estão seguindo três caminhos:

– A Internet "pública", na qual corporações, vendedores e organizações constróem páginas Web para a "World Wide Web" que são acessíveis a qualquer usuário ao redor do mundo;

– "Intranets" – redes Internet internas – que fazem uso da tecnologia de comunicação Internet, mas que são tipicamente disponíveis somente a funcionários ou membros de organizações. Este desenvolvimento está afetando diretamente uma família já avançada de programas e práticas de comércio chamadas "groupware" ou "gerenciamento de fluxo de trabalho";

– Tecnologia "Internet-compatível". Um dos impactos da tecnologia Internet mais pesquisados atualmente é a possibilidade de servir como uma interface universal, permitindo que qualquer programa em qualquer tipo de plataforma seja interoperável. Uma grande quantidade de fabricantes de programas para as indústrias de processos químicos, em diversas áreas como projeto auxiliado por computador (CAD), controle de processos, gerenciamento de documentos ou conformidade ambientais, estão lançando versões atualizadas de seus programas com interfaces estabelecendo comunicações de dados via protocolos Internet.

A Internet pública é bem-estabelecida, tendo-se tornado um objetivo competitivo para muitas companhias que querem mostrar que podem empregar a última tecnologia rapidamente. Para muitas empresas de processos químicos, muito do valor da Internet está na obtenção de informações para segmentos alvos do público, incluindo clientes, fornecedores, estocadores e analistas industriais. A Internet ajuda a tornar mais eficiente a interface com sua audiência oferecendo um canal de comunicações diuturno, relativamente barato, através do qual vendedores possuem a oportunidade de comercializar seus produtos para uma base de consumidores em crescimento. Com os iminentes avanços na tecnologia, os fabricantes de produtos químicos estarão aptos a fornecer não somente informações de rotina sobre produtos ou capacidades de produção, mas soluções específicas para as necessidades de seus clientes através de programas desenvolvidos em seu próprio site Web. Através da revisão dos registros de transações e visitas ao site Web, pode-se determinar quem acessou o sistema, o resultado daquele acesso e planejar as atividades necessárias de acompanhamento do cliente.

A Web pode também ser efetiva como uma ferramenta de recrutamento, sendo que atualmente muitas companhias de processos químicos oferecem informações de emprego em suas páginas. Mas a principal função da Web no presente parece ser auxiliando a carga de comunicações externas de companhias de processos químicos. O site Web da Air Products and Chemicals (www.airproducts.com), por exemplo, armazena 400 arquivos sobre a empresa e seus produtos. O site Web da DuPont caracteriza os produtos químicos intermediários da companhia, incluindo diagramas. O site Web da Aspen Technology Inc. (www.aspentech.com) possui volumes de literatura sobre os produtos de software e hardware da empresa. A Rohm & Haas apresenta em seu site Web um repositório de planilhas de dados e segurança de materiais (MSDSs) para seus produtos, a partir de um sistema de banco de dados central, convertendo os arquivos para o formato da linguagem de hipertexto, ou html.

Em muitas das corporações multinacionais que dominam os processos químicos, as intranets assumem estruturas variadas. As grande organizações têm adotado alguma versão de arquitetura cliente/servidor, na qual computadores centralizados executam programas que o usuário individual intercepta para realizar o trabalho necessário. As intranets podem ser configuradas facilmente para serem adaptadas a essas redes cliente/servidor. Algumas empresas utilizam as intranets como interfaces para suas comunicações internas ou para sistemas de banco de dados. Na intranet da Monsanto, por exemplo, cerca de 3000 membros do quadro de pessoal possuem acesso local a correio eletrônico, bem como preços de estoque e dados do departamento de recursos humanos. A companhia Rohm & Hass possuia, no final de 1996, seus 12000 funcionários, ao redor do mundo, em 47 locais de produção conectados através de uma intranet. Um primeiro passo utilizado foi a distribuição de políticas e manuais dos departamentos ambientais, de saneamento e de segurança para todos os trabalhadores através da rede. A intranet utilizada pela Dow Chemical Inc. oferece um caminho direto para informativos da empresa, assim como compartilhamento de mensagens de correio eletrônico e documentos. Outras companhias pretendem ter seus sistemas intranet realizando outras tarefas além de simplesmente distribuir informação.

Mas o impacto da Internet reside em como os programas estão sendo projetados. Os produtos da empresa Gensym Corp. oferecem um bom exemplo. A companhia oferece "sistemas inteligentes" usados no controle supervisório de processos, diagnóstico de processos e otimização de projeto; o mais conhecido é o programa G2. Em maio de 1996, a Gensym apresentou dois produtos compatíveis com a Internet, o G2 WebLink e o G2 WebMiner. O WebLink permite que qualquer cliente autorizado, não apenas usuários G2, obtenha acesso às informações no G2 via Internet ou intranets.

Por exemplo, G2 tem ferramentas poderosas para construção de bancos de dados com dados de produção em tempo real, os quais podem ser visualizados por um gerente de produção utilizando o WebLink. O WebMiner permite que usuários possam colher dados da Internet ou intranets e colocá-los no G2 para uso posterior.

Programas de projeto auxiliado por computador (CAD) estão seguindo um caminho semelhante. A EA Systems desenvolve dois produtos, o PlantWeb e o PlantBrowser, os quais possibilitam que usuários recuperem dados de um arquivo de projeto, gerem um modelo 3D e o enviem para um usuário via Internet. Em 1996, a Rebis apresentou o PlantWorld e o PlatLife Browser, projetados para oferecer acesso compatível com a Internet para seus produtos CAD. O PlantWorld também permite que seus usuários componham arquivos em linguagem de realidade virtual (VRML) a partir de informação 3D.

Em junho de 1997, a Sgi-Cray anunciou que tinha se unido à Cadcentre Ltd. e ao Imperial College para o desenvolvimento de uma combinação hardware-software chamada Virtual Plant. Seus usuários podem visualizar imagens 3D de instalações químicas, as quais podem ser combinadas a dados em condições operacionais e variáveis de processo, entre outros.

No controle de processos, as companhias já estão direcionando a tendência de ferramentas industriais utilizando tecnologia Internet. A AccessWare Inc., por exemplo, fornecedora de programas e serviços de integração de sistemas para controle supervisório e aquisição de dados rescreveu seu módulo de produção para funcionar como um servidor Internet, permitindo que usuários equipados com um navegador comum possam ler dados dos banco de dados ViewPoint.

Em um futuro próximo, de acordo com a Sun, desenvolvedores de programas comercializarão applets Java ou partes de aplicativos. Ao invés de comprar programas completos, as companhias comprarão os applets que necessitam, e usarão a Internet para buscar e distribuir dados que desejem obter. Eventualmente, usuários alugariam applets e pagariam uma licença para o desenvolvedor (Basta e Kim, 1996). A próxima geração de navegadores Web, que já são compatíveis com as tecnologias Java e ActiveX – produto desenvolvido pela Microsoft Co. (www.microsoft.com) – , permitirão que usuários remotos se conectem e utilizem programas executando em servidores poderosos, acessando aplicações que não poderiam executar eficientemente em computadores locais. Muitos vendedores, incluindo fornecedores de programas para dinâmica de fluidos computacional, simuladores de processo e controle já estão trabalhando em protótipos para essas aplicações (Fouhy et al., 1997).

Muitos desenvolvedores de produtos nas áreas de controle de processos, projetos de instalações e simuladores estão também atentos para o uso dos padrões comuns Internet como um caminho para extrair dados de estruturas abertas que possam garantir o intercâmbio de informação entre empresas. A Honeywell Inc. lançou seu controlador híbrido PlantScape, um sistema orientado a objetos que integra sistemas de controle de processos. O produto oferece navegação segura e permite que operadores acessem documentos em uma intranet. Similarmente, a Fischer-Rosemount Systems Inc. restruturou sua tecnologia de controle introduzindo o PlantWeb, um sistema de gerenciamento de bens e mercadorias. O sistema utiliza comunicações modernas para elevar a "inteligência" nas áreas de projeto e distribuir informações de equipamentos para computadores espalhados pela empresa e utilizados por seus engenheiros.

Recentemente foi desenvolvido um programa cliente Web, denominado FactoryLink Enterprise Control System, que permite controle pleno ou acesso apenas para leitura de aplicativos servidores de instalações químicas utilizando o programa da empresa US Data Inc.. O programa usa as tecnologias de comunicação Internet-intranet e o acesso pode ser através de uma rede local, linha discada ou conexões Internet (Chemical Engineering, 1997b).

2.2.3. Comércio Eletrônico

Seguindo os avanços em ferramentas Web para projeto auxiliado por computador, controle de processos e planejamento de recursos da empresa, o comércio eletrônico (EC – Eletronic Commerce) está agora chamando a atenção de vendedores de produtos químicos e desenvolvedores de programas.

O comércio eletrônico é baseado em uma série de formulários computadorizados que automatizam transações comuns como pedidos de compra, faturas, notas de embarque e pedidos de cotação de preços. No passado, este sistema de intercâmbio de dados era chamado intercâmbio eletrônico de dados (EDI – Eletronic Data Interchange). Uma vez que o EDI elimina múltiplas cópias em papel, ele pode potencialmente reduzir custos de transações em mais de 90%. Nas indústrias de produtos químicos, o EDI pode reduzir em média em 40% o tempo de ciclo para funções do negócio como entrada de pedidos, produção, logística e transações financeiras. Estima-se que em 2002, mais de 300 bilhões de dólares de mercadorias e serviços serão comercializados pela Internet, contra os 8 bilhões de 1996 (Fouhy et al., 1997). Estima-se também que cerca de 17 milhões de dólares deste comércio será composto pela participação de fabricantes de papel, plásticos e outros produtos não-duráveis (Chowdhury, 1997b).

Atualmente, o comércio eletrônico é dominado pelos jogos, periféricos e programas de computador, CDs de música e serviços de notícias. Mas algumas indústrias de produtos químicos estão utilizando a rede também para realizar negócios e para comprar equipamentos. Não são os vendedores de programas simuladores, de projeto ou de controle que estão modificando as tendências no comércio via Internet. A área de comércio de equipamentos usados, por seus baixos custos, já é um sucesso comercial na indústria, com muitos revendedores de equipamentos operando sites modernos que são mais do que painéis de publicidade. A Louisiana Chemical Equipament Co. estima que 10% de suas vendas já venham de seu site Web. O custo para desenvolvimento e implementação, incluindo um banco de dados pesquisável foi de aproximadamente 12 mil dólares (Fouhy et al., 1997).

Recentemente a Air Products and Chemicals Inc. introduziu um serviço Internet vendendo gases especiais e equipamentos selecionados. A empresa de consultoria DeWitt & Co. desenvolveu um sistema Internet, denominado Chematch On Line, para unir compradores e vendedores de produtos petroquímicos em tempo real. O serviço funciona como uma fonte referencial e confiável de preços e focaliza-se principalmente no comércio de benzeno, tolueno, misturas de xilenos, metanol e éter metil terc-butílico (Chowdhury, 1997b).

O comércio Internet tem simplificado os negócios ao ponto de algumas das maiores companhias químicas estarem comercializando suas próprias instalações e equipamentos usados. A Union Carbide oferece uma instalação descomissionada e itens de equipamento individuais a partir de um banco de dados instalado em seu site Web, incluindo os preços dos equipamentos (Fouhy et al., 1997).

2.3. A LINGUAGEM JAVA E A ENGENHARIA QUÍMICA

Ao longo da última década os fabricantes têm investido em uma série de tecnologias para criar um infra-estrutura computacional que automatize várias fases do processo de desenvolvimento de seus produtos. O objetivo a longo prazo tem sido a criação de uma rede de computadores que permita a engenheiros, gerentes, e executivos a condução e gerenciamento do desenvolvimento de produtos como um processo unificado (Deitz, 1997). Na adoção dessas tecnologias, todavia, muitos fabricantes têm obtido sucesso na automatização de tarefas individuais, cada qual com uma série discreta de programas e tipos de dados associados a ela. No processo de criação de sistemas para interligação entre os processos, alguns pesquisadores e desenvolvedores de programas estão começando a visualizar um novo mecanismo, baseado na Internet, para intercâmbio e acesso a dados de engenharia.

Tal mecanismo poderia fornecer a infra-estrutura necessária para o gerenciamento do desenvolvimento de produtos como um processo comercial único.

A Internet é o maior fator nesse desenvolvimento, visto possibilitar um mecanismo de comunicação para empresas de virtualmente qualquer tamanho e complexidade. Pesquisadores de universidades e desenvolvedores de programas CAD estão explorando o potencial da utilização da linguagem Java para o desenvolvimento de consultores de projeto – programas que possibilitam assistência em alto nível na realização de projeto de processos específicos, tal como a criação de um molde. Esses programas podem ajudar engenheiros a ir além do emprego de sistemas CAD/CAM/CAE para manipular geometrias, usando-os para manipular e avaliar conceitos de projeto alternativos. O objetivo global desses esforços é o desenvolvimento de programas para engenharia orientados mais ao processo do que à tarefa.

A linguagem Java tem despertado interesse considerável entre desenvolvedores de programas, que vêem nela um mecanismo para a solução de um antigo problema da indústria. Os clientes idealmente querem programas que estejam personalizados para seus produtos e processos comerciais. A linguagem Java já está provando ser uma ferramenta capaz de estender as funções de programas existentes, de forma que os mesmos possam reunir mais amplamente as necessidades de um produto ou processo de engenharia em particular.

Os caminhos pelos quais applets Java podem realizar os benefícios da computação em rede envolvem o fornecimento de funções adicionais para redes de bancos de dados de produtos e a adição de novas capacidades às redes de computadores usadas por equipes de desenvolvimento de produtos. Alguns analistas industriais prevêem um mercado potencial para applets que funcionam desta forma, assim como para consultores de projeto ou manufatura baseados em linguagem Java. Enquanto alguns desses applets estão sendo desenvolvidos para trabalhar como uma característica integrada de programas CAD particulares, outros aproveitam-se do grau de independência de plataforma da linguagem Java para distribuir programas genéricos que funcionam com qualquer programa CAD.

Embora muitos observadores acreditem que é ainda muito cedo para dizer onde a linguagem Java será implementada mais eficientemente em ambientes de engenharia e produção, engenheiros em empresas privadas e em universidades já estão experimentando a nova tecnologia. A habilidade de Java executar virtualmente em qualquer plataforma é talvez a razão mais compulsiva para o intenso interesse de vendedores de programas e equipamentos computacionais na tecnologia. Java poderia ter um impacto significativo no processamento de transações, em aplicações norteadas pelo processo tais como

CAD/CAM, ou aplicações em processos industriais diretamente. Os desenvolvedores de programas poderiam definir suas aplicações oferecendo um conjunto básico de funções. Quando um usuário tivesse interesse em tentar uma nova função que ele não adquiriu inicialmente, poderia clicar em um applet Java embutido na aplicação e acessá-la dinamicamente (Deitz, 1996).

Além disso, uma massa crítica de applets Java para engenharia serão necessários para que a tecnologia possa ganhar mais ampla aceitação na comunidade da engenharia. A lista de desenvolvedores de programas interessados na aplicação de Java está crescendo a cada dia. Essas empresas incluem vendedores de CAD/CAM, programas para gerenciamento de documentos, para gerenciamento de dados de produção e para planejamento de recursos empresariais. Engenheiros em companhias privadas, universidades e aplicações de consultoria estão também entrando em ação.

2.3.1. Desenvolvimentos Java

Durante o ISA Show em Chicago, realizado em outubro de 1996, foi apresentada uma série de estratégias e produtos de controle de processos conectados através da Internet. A Sun planeja trabalhar com desenvolvedores de equipamentos e programas computacionais para lançar uma especificação para interfaces de programas Java. O resultado serão aplicações de controle de processo em tempo real que são independentes de plataforma, e podem rodar sobre intranets corporativas assim como sobre a rede. Entre as companhias que colaboram no projeto incluem-se AGG Systems Control, Elsag Bailey, Foxboro Co., Hewlett-Packard Co. e Honeywell Industrial Automation. Dois produtos lançados em 1997 utilizando a tecnologia Java permitem que usuários monitorem processos remotamente pela rede: a ferramenta Scout da Wonderware Corp., e o sistema WizBrawze da PC Soft International (Chemical Engineering, 1996).

Utilizando tanto Java quanto ActiveX, a Intuitive Technology desenvolve interfaces para grandes usuários da indústria de automação. A empresa utiliza a tecnologia cliente/servidor, com modelos de dados e métodos padrões ISA, permitindo o intercâmbio de informações através de múltiplas plataformas. O programa Web@aGlance mostra dados do sistema de informação do processo na Web, ou na intranet da companhia, em um formato tabular, gráfico e em páginas Web usando modelos padrão. A tecnologia usa tanto Java quanto ActiveX para distribuir gráficos DCS, HMI e SCADA para a Web. O custo da edição básica é de 399 dólares por servidor Web. A Biblioteca @aGlance/IT (Java) para Animação de Processo e o Editor de Animação Web@aGlance animam processos para navegadores Web, permitindo OEMs para desenhos herdados de clientes. O custo de animação varia de 899 dólares para 10 usuários em 24 horas para 3.495 dólares para uso ilimitado.

Mais de 130 companhias estão utilizando Web@aGlance, estando entre elas Dow Chemical (no mundo todo), Shell Oil (sites nos Estados Unidos), Mitsubishi Chemical (Japão), Eli Lilly (Inglaterra), Motorola, entre outros (Control Engineering, 1997).

A SoftPLC Corp. usa Java em seus programas SoftPLC e ViewPoint. A companhia selecionou uma implementação industrial da Máquina Virtual Java (JVM) chamada Mach J. Mach J permite ao programador da aplicação gerenciar diretamente a memória, evitando pausas intermediárias na execução do programa. Em geral seu desempenho é melhor do que a JVM da Sun, e pode ser usada na execução de ambientes não suportados pela Sun, como coprocessador e cartões de controle de movimento.

Em setembro de 1997, a Matra Datavision (www.matra-datavision.com) anunciou seu conjunto de programas CAD/CAM/CAE de última geração, o Euclid Quantum. Engenheiros podem projetar no Euclid Designer enquanto eles navegam na Internet em busca de uma componente de projeto, sem haver a necessidade de sair de uma programa e carregar outro. Foram utilizadas no desenvolvimento do programa as tecnologias Internet e Java para oferecer uma estrutura para gerenciamento de todos os dados associados com um produto, através de seu ciclo de vida completo (Deitz, 1997).

A Westinghouse Eletronic Corp. está introduzindo um navegador compatível com Java para visualização remota de diagramas, tendências e dados de sistema de controle de processo através da tecnologia intranet ou Internet (Chemical Engineering, 1997b).

Os desenvolvedores de programas da Bentley Systems Inc. têm investigado possibilidades de utilização da Internet, Java e Programação Orientada a Objetos para gerenciamento das conexões existentes entre documentos criados por engenheiros. A empresa licenciou a Máquina Virtual Java da Sun e incorporará a linguagem Java em seu produto MicroStation/J a ser lançado no início de 1998 (Chemical Engineering, 1997a). Com o programa MicroStation, engenheiros podem navegar na Web, criar modelos sólidos, e desenvolver conteúdo Web em um único pacote. Se houver a necessidade de inclusão de um componente de um fornecedor externo em seu projeto, o usuário pode usar seu navegador para encontrar o componente, transferir o arquivo CAD do vendedor e então publicar aquele arquivo com a definição do produto em uma página Web do projeto. Com um navegador inteligente, pode-se realizar buscas para encontrar a interação mais atualizada de um projeto ou estar certo de ser avisado quando um projeto for alterado. Recentemente, o programa MicroStation foi integrado com a tecnologia FastTrack Server e Enterprise Server da Netscape Communications. O produto resultante – uma infra-estrutura de rede chamada Engineering Back Office – conecta dados

em grande escala pela integração de plataformas de engenharia e dados com bancos de dados corporativos e outros sistemas de informação empresariais (Deitz, 1997).

A Bentley anunciou recentemente o lançamento do programa ModelServer TeamMate, o qual fornecerá ferramentas servidoras de gerenciamento de documentos de engenharia e fluxo de trabalho. A empresa está trabalhando também na especificação Open Engineering Connectivity, um conjunto de interfaces de programação (APIs) abertas, baseadas em rede, para todos os produtos ModelServer. Com as APIs, outros desenvolvedores de programas e os clientes da empresa estarão aptos a personalizar aplicações clientes ModelServer existentes ou criar novas aplicações usando ferramentas Java.

O desenvolvimento de applets em linguagem Java específicos para Engenharia Química em especial pode ser considerado reduzido até o momento. A maior fonte de consulta mundial sobre a utilização da linguagem Java nas mais variadas áreas, disponível através da rede Internet, é o site Web da Gamelan (www.gamelan.com). Nesse site pode-se ter uma avaliação atualizada e recente da linguagem Java em várias áreas de aplicação. O desenvolvimento de produtos de interesse para a comunidade científica representa uma das áreas de maior expansão e com maiores perspectivas de crescimento. A estrutura na qual os aplicativos desenvolvidos por pesquisadores de vários países pode ser generalizada em cinco áreas principais: Engenharia, Matemática, Física, Química e Biologia. Há que se destacar a pequena participação da comunidade científica brasileira nestes repositórios. Um sumário do desenvolvimento no mundo nos campos de Engenharia e de Química é apresentado a seguir.

2.3.2. Aplicativos em Java para Engenharia

O repositório Gamelan apresenta em seu site Web 136 applets categorizados na área Engenharia (Developer.com, 1998a). O primeiro applet foi cadastrado em 12/10/95, consistindo de uma ilustração de transistores e circuitos MOS e de lógica discreta. Entre os trabalhos destacados pelo site, alguns são descritos a seguir.

Kazmer (1996), da Universidade de Massachussets, desenvolveu o que pode ser o primeiro applet Java de engenharia, um estimador de custo para peças moldadas por injeção. O applet Injection Molding Cost Estimator, Figura 2.2, permite que usuários entrem com o tipo do material, os ciclos de produção, o número de cavidades por molde e a complexidade da peça. O applet estima os custos de processamento, materiais e ferramental. Pode-se alterar o comprimento, largura ou peso da peça, enquanto uma imagem da peça é atualizada dinamicamente em uma janela. O applet também ajuda na

investigação do impacto causado no tempo de ciclo pela alteração da espessura da peça ou das aberturas.

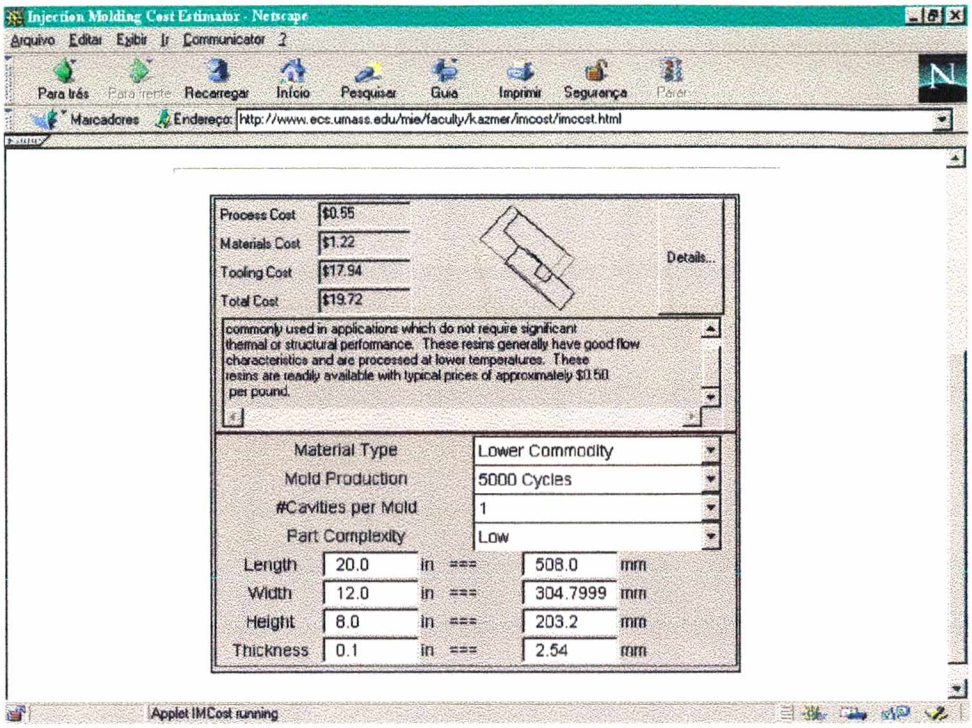


Figura 2.2 – Applet Injection Molding Cost Estimator

Oh (1996), do Computational Aerospace Sciences Laboratory, MIT, desenvolveu um applet para um túnel de vento virtual em Java, empregando uma simulação de dinâmica de fluidos computacional bidimensional. O Java Virtual Wind Tunnel, Figura 2.3, é um applet que usa métodos da dinâmica de fluidos computacional (CFD) para simular o fluxo de ar sobre um objeto bidimensional. O applet resolve as equações de movimento (equações de Euler) em tempo real. Apesar de, segundo o autor, se tratar apenas de um protótipo, mostra claramente como a linguagem Java pode ser usada para construir ferramentas educacionais e de pesquisa para cursos de mecânica de fluidos de graduação e pós-graduação. A nível de graduação, as simulações podem ajudar estudantes a visualizar e entender fluxo de fluido 2D; a nível de pós-graduação, ele pode ajudar estudantes a entender o poder e limitações da dinâmica de fluidos computacional. A demonstração mostra o fluxo de ar através de um canal com o impacto em uma das paredes. Uma onda de choque aparecerá sobre um impacto onde o fluxo salta da velocidade supersônica para subsônica. O movimento dos "sliders" altera as condições de fluxo e causa o aparecimento/desaparecimento do choque.

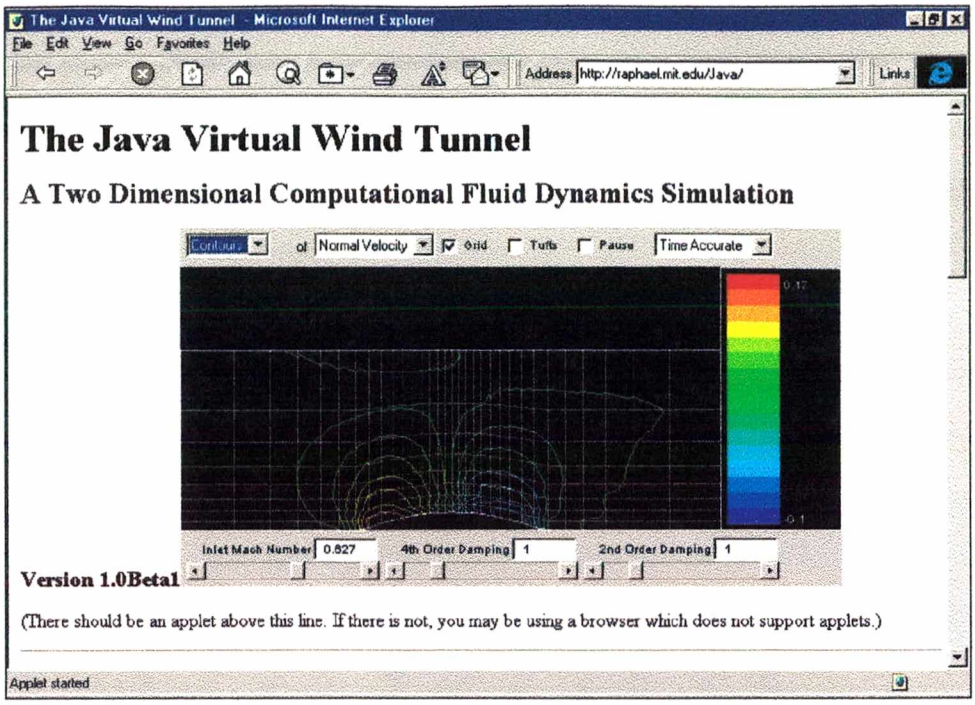


Figura 2.3 – Applet Java Virtual Wind Tunnel

Devenport (1996), do Department of Aerospace and Ocean Engineering, Virginia Tech, desenvolveu um applet para uma máquina de fluxo ideal, como um applet Java educacional para estudo de fluxo elementar ideal. O Ideal Flow Machine, Figura 2.4, é destinado a estudantes aprendendo fluxo ideal elementar. O termo 'fluxo ideal' descreve o caminho no qual um fluido (líquido ou gás) move-se quando os efeitos de compressibilidade e viscosidade são desprezíveis. Fluxo ideal é geralmente o primeiro tipo de movimento de fluido estudado por estudantes de engenharia, devido à sua simplicidade. Grandes partes do fluxo em navios, submarinos, carros e aviões leves são ideais. Este applet é projetado para oferecer aos estudantes um ambiente onde eles possam experimentar e visualizar fluxos ideais elementares bidimensionais e desta forma melhor entendê-los. Fluxos ideais são soluções para a equação de Laplace. Esta equação diferencial é linear, o que significa que adicionando em conjunto (superpondo) qualquer número de fluxos ideais produz um novo fluxo ideal. Uma técnica para se encontrar a solução para problemas de fluxo complexo é iniciar com fluxos ideais muito simples, que são facilmente entendidos e descritos, e então adicioná-los para produzir as características do fluxo complexo desejadas. Este é exatamente o processo modelado na Máquina de Fluxo Ideal.

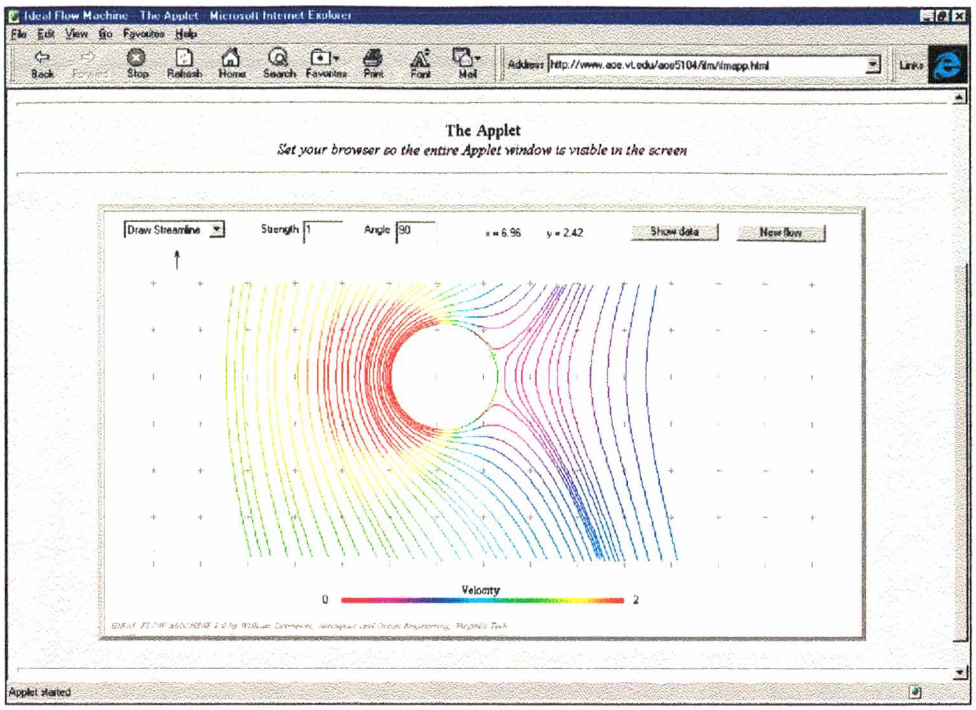


Figura 2.4 – Applet Ideal Flow Machine

Bhattacharjee (1996), do Mechanical Engineering Department, San Diego State University, desenvolveu o applet Band Radiation Deamon, implementando um calculador de radiação gasosa infravermelha, Figura 2.5. O cálculo de radiação térmica , que envolve gases como CO_2, H_2O, CO, é geralmente abordado em aulas de Radiação Térmica no nível de graduação em Engenharia Mecânica. Este applet emprega correlações exponenciais de banda larga para o cálculo de emissão de uma coluna isotérmica de várias espécies. O usuário fornece a temperatura, tamanho do caminho, pressão parcial das espécies e a pressão total, e seleciona uma banda particular de uma lista fornecida pelo applet. A radiação da banda é calculada e exibida.

Young e Szollar (1997), do Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, desenvolveram o applet em Java Anneal-O-Matic. O applet Anneal-O-Matic, Figura 2.6, cria uma distribuição randômica de 100 células conectadas por uma malha randômica. Cada célula possui duas portas de entrada e uma porta de saída. Cada porta de saída é conectada a duas portas de entrada. Cada porta de entrada, é conectada a apenas uma porta de saída. Este arranjo é então usado como um teste de caso para demonstrar recozimento simulado e corte mínimo, dois algoritmos comumente utilizados em CAD eletrônico para otimizar a distribuição de células. O applet Anneal-O-Matic demonstra esses algoritmos permitindo ao usuário interativamente aplicá-los ao arranjo e visualizar seu efeito através de uma interface inovadora com o usuário. O applet se propõe a

ser uma demonstração da utilidade tanto das ferramentas de visualização bem como dos próprios algoritmos.

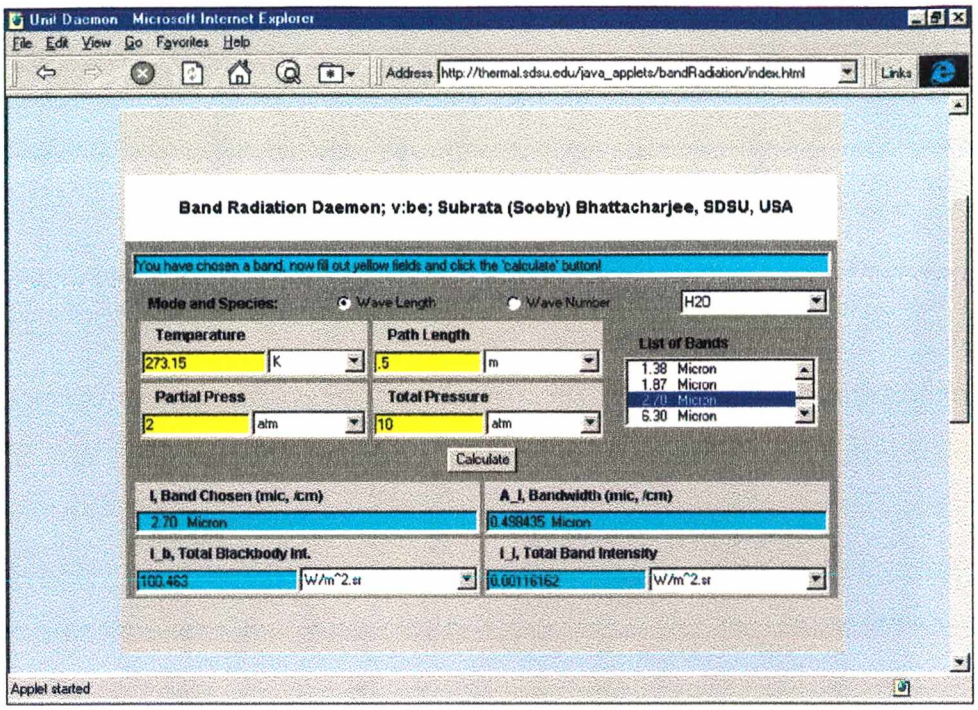
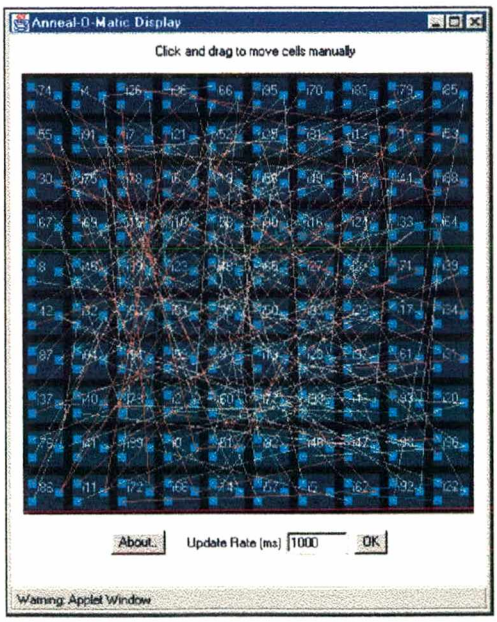
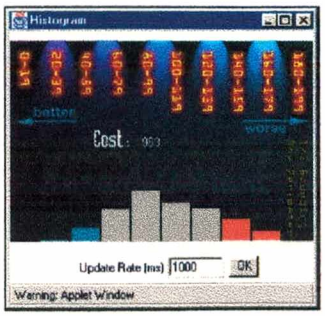


Figura 2.5 – Applet Band Radiation Deamon



(a) Monitor do Anneal-O-Mastic



(b) Histrograma do programa Anneal-O-Matic

Figura 2.6 – Applet Anneal-O-Matic

Connolly (1997), da University of Massachusetts, e do SRI International, Califórnia, desenvolveu um applet Java para a simulação dinâmica de um braço biarticulado. O applet Dynamic Simulation of a 2-Link Arm é mostrado na Figura 2.7. A direita do applet é exibido um gráfico mostrando uma seção de superfície com energia no espaço fase (ângulos de junção mais velocidades de junção) no qual a ligação está se movendo. A abscissa do gráfico representa o ângulo da segunda junção, enquanto a ordenada representa a velocidade angular daquela junção. Um ponto é graficado sempre que a primeira junção passa 1,5708 radianos na direção horária. Pode-se obter ainda um gráfico acelerado da seção plana de fase. Diferentes características podem ser obtidas variando os comprimentos das ligações e os ângulos iniciais das junções. Como uma modificação deste applet, foi adicionado um motor nas junções juntamente com um laço servo resultando em um braço de posição controlada. Esta simulação implementa um controlador PD, com um laço servo executando a 200Hz.

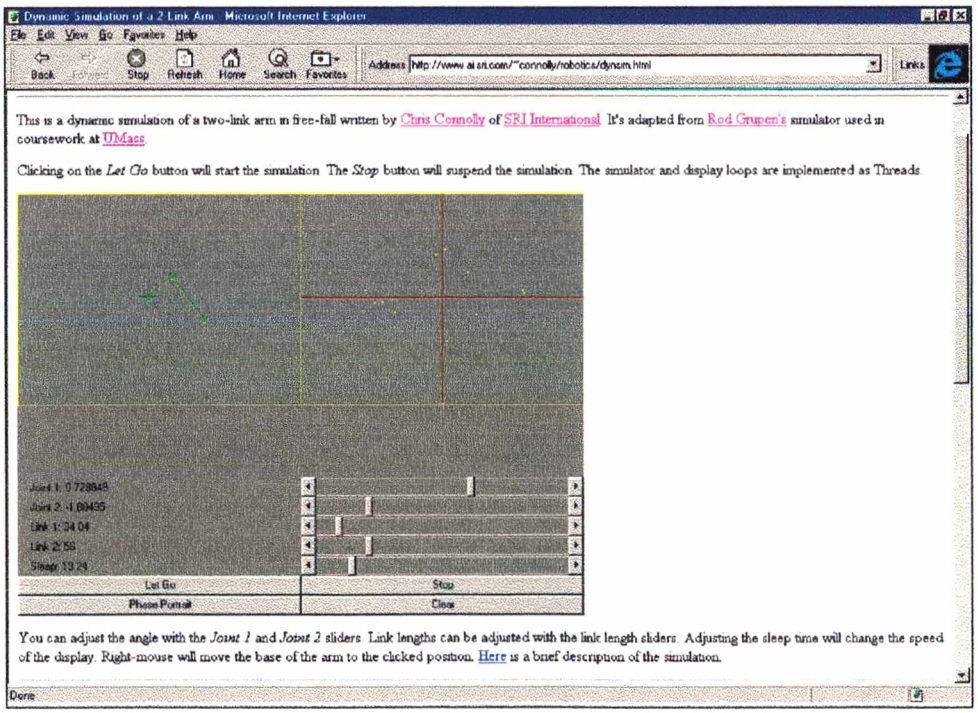


Figura 2.7 – Applet Dynamic Simulation of a 2-Link Arm

Van Rienen (1997), desenvolveu um aplicativo Java denominado DigSim. DigSim V2, Figura 2.8, é um simulador digital profissional de alta velocidade escrito em código nativo para as plataformas Windows NT e 95. Muitas falhas de simulação foram corrigidas em sua última versão devido à utilização de uma ferramenta completamente nova de simulação de alta velocidade. Dezenas de funções foram utilizadas, e centenas de componentes foram incluídos.

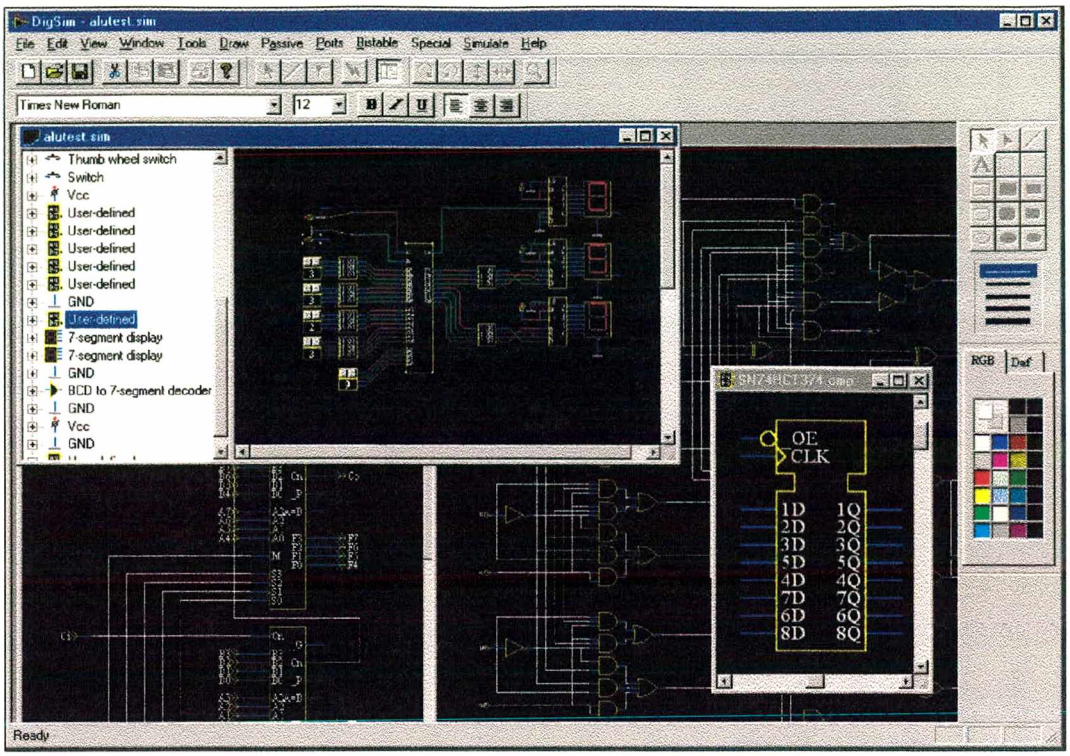


Figura 2.8 – Aplicativo DigSim V2

2.3.3. Aplicativos em Java para Química

O repositório Gamelan apresenta em seu site 42 applets cadastrados na área Química (Developer.com, 1998b). O primeiro applet foi cadastrado em 03/10/95, consistindo de um renderizador de modelos moleculares 3D. Alguns dos trabalhos relacionados no site Web são descritos a seguir.

Grayce (1996), do Chemical Departament da University of California at Irvine, desenvolveu um applet para realizar a simulação de caixa elástica que ilustra a Segunda Lei da Termodinâmica. Manipulando-se os parâmetros da simulação pode-se ilustrar os fatos fundamentais envolvidos no fluxo de energia. A Figura 2.9 mostra a applet The Second Law em execução.

O applet Steam Properties Calculator, Figura 2.10, desenvolvido pela Creative Engineering Software Solutions (1997), se destina a fornecer as propriedade de transporte e termodinâmicas de vapor. Os resultados são baseados na formulação IFC 1967 e são consistentes com as tabelas de vapor ASME. Através do fornecimento do valor de duas propriedades conhecidas do vapor d'água, podem ser obtidas as demais propriedades. Para calcular propriedades de saturação, deve-se informar a qualidade e a temperatura ou a pressão.

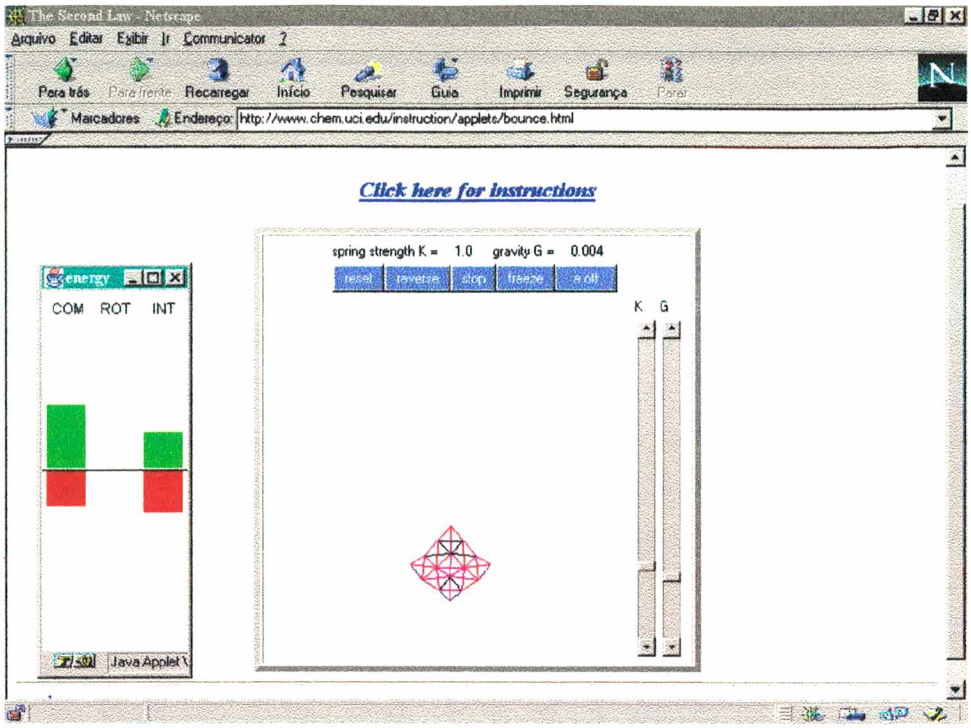


Figura 2.9 – Applet The Second Law

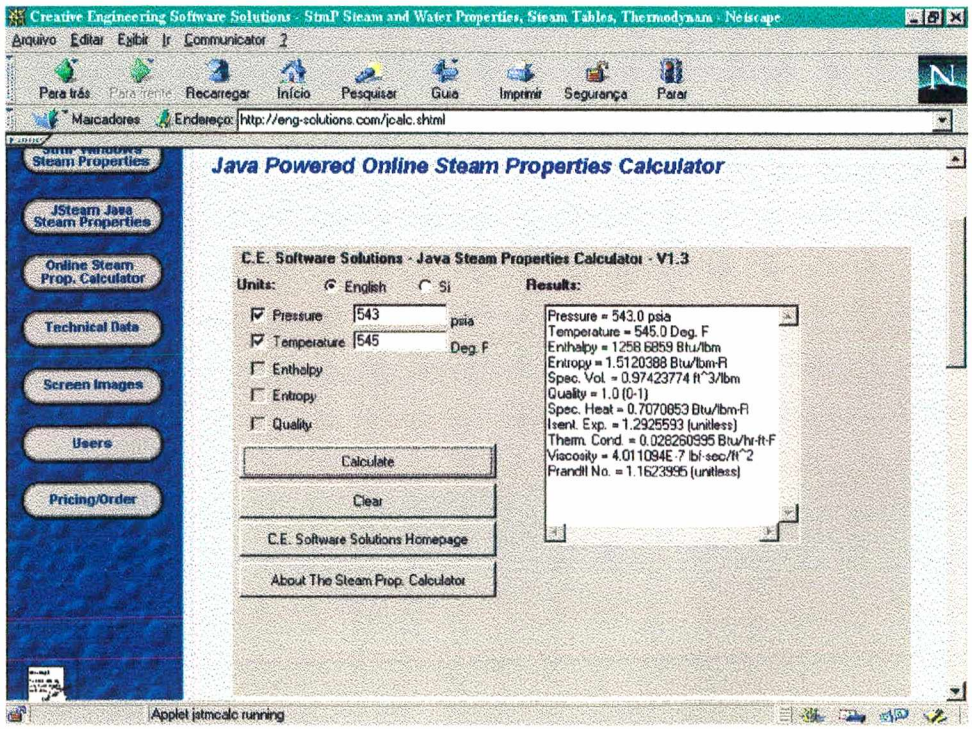


Figura 2.10 – Applet Steam Properties Calculator

Wieser (1997), da Octavian Micro Development Inc., desenvolveu o applet TBS-EOS Calculator, o qual permite calcular os fatores de compressibilidade, Z, usando a equação de estado cúbica de Trebble-Bishnoi-Salim. Os dados de temperatura, pressão e volume críticos, fator acêntrico e massa molar são obtidos de um banco de dados. Após o usuário especificar a temperatura e pressão nas quais deseja-se calcular as frações de líquido ou vapor, o applet fornece os valores de Z necessários para determinação do volume. Outro applet Java desenvolvido pelo mesmo autor, denominado Mass Calculator, permite o cálculo da massa de uma molécula.

Moros (1997), do Institute of Technical Chemistry, University of Leipzig, desenvolveu uma simulação interativa do comportamento de tanques agitados em série com relação ao tempo de residência. A mudança de concentração do traçador é mostrada em uma animação simples, assim como em um diagrama concentração vs. tempo. Todos os parâmetros podem ser alterados e uma barra de botões é usada para controlar o experimento. São incluídas informações de referência sobre a Teoria da Distribuição de Tempo de Residência e links para sites relacionados. Essa aplicação inclui a utilização de applets Java e da linguagem Javascript. A Figura 2.11 mostra a aplicação Residence Time em execução.

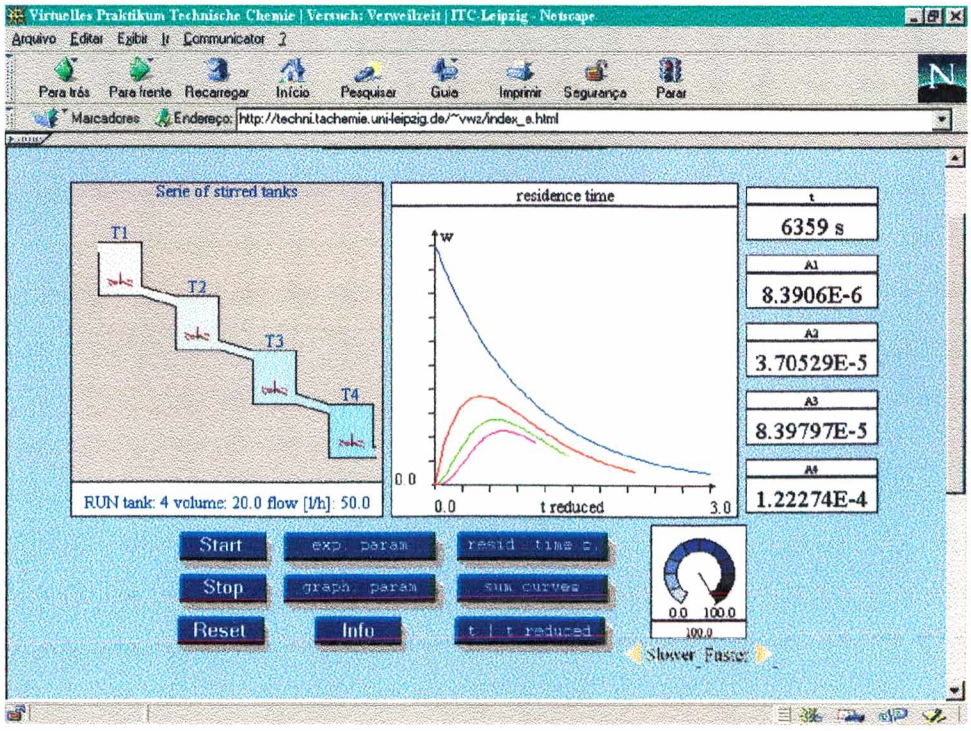


Figura 2.11 – Applet Residence Time

2.4. PACOTES COMERCIAIS PARA ENGENHARIA QUÍMICA

2.4.1. Pacotes de Ensino

Um trabalho realizado na Universidade da África do Sul (Steyn et al., 1996) analisa as vantagens e desvantagens dos pacotes de ensino atualmente disponíveis no mercado. Uma série de aulas auxiliadas por computador (CAL – Computer Aided Learning) para o primeiro ano de Química Analítica foram ministradas nesta universidade e tornadas disponível para os estudantes como uma atividade opcional de ensino. A avaliação do curso foi realizada no ambiente instrucional presencial para obtenção de informações sobre a eficiência desta técnica na instrução do ensino à distância na África do Sul. Um percentual relativamente baixo (15%) dos estudantes que submeteram indicações utilizaram o curso, embora uma quantidade consideravelmente maior (28%) dos estudantes reclamaram sobre o acesso a computadores adequados. Razões para o baixo percentual de uso são principalmente devidas a fatores financeiros e limitações de tempo. Embora os resultados de estudantes que realizaram o curso tenham sido significativamente melhores somente ao nível de confiança de 80%, os estudantes responderam positivamente e solicitaram maior contato com CAL. Torna-se claro que os estudantes devem ter a possibilidade de acessar aulas livremente para garantir seu benefício máximo. A implementação de um meio interativo para educação requer um aumento nos níveis de educação em computadores pelos estudantes, assim como a garantia de igual acesso a todas as particularidades do processo de instrução.

2.4.2. Simuladores de Processos

A simulação de processos tem se tornado uma das tarefas rotineiras realizadas por engenheiros em indústrias de processos químicos. Vários pacotes computacionais permitem que operadores de processo, técnicos e supervisores tomem decisões a qualquer momento sobre um processo baseados nos resultados de simulação exibidos graficamente em uma tela de computador. A simulação de processos tem mudado fundamentalmente não somente a forma como a engenharia é praticada no chão da fábrica, mas também o trabalho das equipes de projeto (Glasscock, 1994).

Ao contrário de muitos outros tipos de aplicações, que necessitam de cientistas, engenheiros de construção ou projetistas de sistemas elétricos, a simulação de processos tem sido realizada por e para engenheiros químicos. De uma base limitada de usuários nos anos 70, foi rapidamente adaptada para o computador pessoal, e é agora amplamente utilizada por engenheiros de projeto e produção, assim como por pesquisadores e estudantes (Basta, 1996).

A simulação de processos está substituindo experimentos em escala piloto com vantagens em muitos casos, e executando uma função complementar em outros. Para processos envolvendo sólidos e fluxos de reciclo, por exemplo, plantas-piloto podem ainda ser necessárias. Mas a simulação de processo constitui uma parte importante do planejamento de tais experimentos piloto.

A modelagem de processos reduz o ciclo de desenvolvimento de novos processos e produtos através da identificação de problemas e oportunidades nos estágios iniciais da pesquisa. Nos projetos de processo, pode-se reduzir o custo de um novo processo por milhões de dólares permitindo que engenheiros simplifiquem sistemas, avaliem estratégias alternativas de controle e tentem técnicas modernas de separação para reduzir ou eliminar desperdícios.

Mais do que isso, engenheiros estão usando os atuais programas de simulação para determinar problemas de operabilidade devido a mal funcionamento de equipamento, e para eliminação de gargalos de um processo. Mesmo que melhoras incrementais nos projetos e operações sejam relativamente fáceis de se obter, a melhoria fundamental de processos químicos é complicada por uma série de fatores. Os maiores obstáculos para o progresso são: dados de entrada pouco confiáveis e escassos, o grande número de variáveis de processo, e as interações altamente não-lineares entre as variáveis.

A maior mudança nos simuladores atuais reside principalmente nas novas interfaces com o usuário, as quais permitem que engenheiros facilmente construam, modifiquem e executem um modelo de processo. Outro aspecto dos simuladores de processo atuais é sua "robustez". As ferramentas são equipadas com melhores algoritmos para permitir que uma classe mais ampla de usuários possa manejar problemas que estavam antes apenas reservados a especialistas. Os pacotes atuais usualmente possuem séries pré-construídas de unidades de operação que podem ajudar usuários inexperientes a obter convergência e soluções rápidas.

Nos primórdios da simulação de processos, a geração de resultados freqüentemente levava horas e às vezes dias. Como resultado, a modelagem estava limitada a questões que não necessitavam respostas rápidas. Atualmente, os resultados de simulação estão disponíveis a tempo de resolução de um mal funcionamento da unidade. Os problemas podem ser rapidamente entendidos observando as curvas de resposta graficadas em uma tela de computador. A alta velocidade de computação e a existência de interfaces amigáveis são o coração do novo paradigma da simulação, porque agora eles permitem que modelos possam ser usados para resolver problemas difíceis durante a fase de projeto do processo em questão.

Finalmente, programas estão sendo rescritos em linguagens orientadas a objetos, como C++ e Java. Essas linguagens permitem aos desenvolvedores reutilizarem objetos como módulos adicionais, aumentando significativamente a confiança do código e reduzindo o tempo de desenvolvimento.

O mercado global de simulação, que também inclui simuladores para treinamento e validação de sistemas de controle, totalizou em 1997 487 milhões de dólares anuais, sendo que a previsão para o ano 2001 é de 1 bilhão de dólares (Basta, 1997).

Segundo dados divulgados pela Automation Research Corp. na Chemical Enginnering Chemputers Conference 5 (New Orleans, fev 1996), a simulação para projetos de engenharia (objeto tradicional da simulação de processos) alcançou 150 milhões de dólares em todo o mundo em 1996, com crescimento anual de 10,9%. Já o mercado de simulação baseada em algoritmos de controle (MPC - model-based process control) alcançou 240 milhões de dólares, a taxa de crescimento de 22,5%.

Outra área em rápida expansão para a simulação com objetos, adotada como um dos objetivos do consórcio CAPE-Open, um grupo empreendedor de companhias químicas, de engenharia e de software da Europa que por vários anos trabalha para desenvolver programas simuladores abertos, a fim de se obter melhor interoperabilidade. O consórcio CAPE-Open foi consolidado pela União Européia no início de 1997, e tem como objetivo manter contratos com desenvolvedores de protocolos não-proprietários para o desenvolvimento de programas de engenharia que fazem uso da tecnologia de objetos.

2.4.2.1. Simuladores para Treinamento

Está se tornando cada vez mais fácil construir modelos de processos não somente para projeto, mas também para treinamento realístico de operadores de instalações químicas. O treinamento baseado em computador melhora as operações da instalação química, através da melhoria na resposta do operador a mudanças no processo, reduzindo o risco de emergências e reduzindo tempos de entrada em operação para unidades novas e existentes de processo (Winter, 1996). Uma grande melhoria nos simuladores de treinamento é alcançada conectando-se diretamente o modelo ao console DCS (Distributed Control Systems). O benefício chave é o realismo obtido ao fazer o operador utilizar o mesmo equipamento que utiliza para interagir com a unidade real.

Um bom exemplo da aplicação desse tipo de simuladores é a experiência observada na empresa Conoco Ltd., que durante os anos 80 usou simuladores para treinamento de operadores em vários lugares, incluindo sua refinaria na Inglaterra. Os simuladores eram baseados primariamente em

hardware genérico e modelos de simulação generalizados. Mesmo sendo úteis, os simuladores não representavam adequadamente o processo ou os instrumentos da sala de controle. Em 1990, o sistema de treinamento foi atualizado para suas unidades de craqueamento catalítico fluidizado (FCC), controlado por DCS, e de distilação de óleo cru (CDU). Em 1993, a empresa adicionou sistemas de treinamento para a planta de craqueamento gasoso e na unidade de mistura de gasolina. No final do mesmo ano, sistemas adicionais de treinamento foram adicionados para o complexo de destilação de óleo cru e o coqueificador de retardamento. Em algumas unidades, os engenheiros de projeto da Conoco também utilizam os simuladores para validar cálculos e configurações do DCS. Com esta ferramenta, pode-se eliminar erros de projeto e identificar oportunidades de melhorias bem antes da entrada em operação da instalação. A refinaria selecionou um simulador personalizado da empresa Honeywell Hi-Spec Solutions, Inglaterra. Com o uso dos simuladores de treinamento, muitas unidades da Conoco reduziram tempos de entrada em operação em 60-70% após paradas programadas. Em muitos casos, entradas em operação para novas unidades, que anteriormente necessitavam de 1 a 2 semanas, agora podem ser completadas em apenas 2 a 3 dias. Além disso, os simuladores de treinamento possibilitam que o tempo necessário para treinamento de novos operadores seja reduzido em mais de 50% (Winter, 1996).

O sistema Trainer, aplicado nas unidades da Conoco, inclui consoles de operador especialmente projetados que são idênticos aqueles utilizados na sala de controle do processo sendo simulado. Um modelo matemático representa parte ou todo o processo em estudo. Pelo interfaceamento com as facilidades do controle DCS, o modelo do processo pode ser configurado usando dados de controle e de engenharia detalhados, em tempo real. O programa de simulação utiliza esses dados – em conjunto com os algoritmos de controle usados pelos controladores do processo – para criar um modelo que reflita o comportamento atual e dinâmico do processo. Usando um console independente, os instrutores podem selecionar condições específicas de operação, definir exercícios de treinamento, especificar mal funcionamentos, e monitorar o desempenho do operador. Rotinas típicas de treinamento incluem entrada em operação, operação normal, operação com mudanças no processo, paradas de rotina e de emergência, e procedimentos para otimização do processo.

2.4.2.2. Simuladores de Processo Integrados

Nos últimos anos, muitos engenheiros nas áreas de projeto e controle de processos aguardavam ansiosamente por um novo tipo de produto de simulação: um que combine as operações em estado estacionário e dinâmico. Por décadas, sistemas em estado estacionário têm sido o estado da arte nessa área (Basta, 1997).

Ao longo dos últimos seis anos, no entanto, a evolução mais significativa tem sido a substituição da simulação em estado estacionário pela simulação dinâmica. Cada vez mais os desenvolvedores líderes na simulação estão investindo no uso de suas capacidades dinâmicas para otimização de processo online (Basta, 1996).

Os sistemas integrados representam atualmente o estado da arte em simulação de processos. Com mínimo esforço, projetistas de processo podem agora alternar de um tipo para outro, evitando a redigitação de dados, e acelerando o processo pelo qual um projeto inicial (geralmente no formato estado estacionário) avança para o melhor caso em uma otimização dinâmica.

A simulação dinâmica permite ao projetista testar um processo sob estado não-estacionário, rapidamente alterando condições, tal como entradas em operação e paradas. Mas mais importante, abre a porta para a utilização de programas simuladores com modelos para realização de controle do processos. Tal "controle de processo baseado no modelo" (MPC) tem sido considerado por vários anos, mas com simuladores baseados em algoritmos de controle, não na termodinâmica fundamental ou balanços de massa de processos químicos. O mercado para simulação MPC está agora em franca evolução.

Atualmente, três empresas dominam o campo da simulação integrada. Cada uma tem aprimorado características específicas em seus últimos produtos. Esses vendedores têm expandido a quantidade de ofertas disponíveis para as necessidades de projeto e controle.

– Aspen Technologies Inc. – O mais novo sistema apresentado por essa empresa no mercado de simulação é o DynaPlus. O programa combina dois produtos existentes – AspenPlus, um pacote de estado estacionário, e SpeedUp, um pacote dinâmico – com um "ambiente integrado" que trata fluxo de dados de um sistema para outro. A empresa tem atualizado significativamente os métodos de estimação de propriedades físicas de seus programas. Além deste sistema, a Aspen disponibiliza a seus clientes uma variedade de ferramentas para otimização e controle online (RT-Opt e DMCPlus), modelagem para destilação (Split), projeto de processo polimérico (Polymer Plus) e processos de síntese (Advent). A empresa possui ainda alianças com, entre outros, Gensym Corp., para modelagem de sistemas especialistas, e Hyperion para treinamento de operadores.

– Simulation Sciences Inc. – A empresa lançou recentemente seu produto dinâmico no mercado, denominado Protiss. Em 1995, a SimSci introduziu uma interface sofisticada, denominada ProVision. Essa interface fornece acesso tanto para o produto Protiss, quanto para o produto da empresa para

estado estacionário, o Pro/II. O ProVision é definido pela empresa como possuindo uma interface não-proprietária para visualização de processos e tem se tornado uma das especificações técnicas disponíveis para a indústria. A SimSci diz que seu enfoque é baseado em componentes, e que consegue a unificação de todos os seus produtos sobre uma mesma interface. A SimSci enfatiza a habilidade da empresa em modelar muitas, se não todas, as facilidades de uma grande refinaria ou indústria química. No entanto, o sistema necessita de uma workstation Unix de alto desempenho para sua operação. Um versão para plataforma PC, rodando Microsoft Windows está em desenvolvimento. Outros produtos da empresa incluem o ROM (Rigorous Online Modeling) e o Pipephase (para análise de tubulações). O programa de desenvolvimento Romeo, configurado em 1996 em trabalho conjunto com a Shell Development Co., tem o objetivo de produzir um modelador em tempo real baseado em equações, para aplicações online.

– Hyprotech Ltd. – A empresa é atualmente a terceira maior do mercado e foi a primeira a apresentar, em 1995, um produto combinado para estados estacionário e dinâmico, denominado Hysys. Seu sistema foi recentemente atualizado para a versão 2.0. O programa foi desenvolvido em código compatível com Microsoft 32 bits, e utiliza os padrões Microsoft OLE para a comunicação entre aplicações PC. Através do Hysys é possível aos usuários converter um projeto em estado estacionário para dinâmico com um simples pressionar de teclado. A Hyprotech está ampliando a quantidade de produtos disponíveis no mercado, e está empreendendo vigorosamente uma arquitetura baseada em componentes. Seus produtos estão baseados em uma estrutura na qual as funções mais importantes de um simulador podem ser organizadas em 20-30 componentes, os quais são utilizados tanto em novos projetos quanto em projetos existentes. Entre os novos produtos estão o Hycon, um programa para processo de síntese, e o Polyred, um simulador de polimerização, e produtos para projeto conceptual e interfaceamento com sistemas de controle. Além disso, a empresa vem firmando recentes alianças para o desenvolvimento de produtos com a CAE Eletronics, uma empresa de simulação para treinamento, com a Pavilion Technology, um desenvolvedor de rede neural, e com a MDC Consultants, para otimização e controle de processo em tempo real.

Nenhum desses ambientes integrados é de baixo custo – licenças iniciam em torno de 20.000 dólares por ano, e aumentam dependendo das opções escolhidas. E a simulação dinâmica não é para qualquer um – muitas funções de engenharia podem ser feitas com um produto em estado estacionário, e trabalho dinâmico demanda considerável especialização.

2.4.3. Pacotes Matemáticos e Estatísticos

Entre o grande poder de computadores pessoais e a habilidade com a qual os programadores estão desenvolvendo programas, os pacotes matemáticos e estatísticos perderam sua tradicional aura de serem chatos, difíceis de usar e lentos (Basta, 1996). Os fatores que mais influenciaram nesta evolução foram o aparecimento de interfaces gráficas amigáveis e a grande redução no preço de programas para computadores pessoais, já sendo possível encontrar programas totalmente gratuitos e de grande utilidade. O precursor nesta área foi um desenvolvedor de programas, a DSP Development Co. (www.dadisp.com), que no início de 1996 anunciou a disponibilidade de uma edição para estudantes de seu programa baseado em planilha eletrônica, denominado DADisp, para análise de dados, estatística, gerenciamento de dados e análise da transformada rápida de Fourier. Apesar da versão profissional ser significativamente mais poderosa, este programa oferece capacidades extensíveis. O programa pode ser obtido diretamente do site Web da empresa.

Uma inspeção dos vários pacotes disponíveis atualmente mostra três particularidades nas quais os programas matemáticos e estatísticos podem ser avaliados: a variedade de seus algoritmos, a quantidade e qualidade de seus métodos para exibição de dados de acordo com métodos estatísticos padrão, e a "abertura", definida pelo sistema operacional utilizado e as conexões com outros programas, como saídas para sistemas de controle. Enquanto muitos programas são caracterizados como de propósito geral – significando que eles podem ser usados para praticamente qualquer problema de análise de dados – muitos deles são dirigidos a uma aplicação específica: pesquisa, controle de produção, controle de qualidade ou análise de negócio, entre outros.

Um dos últimos programas desenvolvidos, o TK Solver, com interface para MS-Windows 3.1, é altamente orientado para resolução de problemas de engenharia, ao invés de pesquisa matemática. Problemas abordados tipicamente são aqueles envolvendo sistemas de equações não-lineares, e ajuste de curvas. Uma das características do programa é a habilidade de iniciar com um conjunto de pontos e expressões de contorno e projetar um conjunto de equações que produz tais pontos finais.

Outro programa de ajuste de curvas, o TableCurve 3D, pode armazenar o registro para o maior número de equações pré-definidas que ele trata: 450 milhões. O programa é disponível em uma versão 32 bits de seu desenvolvedor, Jandel Scientific Software. Após os dados terem sido inseridos, o programa gera superfícies gráficas das soluções que melhor se ajustam aos dados. O usuário pode alternar entre as soluções propostas, decidindo qual deseja investigar em maior detalhes.

Outro programa possui uma técnica mais objetiva para análise estatística. A Stat-Ease Inc. oferece o programa Design-Ease para problemas de projeto de experimentos, e o Design-Expert para controle estatístico de qualidade. Uma metodologia para o último pacote tem sido trabalhada especificamente para o "projeto de mistura para formulações ótimas", que é como a Stat-Ease caracteriza o problema associado com processamento de produtos misturados ou formulados. Ambos os pacotes executam em computadores Windows e Macintosh.

O produto de peso entre pesquisadores para análise matemática é definitivamente o Mathematica, da Wolfram Research Inc. A última versão do programa, a versão 2.2, está disponível numa grande variedade de sistemas operacionais, incluindo Unix, DOS e Windows, e Macintosh System 7. Ao lado de ajuste de superfícies, resolução de equações diferenciais e outras ferramentas matemáticas padrão, o programa possui um grande número de "pacotes" (módulos), oferecendo funcionalidade especializada para projeto mecânico, lógica fuzzy e séries temporais.

MathCad+, agora disponibilizado pela MathSoft Inc. na versão 6.0, é outro pacote amplamente utilizado. Sua nova característica é de interação com o Visio Express, um componente do pacote gráfico Visio, para produção de diagramas e fluxogramas que podem ter funções matemáticas embutidas. A utilização do padrão Microsoft OLE (Object Linking Embending) é o que torna isto possível.

Outro programa de análise matemática poderoso, que é usado amplamente nas indústrias de processos químicos, é o Maple V, da Waterloo Mapple Software. Agora disponível na edição Release 4 Power, o programa trata numerosas funções matemáticas avançadas. A tecnologia da empresa é também utilizada como código componente no Mathcad, Matlab e outros programas. A saída do programa pode ser gerada como códico em Fortran ou C, e pode ser traduzido para impressão com o sistema de publicação matemático LaTex. Uma versão em menor escala, que fornece alguma funcionalidade e facilidade de uso, é o Theorist, versão 2.0, disponível para computadores Windows e Macintosh.

2.5. PROBLEMAS DE ENGENHARIA DE REAÇÕES QUÍMICAS

A seguir são descritos alguns problemas típicos de Engenharia de Reações Químicas, os quais são utilizados neste trabalho para o desenvolvimento de simuladores Java via Internet.

2.5.1. Reator CSTR com Camisa de Resfriamento

Uma reação de 1ª ordem é realizada em um reator CSTR de 50 litros (Cancelier et al., 1996). O fluxo volumétrico de alimentação é de 5 l/min e a concentração do reagente A é de 2 kg/l. A temperatura do fluxo de alimentação é de 50°C. A constante pré-exponencial cinética da reação é de 0,38 min^{-1} e a energia de ativação é de 2500 J/mol. A reação é exotérmica com ΔH = -20 kcal/mol. O reator é resfriado por uma camisa de 5 litros, sendo que a água refrigerante entra na camisa com um fluxo volumétrico de 1 l/min e com temperatura de 10°C. A modelagem matemática do problema conduz a um sistema de 3 equações diferenciais ordinárias, representando os balanços de massa, de energia no reator e de energia na camisa, respectivamente:

$$V.\frac{dC_A}{dt} = F.C_{A_0} - F.C_A - V.k.C_A \qquad \text{(Eq. 2.1)}$$

$$V.\rho.\overline{C_p}.\frac{dT}{dt} = \rho.F.\overline{C_p}.T_0 - \rho.F.\overline{C_p}.T + \Delta H.V.k.C_A - U.A.(T - T_c) \qquad \text{(Eq. 2.2)}$$

$$V_c.\rho_c.\overline{C_{p_c}}.\frac{dT_c}{dt} = \rho_c.F_c.\overline{C_{p_c}}.T_{c0} - \rho_c.F_c.\overline{C_{p_c}}.T_c + U.A.(T - T_c) \qquad \text{(Eq. 2.3)}$$

2.5.2. Decaimento Catalítico em um Reator CSTR

O craqueamento em fase gasosa de uma reação de gasóleo leve (Fogler, 1992)

Gasóleo (g) → Produtos de craqueamento (g) + Coque (s)

$$A \rightarrow B$$

é realizado em um reator CSTR fluidizado. A taxa de decaimento catalítico é de primeira-ordem em relação à atividade, e ordem *q* em relação à concentração do reagente. Assume-se que o leito pode ser modelado como um tanque contínuo bem-misturado. A modelagem do problema conduz a um sistema de equações diferenciais acopladas descrevendo reação e desativação simultâneas:

$$V.\frac{dC_A}{dt} = V_0.C_{A_0} - V.C_A - k.q.C_A.V \qquad \text{(Eq. 2.4)}$$

$$\frac{da}{dt} = -k_d.a.C_A^{\ q} \qquad \text{(Eq. 2.5)}$$

2.5.3. Reator em Batelada Adiabático

Propilenoglicol é produzido pela hidrólise de óxido de propileno (Fogler, 1992). A reação se realiza prontamente em temperatura ambiente quando catalizada por ácido sulfúrico. O reator é alimentando com 2500 lb/h (43.03 lb mol/h) de óxido de propileno (PO). A corrente de alimentação consiste de (1) uma mistura equivolumétrica de óxido de propileno (46,62 ft^3/h) e metanol (46,62 ft^3/h), e (2) água contendo 0,1% em peso de H_2SO_4. A vazão volumétrico de água é de 233,1 ft^3/h, que é 2,5 vezes a vazão de metanol-PO. A alimentação molar correspondente de metanol e água é de 71,87 e 802,8 lb mol/h, respectivamente. Há um aumento imediato na temperatura de 17°F com a mistura das duas correntes de alimentação causado pelo calor de mistura. Considerando a instalação de um novo CSTR, revestido de vidro, 175 galões, deve-se fazer uma rápida verificação da dinâmica da reação. Tem-se disponível um reator de mistura em batelada de 10 galões isolado e instrumentalizado. O reator é carregado com 1 galão de óxido de propileno, 1 galão de metanol e 5 galões de água contendo 0,1% em peso de H_2SO_4. A temperatura inicial de todos esses materiais é 58°F. A modelagem matemática do problema conduz a uma equação diferencial ordinária que deve ser resolvida simultaneamente com o balanço de energia:

$$\frac{dX}{dt} = k.(1 - X) \qquad \text{(Eq. 2.6)}$$

$$T = 535 + 90{,}45.X \qquad \text{(Eq. 2.7)}$$

2.5.4. Oxidação de SO_2

A alimentação para um conversor de SO_2 é de 7900 lb mol/h, e consiste de 11% de SO_2, 10% de O_2, e 79% de inertes (Fogler, 1992). O conversor é composto por 4631 tubos empacotados com catalisador, cada qual com 20 pés de comprimento. Os tubos têm 3 polegadas de diâmetro externo e 2,782 polegadas de diâmetro interno. Os tubos serão resfriados por um líquido em ebulição a 805°F, tal que a temperatura do refrigerante é constante acima deste valor. A pressão de entrada é de 2 atm. A modelagem matemática do problema conduz a três equações diferenciais acopladas que devem ser resolvidas simultaneamente:

$$\frac{dX}{dW} = \frac{-r'_A}{F_{A_0}} = 5{,}329.k.\sqrt{\frac{1-X}{X}}.\left[\frac{0{,}2-0{,}11.X}{1-0{,}055.X}.\frac{P}{P_0} - \left(\frac{X}{(1-X).K_p}\right)^2\right] \qquad \text{(Eq. 2.8)}$$

$$\frac{dT}{dW} = \frac{5{,}11.(T_a - T) + (-r'_A).[-\Delta H_R(T)]}{0{,}188.(\sum \Theta_i.C_{p_i} + X.\Delta C_p)} \qquad \text{(Eq. 2.9)}$$

$$\frac{dP}{dW} = -\frac{1{,}12.10^{-8}.(1-0{,}055.X).T}{P}.(5500.\mu + 2288) \qquad \text{(Eq. 2.10)}$$

2.5.5. Reator de Leito Fixo com Dispersão Axial

O balanço de massa do reagente de uma reação simples, irreversível, de ordem m e conduzida isotermicamente em um reator de leito fixo, considerando apenas a difusão axial é descrito pela equação de projeto (Finlayson, 1972):

$$\frac{dy(x)}{dx} = \frac{1}{Pe}.\frac{d^2y(x)}{dx^2} - Da.[y(x)]^m \qquad \text{(Eq. 2.11)}$$

onde: $0 < x < 1$, variável espacial (adimensional)

$y(x) = c(x)/c_0$

$$Pe = \frac{v_z.L}{D_{z,ef}} \quad \text{(n° de Peclet)} \qquad \text{(Eq. 2.12)}$$

$$Da = \frac{k.L.c_0^{\,m-1}}{v_z} \quad \text{(n° de Damköhler)} \qquad \text{(Eq. 2.13)}$$

m = ordem da reação

Esta equação está sujeita às seguintes condições de contorno:

$$-\frac{1}{Pe}.\left.\frac{dy(x)}{dx}\right|_{x=0} = 1 - y|_{x=0} \qquad \text{(Eq. 2.14)}$$

$$\frac{1}{Pe}\cdot\left.\frac{dy(x)}{dx}\right|_{x=1}=0 \qquad \text{(Eq. 2.15)}$$

Após a aplicação do método da colocação ortogonal, que consiste na aproximação do perfil de concentração y(x) por um polinômio em x de grau N+1, obtém-se a equação de resíduo nulo nos pontos de colocação:

$$\sum_{0}^{N+1} C_{ij}.y_j - Da.[y_i]^m = 0 \qquad \text{(Eq. 2.16)}$$

para i=1, ..., N

Os valores y_0 e y_{N+1} são calculados pela aplicação da aproximação nas condições de contorno do problema. Assim:

$$\sum_{0}^{N+1}\left[-\frac{1}{Pe}.A_{0,j}.y_j\right]-1+y_0=0 \qquad \text{(Eq. 2.17)}$$

$$\sum_{0}^{N+1}\left[\frac{1}{Pe}.A_{N+1,j}.y_j\right]=0 \qquad \text{(Eq. 2.18)}$$

Como resultado da aplicação do método da colocação ortogonal, portanto, obtém-se um sistema de N+2 equações algébricas não-lineares.

2.6. PROCESSOS ABORDADOS

A seguir são descritos dois processos químicos, cujas unidades experimentais existentes no Departamento de Engenharia Química são utilizados neste trabalho para a implementação de um sistema cliente/servidor de aquisição e controle à distância.

2.6.1. Microgravimetria

Os sistemas de reação microgravimétricos ou termogravimétricos apresentam um enorme potencial de utilização em diversas áreas da catálise heterogênea e podem ser utilizados com sucesso na caracterização de superfícies ácidas sujeitas à desativação por bases nitrogenadas. Quando devidamente operados, e combinados com outros métodos de análise, esses sistemas mostram-se

especialmente interessantes na determinação da heterogeneidade das superfícies, como por exemplo, a distribuição da força ácida de diferentes sítios ativos (Vedova, 1996). A variação de peso de um catalisador com a variação das condições experimentais pode ser usada para uma variedade de estudos. Um sistema microgravimétrico, equipado com uma microbalança, pode ser usado com apenas poucos miligramas de catalisador e pode detectar variações de peso da ordem de até 0,1 micrograma. As medidas podem ser feitas em temperaturas constantes ou programadas, em pressões constantes ou variáveis, e sob condições estáticas ou de fluxo contínuo.

Este método é usado para estudos de adsorção-desorção, particularmente em estudos cinéticos de formação de coque, desidratação, adsorção de venenos, adsorção de complexos, regeneração de catalisadores, etc., como função das condições de reação. É também útil em estudos de caracterização catalítica, isto é, na oxidação de catalisadores, na qual o peso ganho ou perdido pode revelar o estado de oxidação do material, bem como sua estabilidade com relação ao meio reacional.

O Departamento de Engenharia Química da Universidade Federal de Santa Catarina dispõe de um sistema microgravimétrico para dar suporte à modelagem de superfícies heterogêneas e serve à caracterização de aluminas preparadas no Laboratório de Cinética, Catálise e Reatores Químicos - LABORE - e catalisadores zeolíticos comerciais. O sistema de reação microgravimétrico LABORE foi doado pela American Oil Company (Amoco, Illinois-USA). As Figuras 2.12 a 2.14 mostram os principais equipamentos e dispositivos auxiliares do sistema microgravimétrico LABORE.

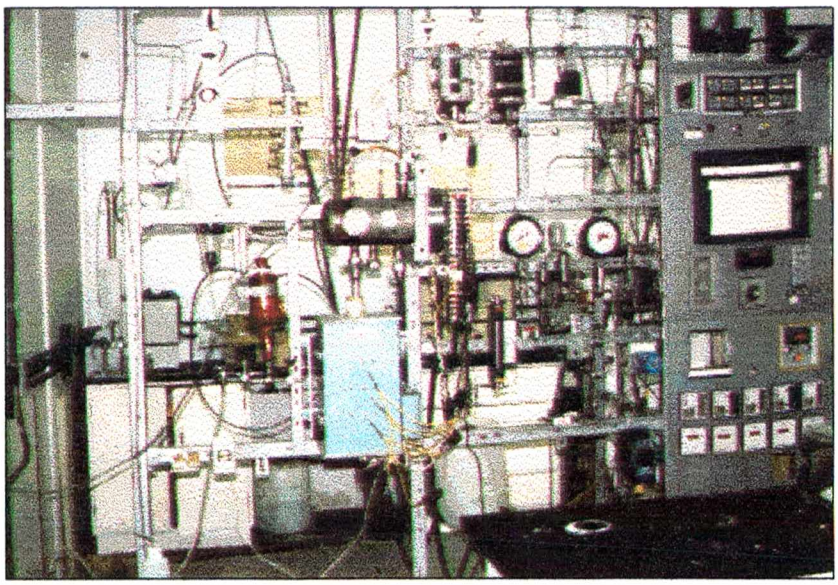


Figura 2.12 – Visão geral do sistema microgravimétrico LABORE/UFSC

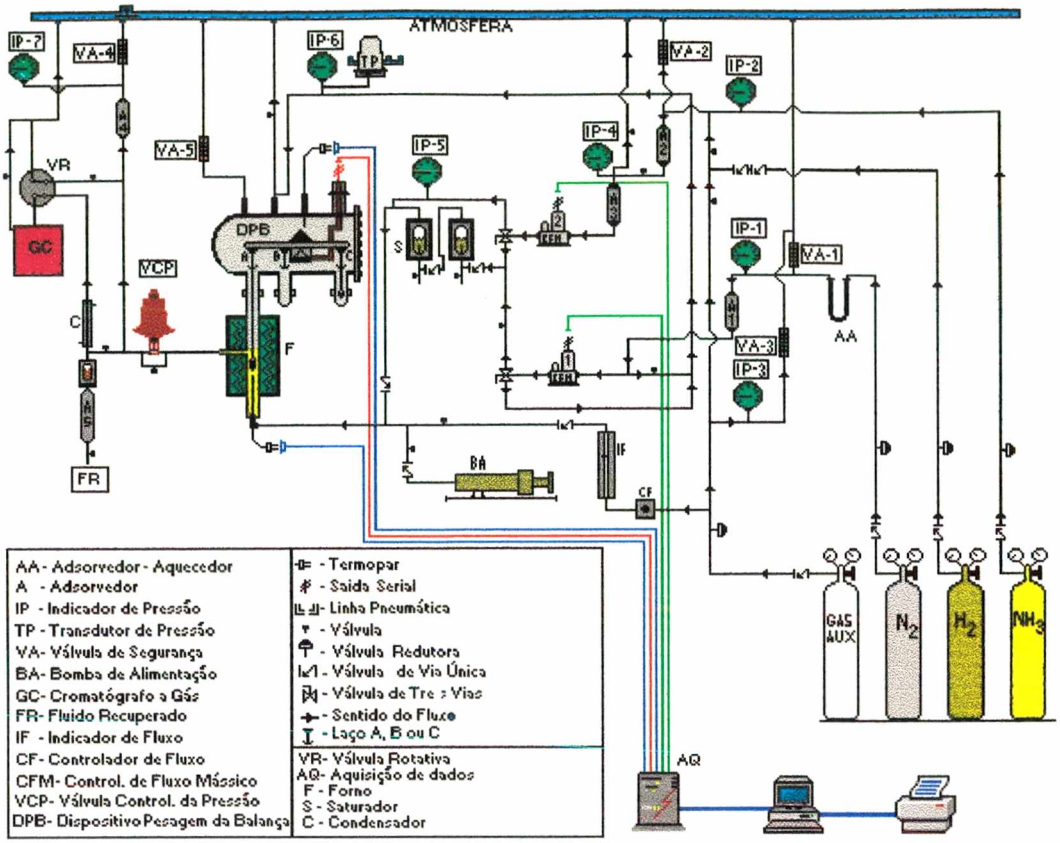
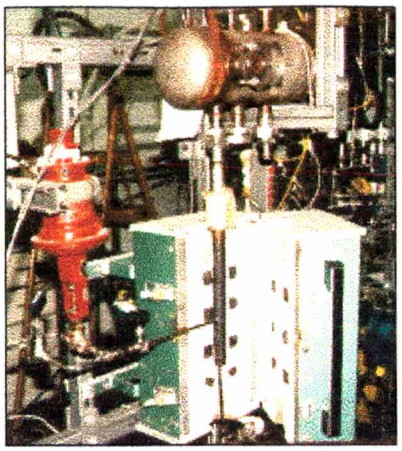


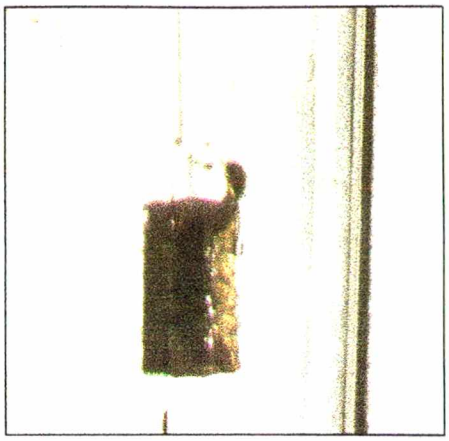
Figura 2.13 – Fluxograma do sistema microgravimétrico LABORE/UFSC

O Sistema Microgravimétrico LABORE consiste de um dispositivo de pesagem da balança (DPB), modelo Cahn 2000, com as seguintes características: capacidade de até 3,5 g, máxima variação de peso de 100 mg e sensibilidade máxima de 0,1 μg. Através de um hack metálico tem-se o suporte da câmara da microbalança, dos equipamentos auxiliares de medição e controle e dos tubos em aço inox de 1/8" e 1/6" das linhas de fluxo de gás. Para aquisição de dados utiliza-se o sistema de aquisição de dados MQLM-01 e a interface MQI12/8PCE da Microquímica Indústria e Comércio Ltda (Vedova, 1996).

O dispositivo é alojado em uma câmara capaz de suportar altas pressões, confeccionado em aço inoxidável. O dispositivo de pesagem opera segundo um mecanismo de compensação eletromagnética, altamente sensível a pequenos torques produzidos no braço de pesagem da balança. O tubo do "reator", também de aço inox, com 45 cm de comprimento e 2,5 cm de diâmetro, conecta-se à câmara da balança, e é envolto por um forno tubular Thermcraft, modelo 114-9-3ZV com três zonas de aquecimento, podendo chegar a 1000°C. Uma haste de vidro de 0,3 mm de diâmetro é utilizada para suspender uma "cestinha" de ouro, local onde é depositada a amostra.



(a) balança e forno elétrico



(b) cestinha para alojamento da amostra

Figura 2.14 - Sistema microgravimétrico LABORE/UFSC

As linhas de fluxo gasoso são de aço inox; sete indicadores de pressão, quatro adsorvedores, dois saturadores, resistências, termopares, dois controladores de fluxo mássico, um indicador de fluxo volumétrico, válvulas de segurança, válvulas redutoras etc., estão devidamente distribuídos ao longo da linha. Sobre a câmara da balança situa-se a entrada do gás inerte (nitrogênio), responsável pela manutenção de uma sobrepressão que visa proteger o dispositivo de pesagem. Uma válvula de controle, acionada pneumaticamente, mantém a pressão do sistema constante.

O sistema de aquisição de dados consta de um painel de controle do dispositivo de pesagem da microbalança Cahn 2000, um sistema de aquisição MQLM-01 (Figura 2.15), e um microcomputador 486 DX2.

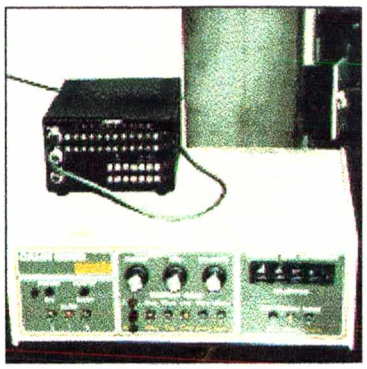


Figura 2.13 - Unidade MQLM-01 e painel de controle da microbalança.

Para ser alcançada a sensibilidade máxima prevista, as condições de operação precisam ser as mais ideais possíveis, evitando-se toda sorte de perturbação que possa se manifestar como ruído e alterar o peso da amostra, tais como vibrações, variações de vazão, pressão, etc.

2.6.2. Ultrafiltração Tangencial

O processo de ultrafiltração vem sendo utilizado com sucesso pelas indústrias do setor alimentício, em particular pelas indústrias de sucos de frutas e de produtos lácteos, como um processo alternativo na clarificação e concentração destes produtos, respectivamente. A crescente preocupação com o problema energético, a busca de produtos alimentícios de melhor qualidade e a valorização de seus subprodutos, vêm privilegiando o surgimento de processos alternativos de clarificação, fracionamento e concentração não convencionais. Dentre estes processos destacam-se os de separação por membranas representados, principalmente, pela ultrafiltração. O mercado mundial movimentou em 1990 cerca de 5 bilhões de dólares em sistemas de separação por membranas. A nível industrial o aumento na utilização vem crescendo entre 12 e 15% ao ano (Porto et al., 1994).

Através de uma membrana de ultrafiltração é possível a retenção de macromoléculas, colóides e materiais em suspensão e a permeação de água, açúcares, sais minerais e outras pequenas moléculas. O sucesso das pesquisas na área de polímeros, a partir da década de 70, contribuiu decisivamente para o surgimento de membranas orgânicas poliméricas com seletividade, resistência mecânica e taxa de permeação compatíveis com a realidade industrial. Há poucos anos surgiram as membranas cerâmicas, confeccionadas a partir de alumina ou zircônia, que pela sua própria natureza são mais resistentes mecanicamente, suportam temperaturas elevadas, além de serem quimicamente inertes, favorecendo o uso em altas pressões. Podem ser esterilizadas à vapor, e facilitam a limpeza com soluções ácidas e alcalinas, quimicamente agressivas.

As indústrias lácteas foram as primeiras a utilizar com sucesso a ultrafiltração no fracionamento e concentração do leite e do soro de queijos. Como o processo não envolve temperaturas elevadas e nem mudança do estado físico da água, como nos processos convencionais de concentração, são mantidas as propriedades nutricionais e funcionais de seus componentes (Juliano e Petrus, 1987).

Mais recentemente, a ultrafiltração e a osmose inversa vêm despertando o interesse das indústrias de sucos de frutas como importantes alternativas na clarificação e pré-concentração do produto. O produto clarificado por ultrafiltração (suco) é livre de macromoléculas como as enzimas, pectina e amido, apresentando qualidade superior, não se turvando ou escurecendo durante a concentração e/ou estocagem. Além disso, não há perda de produtos voláteis e dos responsáveis pelo sabor e aroma, já que o processo pode ser conduzido à baixa temperatura e estes produtos são permeáveis e, portanto, incorporados no suco clarificado. Durante a ultrafiltração do suco, ocorre, ainda, uma

redução de microorganismos no permeado, de até 9 ciclos logarítmicos, assegurando a estabilidade biológica do produto.

Embora algumas indústrias utilizem a ultrafiltração, o seu uso e a sua difusão estão seriamente prejudicados devido ao problema relacionado com o rápido entupimento dos poros das membranas com queda lenta, mas gradual, da velocidade de permeação. Mesmo utilizando-se o fluxo paralelo à superfície filtrante, ocorrem dois fenômenos distintos e que vêm sendo estudados recentemente: a polarização por concentração e a colmatagem. O primeiro é caracterizado por uma maior concentração de macromoléculas junto à superfície filtrante e não pode ser evitado, mas sim controlado. O segundo fenômeno é caracterizado pelo bloqueio parcial ou total dos poros das membranas pelas partículas de tamanho igual ou ligeiramente inferior. Alguns dispositivos ou artifícios vêm sendo propostos, como agentes descolmatantes capazes de manter uma alta velocidade de permeação que, juntamente com a seletividade, são os parâmetros mais importantes para o desempenho do processo.

Os modos de operação que prometem ser mais viáveis no processamento do suco de frutas são o fluxo-inverso e a pulsação. O fluxo inverso do permeado faz com que a maioria das partículas potencialmente colmatantes e aderidas à superfície filtrante sejam arrastadas, mantendo-se alta a velocidade de permeação. A reversão do fluxo, normalmente é aplicada a intervalos regulares de tempo e com duração de poucos segundos. A pulsação consiste na queda brusca de pressão no cartucho da ultrafiltração e a sua recuperação instantânea após alguns segundos. Com esta variação brusca de pressão é criada uma onda de choque que, quando intermitente, pode ter efeito descolmatante.

No Brasil, a tecnologia de separação por membranas começa a difundir-se. A ultrafiltração vem sendo apresentada como um processo bastante promissor junto às indústrias de produtos lácteos e de sucos. Nestas últimas, em substituição à filtração convencional, que é muito menos eficiente e de custo mais elevado, já que requer coadjuvantes de filtração não recuperáveis, e maior mão-de-obra. O Estado de Santa Catarina, pioneiramente, vem utilizando a ultrafiltração tangencial na clarificação e pré-estabilização biológica do suco de maçã, antes da concentração, cujo produto vem sendo exportado para os Estados Unidos. A clarificação por ultrafiltração foi uma exigência dos importadores, já que se obtém um produto de altíssima qualidade quando comparado ao obtido pelos processos convencionais de filtração. Acredita-se que os estudos de dispositivos que possam melhorar a taxa de permeação, tornando o processo ainda mais econômico e reduzindo o tempo necessário à limpeza e recuperação das membranas irá certamente contribuir para a consolidação deste processo e a sua difusão junto às indústrias.

Capítulo 3

MATERIAIS E MÉTODOS

3.1 FERRAMENTA DE DESENVOLVIMENTO

Com o objetivo de avaliar a utilização da linguagem Java para o ensino e para a pesquisa em Engenharia Química foram desenvolvidos vários aplicativos. Após avaliações de várias ferramentas de desenvolvimento comerciais, optou-se pelo pacote Symantec© Visual Café Pro versão 1.0 (1996). O pacote Visual Café Pro possui suporte para a Interface de Programação Java API 1.0 e é o primeiro ambiente de Desenvolvimento de Aplicação Rápida (RAD) projetado exclusivamente para a criação de applets e aplicativos Java.

O Visual Café oferece um conjunto completo de ferramentas integradas de desenvolvimento Java, incluindo as seguintes características: desenvolvimento visual, rápida conexão de componentes, desenvolvimento em duplo sentido com completa correspondência entre as ferramentas visuais e o código Java, criação eficiente com rápidos compiladores e um depurador avançado que inclui avaliação de expressões. Entre as ferramentas integradas de desenvolvimento presentes no aplicativo pode-se destacar o Gerenciador de Projeto, o Projetista de Forma, a Biblioteca de Componentes, o Editor de Código, o Navegador de Classes, o Assistente de Interação e um Depurador Gráfico Integrado acoplado com um Compilador Java e ao Visualizador de Applet (Symantec, 1996).

O Projetista de Forma possibilita a visualização e projeto do applet com ferramentas gráficas. A criação da interface com o usuário é realizada arrastando os componentes desejados da barra de componentes para o applet. As propriedades do componente são definidas em uma janela separada de Lista de Propriedades, e as interações entre componentes são adicionadas com o Assistente de Interação. O código Java é automaticamente criado durante o processo de projeto. Os testes do applet Java são realizados no próprio ambiente de desenvolvimento. Os arquivos HTML necessários para a execução de applets através de navegadores podem ser criados e visualizados diretamente no ambiente de desenvolvimento, ou podem ser criados em outro ambiente e adicionados ao projeto.

Devido à sua Biblioteca de Componentes, é possível desenvolver muitos applets sem haver necessidade de criação ou edição manual de código Java. Para programar outras características não

fornecidas pelo ambiente, pode-se manualmente adicionar o código Java necessário com o Editor de Código. Pode-se adicionar tratamento de eventos mais complexos que não podem ser criados com o Assistente de Interação, adicionar processamento de dados específico e escrever componentes próprios e adicioná-los à Biblioteca de Componentes do Visual Café. A Figura 3.1 monstra a ferramenta de desenvolvimento Visual Café Pro 1.0 em execução.

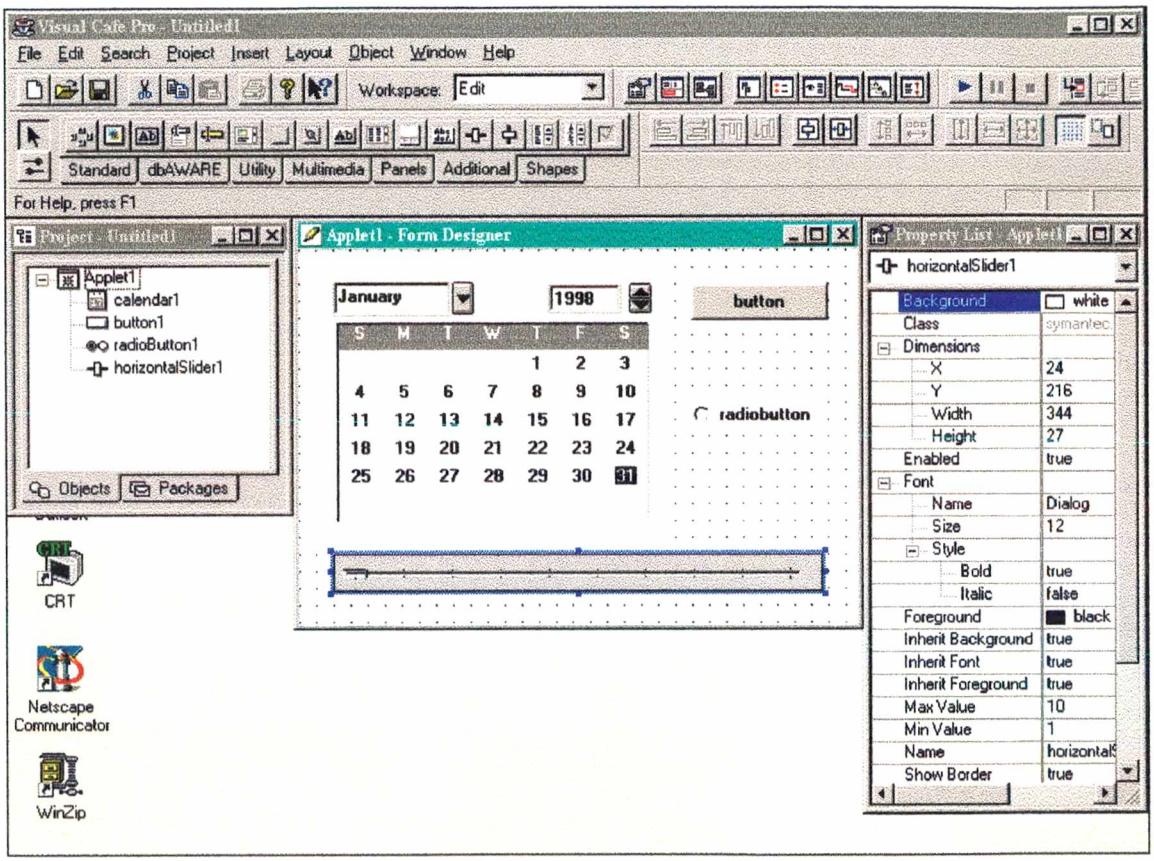


Figura 3.1 – Ferramenta de desenvolvimento Symantec© Visual Café Pro 1.0

3.2 APLICATIVOS DE ENSINO

O estudo realizado no campo de ensino para Engenharia Química inclui o desenvolvimento de um pacote matemático Java para resolução de sistemas de equações algébricas e diferenciais. O estudo inclui ainda a aplicação do pacote desenvolvido, denominado labore.math, no desenvolvimento de applets Java implementando simuladores matemáticos para a resolução numérica dos problemas específicos de Engenharia Química descritos no Capítulo 2 – Revisão Bibliográfica. Por terem sido desenvolvidos em linguagem Java, esses simuladores podem ser acessados por usuários localizados em

qualquer parte do mundo, bastando que os mesmos possuam uma conexão com a rede Internet e que estejam utilizando um navegador compatível com a linguagem Java.

3.2.1. Pacote Matemático labore.math

Com o objetivo de modularizar as rotinas para resolução dos sistemas de equações algébricas e/ou diferenciais resultantes dos modelos matemáticos obtidos dos problemas em estudo, foi desenvolvido um pacote de classes matemáticas denominado labore.math. Com a utilização deste pacote pode-se facilmente desenvolver aplicativos ou applets em Java para a resolução de sistemas de equações algébricas lineares e não-lineares e equações diferenciais ordinárias.

O pacote matemático labore.math é constituído de quatro classes distintas, responsáveis cada qual pela resolução de um tipo específico de sistema de equações, usando um método matemático apropriado.

A Tabela 3.1 lista as classes que compõem o pacote matemático labore.math, bem como o método matemático implementado e o tipo de sistema de equação ao qual se destinam.

Tabela 3.1 – Classes que compõem o pacote matemático labore.math

Classe	Método Matemático	Sistema de Equações
`LUDecomposition`	Decomposição LU	Algébrico linear
`NRaphson`	Newton-Raphson	Algébrico não-linear
`RungeKutta4`	Runge-Kutta 4ª ordem	Diferencial ordinária (problema de valor inicial)
`OrthogonalCollocation`	Colocação Ortogonal	Diferencial ordinária (problema de valor de contorno)

Os três primeiros métodos descritos na Tabela 3.1 foram adaptados como classes Java a partir de algoritmos codificados em C obtidos da literatura (Press, 1988). O método da colocação ortogonal foi modelado com o embasamento teórico obtido do trabalho de Villadsen e Michelsen (1978).

O pacote matemático labore.math estabelece uma estrutura hierárquica entre suas classes, visto que vários dos métodos matemáticos implementados são dependentes uns dos outros. A Figura 3.2 mostra a estrutura hierárquica das classes no pacote matemático labore.math.

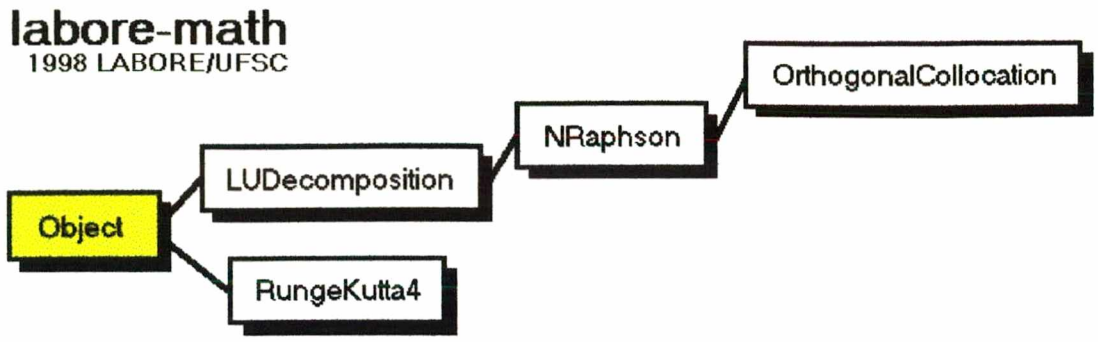


Figura 3.2 – Estrutura hierárquica do pacote matemático labore.math

O método da colocação ortogonal para sistemas de equações diferenciais ordinárias, implementado na classe `OrthogonalCollocation`, por exemplo, reduz o sistema de equações diferencias ordinárias a um sistema de equações algébricas não-lineares, o qual é resolvido pelo método de Newton-Raphson, implementado na classe `NRaphson`.

Com a utilização da característica de herança existente em todas as linguagens de programação orientadas a objetos, ao fazermos a classe `NRaphson` uma superclasse da classe `OrthogonalCollocation` obtemos na classe `OrthogonalCollocation` toda a funcionalidade de resolução de sistemas de equações algébricas não-lineares existente na classe `NRaphson`. Com a definição dessa hierarquia de classes, todas as variáveis e métodos públicos ou protegidos da classe `NRaphson` são diretamente acessíveis pela classe `OrthogonalCollocation`, como se houvessem sido declaradas nessa última. A mesma analogia feita para a classe `OrthogonalCollocation` pode ser estendida para todo o pacote labore.math.

3.2.1.1 Aplicação do Pacote labore.math

A aplicação do pacote matemático para o desenvolvimento de simuladores ou "solvers" matemáticos em linguagem Java é simples e rápida. Todas as classes do pacote labore.math são abstratas, sendo que para sua utilização deve-se criar uma classe derivada da classe matemática correspondente, dependendo do método matemático de resolução desejado. Esta classe conterá a modelagem matemática do problema em estudo, incluindo todas as equações algébricas e/ou diferenciais pertinentes, equações auxiliares e valores de referência para os parâmetros do problema.

Após a criação de uma classe derivada de uma das classes pertencentes ao pacote matemático, deve-se proceder com a chamada a alguns métodos para a especificação do sistema de equações e resolução do mesmo. Todas as classes no pacote matemático seguem basicamente a mesma metodologia e nomenclatura dos métodos responsáveis pela implementação final da classe para resolução de sistemas de equações. São quatro os métodos principais necessários para resolução de um sistema de equações:

– método `defineInitialSystem` – método para definição de parâmetros iniciais relacionados com o sistema de equações em estudo, como número de equações, número de iterações, tolerância nas avaliações, etc.;

– método `system` – método abstrato para especificação do sistema de equações proveniente da modelagem matemática. Dependendo da classe utilizada seu nome variará. Para a classe LUDecompsition os métodos de especificação do sistema são `addCoeff`, `addLine`, `addColumn`, `addMatrixCoeff` e `addMatrix`. Para as classes `NRaphson` e `OrthogonalCollocation` o método de especificação do sistema é `system`. Por último, para a classe `RungeKutta4` o método de especificação do sistema é `derivs`;

– método `calcSystem` – método para resolução do sistema de equações em estudo;

– método `getVariables` – método para obtenção do resultado do sistema de equações, obtido através da aplicação do método matemático.

A descrição do procedimento mínimo para implementação das classes do pacote labore.math na resolução de sistemas de equações pode ser encontrada no Apêndice B – Implementação do pacote matemático labore.math. A relação e descrição dos métodos públicos disponíveis em cada classe em específico pode ser encontrada no Apêndice C – Documentação do pacote matemático labore.math, ou no site Web do pacote labore.math (http://www.labore.ufsc.br/java/labore.math).

3.2.2 Simuladores Matemáticos via Internet

Com o objetivo de avaliar a utilização do pacote matemático labore.math e da linguagem Java no desenvolvimento de aplicativos de ensino para Engenharia Química, foram desenvolvidos alguns simuladores matemáticos para resolução de problemas típicos de projeto e avaliação de reatores químicos. A descrição e a modelagem matemática dos problemas em estudo foram descritas anteriormente no Capítulo 2 – Revisão Bibliográfica.

Uma diferença marcante nos simuladores desenvolvidos em relação aos tradicionais é a possibilidade de se poder acessá-los remotamente. Utilizando um navegador compatível com a linguagem Java é possível a qualquer usuário acessar o simulador através da Internet.

Os simuladores desenvolvidos são acessados pelo usuários através de uma página HTML, na qual o applet Java está embutido. Ao requisitar o endereço URL da página HTML contendo o applet, o navegador inicia a transmissão tanto da página quanto das classes que compõem o applet. Após a transmissão de todas as classes necessárias para a execução do applet, o interpretador presente no navegador inicia a execução do simulador.

Neste trabalho são demonstrados cinco simuladores desenvolvidos para resolução dos problemas em estudo:

- Simulador de um Reator CSTR Encamisado
- Simulador de um Reator Adiabático em Batelada
- Simulador de um Conversor de SO_2
- Simulador de um Reator CSTR Fluidizado sujeito a Decaimento Catalítico
- Simulador de um Reator de Leito Fixo com Dispersão Axial, utilizando um Modelo Estacionário Isotérmico

Utilizou-se no desenvolvimento de cada simulador a mesma estrutura de classes, a qual é composta por duas classes principais:

– Classe contendo a definição da interface gráfica com o usuário: classe derivada de `java.applet.Applet` e responsável pela definição da interface com o usuário. Esta classe permite ao usuário a edição de parâmetros do problema, o controle da simulação e a visualização dos resultados da simulação numérica na forma de gráficos ou de resultados tabulares.

– Classe contendo a definição do modelo matemático: classe derivada de alguma das classes matemáticas do pacote labore.math e responsável pela formulação do problema. Nesta classe são definidas todas as equações algébricas e/ou diferenciais pertinentes, equações auxiliares e valores de referência para os parâmetros do problema. Sendo uma classe derivada de uma das classes do pacote

labore.math, esta classe herda toda a funcionalidade das variáveis e métodos públicos e protegidos de sua superclasse, tendo acesso ao algoritmo e critério a ser utilizado na resolução do problema.

Como exemplo, a Figura 3.3 monstra o fluxo de informações típico existente entre as classes envolvidas no Simulador de um Reator CSTR Fluidizado sujeito a Decaimento Catalítico.

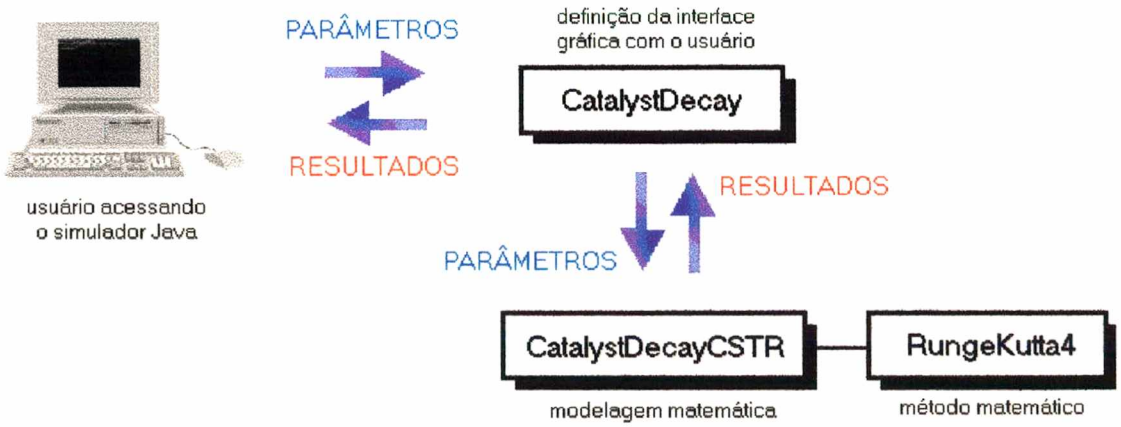


Figura 3.3 – Fluxo típico de informações nos simuladores Java desenvolvidos

3.2.3 Parâmetros de Avaliação Técnica

Os applets simuladores foram carregados da rede Internet e testados para avaliação técnica utilizando-se um computador Pentium 166 MHz rodando Windows 95 e equipado com um modem USRobotics de 33.600 bps.

Para a obtenção dos parâmetros de avaliação foi utilizado o navegador Netscape® Communicator versão 4.03 (1997). A Máquina Virtual Java implementada nesse navegador suporta a versão 1.1.2 da API Java, sendo implementado o interpretador Symantec Java! ByteCode Compiler Version 210.057.

3.2.3.1. Tempo de Transferência do Simulador

Um dos parâmetros mais importantes na avaliação técnica dos simuladores Java desenvolvidos é seu tempo de transferência ("downloading"). Os simuladores necessitam ser transmitidos pela rede Internet do host onde estão armazenados até o host do usuário para que inicie sua execução local. O tempo de transferência dos simuladores envolve a transmissão das classes que compõem o applet e dos demais arquivos associados com a página Web (arquivos HTML, arquivos de imagem, etc.) na qual o applet está inserido. O tempo de transmissão depende de vários fatores como: tamanho dos arquivos das classes, tempo de transmissão de dados entre o host do usuário e o host onde o applet

está armazenado, existência de classes armazenadas na memória "cache" do navegador, entre outros. Em geral apenas poucas classes necessitam ser transmitidas via rede, visto que a grande maioria das classes utilizadas por um applet já são parte integrante do interpretador incorporado em navegadores compatíveis.

Para a avaliação do tempo de transmissão, inicialmente relacionou-se os arquivos de classes que necessitam ser transmitidas via rede, bem como seu tamanho em Kbytes.

Os tempos de transferência foram adquiridos com o auxílio de um cronômetro digital, com precisão de centésimos de segundos, e com a memória cache vazia do navegador. Este procedimento garante que todas as classes necessárias serão transmitidas pela rede. Os valores de tempo de transferência obtidos correspondem ao tempo decorrido entre a solicitação ao navegador da URL correspondente à página HTML na qual o applet está embutido e o início da execução do applet.

3.2.3.2. Tempo de Resolução

Outro parâmetro a ser avaliado tecnicamente nos simuladores desenvolvidos é o tempo despendido na resolução do modelo matemático proposto em cada problema. Este tempo é composto pelo tempo de cálculo do método matemático e pelo tempo de atualização dos resultados na interface gráfica com o usuário. A princípio, os applets simuladores após terem sido carregados no navegador do usuário executam como programas locais; seu desempenho não é afetado pelo fato de ter sido transferido pela rede e independe da localização do host do usuário.

O tempo de resolução é obtido do próprio simulador, que foi projetado para graficamente exibir esta informação ao usuário. Para os problemas com sistemas de equações diferenciais ordinárias utilizando o método de Runge-Kutta, os simuladores exibem o tempo de resolução total. No problema com sistema de equações diferenciais ordinárias utilizando o método da colocação ortogonal, o simulador fornece o tempo de resolução em seus dois componentes: tempo de cálculo e tempo de atualização da interface gráfica.

Juntamente com o tempo de resolução foram registradas a memória total e a memória livre disponível para a Máquina Virtual Java, após a execução do "garbage collection". Utilizou-se o aplicativo Java Console, disponível no navegador Netscape Communicator, para obtenção dos valores de memória total e memória livre e para a execução do garbage collection. Notou-se que sem a utilização do garbage collection, a memória livre era reduzida gradativamente, afetando o tempo de resolução registrado. Os valores do tempo de resolução não apresentavam reprodutibilidade para as mesmas

condições de simulação. Utilizando o garbage collection para manter a memória livre antes do início da simulação, obtém-se valores constantes de tempo de resolução. Desta forma, consegue-se valores de tempo de resolução reprodutíveis e passíveis de comparação e análise. A Figura 3.4 mostra o aplicativo Java Console em execução.

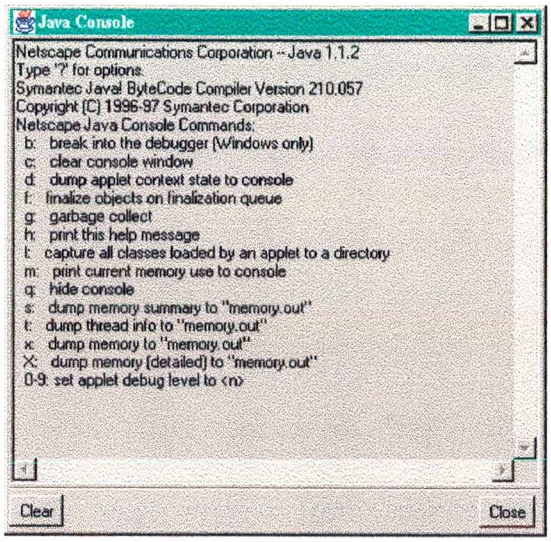


Figura 3.4 – Aplicativo Java Console (Netscape Communicator 4.03)

3.3 APLICATIVOS DE PESQUISA

O desenvolvimento de ferramentas de pesquisa para Engenharia Química incluiu a implantação de um sistema cliente/servidor em linguagem Java para aquisição e controle à distância de equipamentos industriais e de laboratório. O estudo inclui ainda a aplicação do sistema cliente/servidor desenvolvido em duas unidades experimentais existentes no Departamento de Engenharia Química da Universidade Federal de Santa Catatina.

O aplicativo cliente/servidor foi inicialmente aplicado na Unidade de Microgravimetria do Laboratório de Cinética, Catálise e Reatores Químicos – LABORE. Esta unidade experimental é utilizada principalmente para estudos de envenenamento de catalisadores.

A seguir o aplicativo cliente/servidor foi aplicado na Unidade de Ultrafiltração Tangencial do Laboratório de Separação por Membranas – LABSEM. Esta unidade foi adquirida durante a execução deste trabalho e os processos de automatização e calibração da unidade são também parte integrante deste trabalho. A Unidade de Ultrafiltração Tangencial utilizada se destina principalmente a estudos de clarificação de suco de maçã.

3.3.1. Sistema Cliente/Servidor de Aquisição e Controle à Distância

Os sistemas cliente/servidor são em geral concebidos com o intuito de se estabelecer uma conexão com um computador localizado remotamente que, via de regra, contém o aplicativo servidor, a partir de um computador local, com o software cliente, a fim de se obter dados processados ou simplesmente armazenados remotamente.

Através do desenvolvimento de sistemas em linguagem Java utilizando a arquitetura cliente/servidor pode-se implementar um sistema de comunicação entre um aplicativo servidor instalado na unidade na qual deseja-se obter informações sobre determinado equipamento ou processo e um applet cliente acessado pela rede Internet.

O aplicativo servidor realiza a tarefa de automação do equipamento, adquirindo dados e modificando "set-points", com a vantagem de estar apto a aceitar conexões remotas através da rede. Através do applet cliente o usuário pode não apenas visualizar, praticamente em tempo real, a aquisição de dados de determinado processo ou unidade, mas pode também interagir com o mesmo modificando seus parâmetros de aquisição e controle. A Figura 3.5 mostra esquematicamente o funcionamento do sistema cliente/servidor para aquisição e controle à distância.

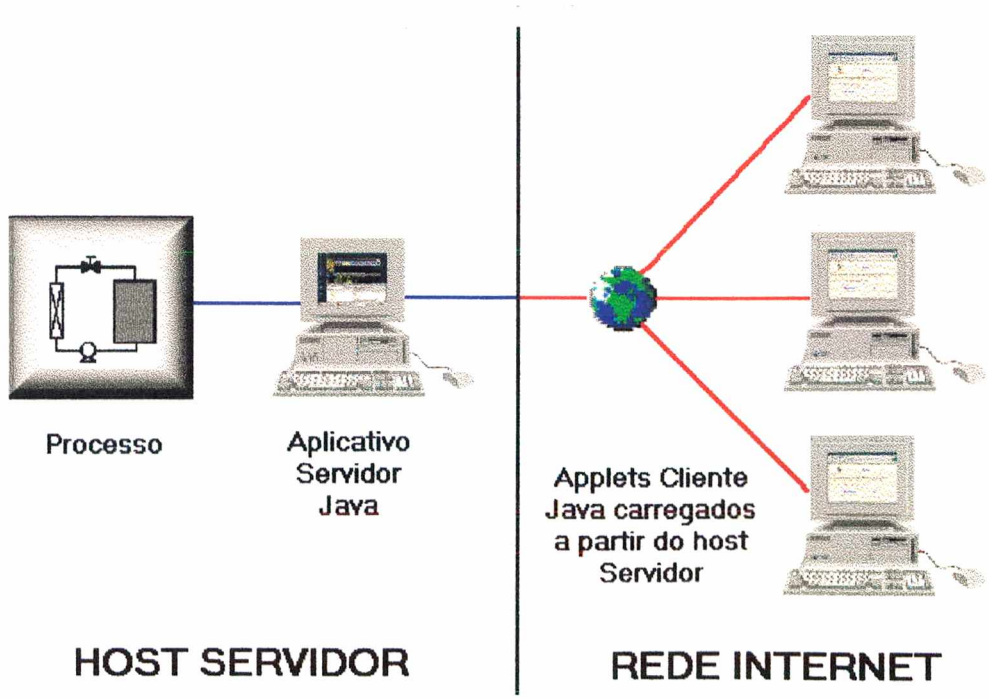


Figura 3.5 – Funcionamento do sistema cliente/servidor para aquisição e controle à distância

Por motivos de segurança implementados pela própria linguagem Java, o aplicativo servidor Java e o applet cliente Java devem ser instalados no mesmo host. Caso o applet cliente Java seja instalado em outro host, e seja utilizado para realizar a conexão com o aplicativo servidor, ocorrerá uma exceção de segurança. Isto impede que alguém mal intencionado possa desenvolver um applet nocivo, que conheça o protocolo de comunicação com o servidor e a porta no qual o mesmo atua, instalá-lo em qualquer servidor Web e conseguir sucesso na conexão com o servidor. Desta forma, apenas o applet cliente instalado no mesmo host do aplicativo servidor possibilitará acesso ao mesmo.

Sistemas semelhantes podem ser utilizados para o controle e/ou acompanhamento de processos críticos, localizados remotamente e/ou em ambientes de grande risco, bem como no ensino à distância, permitindo a realização dos experimentos diretamente pelos alunos localizados distantes dos laboratórios.

O sistema cliente/servidor desenvolvido neste trabalho apresenta uma série de características implementadas a fim de alcançar o objetivo desejado:

Multitarefa: utilizando o suporte multitarefa ("multithread") existente na linguagem Java, tanto o aplicativo servidor quanto o applet cliente possuem várias tarefas independentes executando concorrentemente.

Aquisição e Controle à Distância: através da utilização de métodos nativos implementados em uma Biblioteca de Linkedição Dinâmica (DLL) escrita em linguagem C é possível ter-se acesso a interface de aquisição de dados e realizar a interação com equipamentos e processos. Aliando-se a isso a utilização do suporte à rede oferecido pela linguagem Java, pode-se implementar um servidor para transmitir os dados adquiridos do processo e receber parâmetros enviados pela rede por usuários do sistema.

Níveis de Segurança: através da implementação de senhas de segurança, o sistema oferece a seus usuários dois níveis de acesso: visitante e administrativo. O nível de acesso visitante permite que os clientes conectados no servidor tenham autorização apenas para visualização dos dados adquiridos pelo servidor. O nível de acesso administrativo permite que os clientes possam ser mais do que simples espectadores. Este tipo de acesso permite aos usuários do sistema interagir com o processo, através da alteração de parâmetros como intervalo de aquisição e set-points do processo.

Canal de Comunicação: através de um servidor de comunicação de mensagens é possível estabelecer uma interação direta entre o operador da unidade e os usuários utilizando o sistema pela rede. Isto é de grande utilidade para os objetivos do sistema, pois possibilita que dúvidas possam ser esclarecidas e sugestões registradas, dando o suporte fundamental para a aplicação do sistema tanto como ferramenta de ensino à distância quanto como ferramenta auxiliar no processo produtivo industrial.

Cronograma de Experimentos: através do estabelecimento pelo operador da unidade de um cronograma de experimentos a serem realizados e da disponibilização destas informações a clientes conectados ao sistema, os usuários podem tomar conhecimento de quando o processo estará em funcionamento.

Sistema Multiprotocolo de Transmissão: para a transmissão dos dados relacionados ao controle de conexões remotas e à aquisição e controle do processo o sistema utiliza o protocolo TCP. Para a transmissão dos dados relacionados ao canal de comunicação à exigência quanto à entrega das informações não é crítica e optou-se por utilizar o protocolo UDP.

Duas portas de comunicação no mesmo aplicativo: com o objetivo de facilitar a independência entre os dois tipos de dados transmitidos, o sistema dedica a cada um uma porta de comunicação distinta. Desta forma, cada tipo de informação flui de e para o sistema em portas de comunicação independentes, como se tivéssemos duas canalizações independentes transportando produtos diferentes.

Sistema Multiprotocolo de Comunicação de Instruções: além de haver a separação entre os dois tipos de dados transmitidos quanto às portas e protocolos de transmissão, foram implementados dois protocolos de comunicação de instruções distintos entre cliente e servidor. Para a funcionalidade de controle de conexões remotas e transmissão de dados relacionados com o processo foi desenvolvido o LDAP – Labore Data Acquisition Protocol. Para o controle de conexão e transmissão de dados relacionados com o canal de mensagens foi desenvolvido o protocolo denominado LMCP – Labore Message Communication Protocol. Ambos os protocolos de comunicação serão descritos a seguir.

3.3.1.1. Transferência de Informações entre Cliente e Servidor

Com o objetivo de não sobrecarregar a transmissão dos dados relacionados com o processo em acompanhamento em função das mensagens transmitidas através do canal de comunicação, foram desenvolvidas duas tarefas específicas para cada tipo de comunicação implementada entre cliente e servidor. Cada tarefa é executada independentemente e utiliza protocolos e portas de comunicação distintas.

A comunicação entre cliente e servidor é realizada mediante uma série de instruções específicas que compõem protocolos de comunicação próprios. As instruções são compostas de um comando e seus possíveis argumentos, separados pelo caractere dois pontos ":".

A tarefa responsável pela transmissão e recepção de dados relacionados com o processo é instalada na porta 8190 do servidor e utiliza o protocolo de transmissão TCP. O protocolo TCP permite que haja garantia na entrega de dados entre cliente e servidor. Isto é altamente recomendável em se tratando de dados relacionados com o processo em acompanhamento. Esta tarefa foi implementada utilizando-se um protocolo de comunicação de instruções desenvolvido neste trabalho, LDAP – Labore Data Acquisition Protocol. Este protocolo implementa uma instrução muito importante na avaliação do sistema cliente/servidor Java: o tempo de resposta do servidor. Esse tempo é obtido pelo applet cliente e corresponde ao tempo decorrido entre o envio pelo cliente da instrução DELAY ao servidor e o recebimento pelo cliente da instrução SERVERDELAY, instrução transmitida pelo servidor ao receber a solicitação do cliente. A Tabela 3.2 descreve as instruções que compõem o protocolo de comunicação LDAP no sentido Cliente ⇒ Servidor.

Tabela 3.2 – Instruções do protocolo de comunicação LDAP, quando utilizado no sentido Cliente ⇒ Servidor.

INSTRUÇÃO	FUNÇÃO
CONNECT:<login>:<senha>	Solicitação de conexão no servidor
DISCONNECT	Solicitação de desconexão do servidor
CHRONOGRAM	Solicitação de recebimento do cronograma de experimentos
DELAY	Solicitação de recebimento de resposta do servidor para medida do tempo de resposta
PARAMETERn:<valor>	Envio do parâmetro n do processo
HELP	Solicitação de ajuda

A Tabela 3.3 descreve as instruções que compõem o protocolo de comunicação LDAP no sentido Servidor ⇒ Cliente.

Tabela 3.3 – Instruções do protocolo de comunicação LDAP, quando utilizado no sentido Servidor ⇒ Cliente.

INSTRUÇÃO	FUNÇÃO
CONNECTION ACCEPTED	Confirmação de conexão no servidor
DISCONNECTION ACCEPTED	Confirmação de desconexão do servidor
CONNECTION ABORTED	Informação de desconexão pelo operador
CHRONOGRAN:<n%>:<data>	Envio de um item de cronograma
PARAMETERS ACCEPTED	Confirmação de recebimento de parâmetros
PARAMETERS:<valor1>::<valorN>	Envio de parâmetros adquiridos do processo
SERVERDELAY	Envio de resposta do servidor para medida do tempo de resposta
Mensagem de Erro	Envio de notificação da existência de algum erro na utilização do protocolo LDAP

Por outro lado, a tarefa responsável pela transmissão e recepção de dados relacionados com o canal de conversação é instalada na porta 8180 do servidor e utiliza o protocolo de transmissão UDP. O protocolo UDP fornece um maior desempenho em relação ao protocolo TCP por não haver garantia na entrega dos datagramas. Para a funcionalidade do canal de conversação a não garantia de entrega dos dados é menos importante do que o desempenho. Esta tarefa foi implementada utilizando o outro protocolo de comunicação desenvolvido durante este trabalho, denominado LMCP – Labore Message Communication Protocol.

A Tabela 3.4 descreve as instruções que compõem o protocolo de comunicação LDAP no sentido Cliente ⇒ Servidor. A Tabela 3.5 descreve as instruções que compõem o protocolo de comunicação LDAP no sentido Servidor ⇒ Cliente.

Tabela 3.4 – Instruções do protocolo de comunicação LMCP, quando utilizado no sentido Cliente ⇒ Servidor.

INSTRUÇÃO	FUNÇÃO
CONNECT	Solicitação de conexão no canal
DISCONNECT	Solicitação de desconexão do canal
MSG:<login destino><mensagem>	Envio de mensagem para determinado login de destino

Tabela 3.5 – Instruções do protocolo de comunicação LMCP, quando utilizado no sentido Servidor ⇒ Cliente.

INSTRUÇÃO	FUNÇÃO
CONNECTION OK	Confirmação de conexão no canal
OPERATOR NAME:<login>	Envio do login do operador do canal
USER LOGIN:<login>	Envio de login conectado ao canal
MSG:<login origem><mensagem>	Envio de mensagem de determinado login de origem

3.3.1.2. Applet Cliente Java

O applet cliente Java, assim como os simuladores descritos anteriormente, pode ser acessado através da Internet por qualquer usuário utilizando um navegador compatível e possibilita a conexão com o servidor de experimentação e controle à distância. Após a conexão com o servidor o usuário tem a possibilidade de acompanhar a aquisição de um experimento em andamento, alterar parâmetros de aquisição e controle do processo, informar-se sobre o cronograma de experimentos programados e utilizar o canal de comunicação de mensagens para interagir com outros usuários e com o operador da unidade.

O applet cliente é composto por três classes responsáveis pela implementação de sua funcionalidade, conforme Figura 3.6.

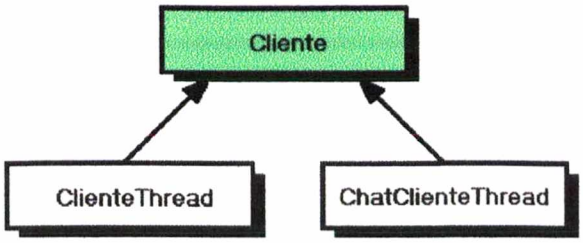


Figura 3.6 – Estrutura das classes do applet cliente Java

Classe `Cliente` - A classe `Cliente`, derivada da classe `java.applet.Applet`, é responsável pela definição da interface gráfica com o usuário e pelo compartilhamento da funcionalidade do cliente. Nesta classe são criadas instâncias das classes `ClienteThread` e `ChatClientThread`, descritas a seguir. A interface gráfica disponibilizada ao usuário possibilita a conexão com o servidor, o envio de parâmetros do processo, o recebimento de dados de aquisição do processo, o acesso ao canal de mensagens e a visualização do tempo de resposta do servidor. A classe `Cliente` possui um temporizador interno que realizar automaticamente a solicitação do tempo de resposta ao servidor a cada dez segundos.

Classe `ClienteThread` - A classe `ClienteThread`, derivada da classe `java.lang.Thread`, implementa a tarefa responsável pela conexão com o servidor, pela recepção de dados sendo adquiridos do processo e pela transmissão de parâmetros de aquisição e controle para o servidor. Ao ser inicializada, a tarefa cria uma instância da classe `java.net.Socket`, através do fornecimento do endereço Internet e porta de comunicação (8190) do servidor. Essa instância é a parte central do cliente Java e é responsável pela criação de um canal de conexão com o servidor. Após criar o socket cliente, a tarefa obtém os fluxos de entrada (instância da classe `java.io.DataInputStream`) e de saída (instância da classe `java.io.PrintStream`) do canal de conexão. A tarefa é executada em um laço infinito e fica aguardando por instruções vindas do servidor. O método `readLine` do fluxo de entrada é utilizado para obtenção de instruções. Após receber alguma instrução, no formato de uma string, a tarefa realiza uma série de testes para determinação da ação apropriada para a instrução recebida. Para envio de instruções ao servidor é utilizado o método `println` do fluxo de saída. Essa classe utiliza o protocolo de transmissão TCP e o protocolo de comunicação de instruções LDAP, descrito anteriormente.

Classe `ChatClienteThread` - A classe `ChatClienteThread`, derivada da classe `java.lang.Thread`, implementa a tarefa responsável pela transmissão de dados relacionados com o canal de mensagens entre cliente e servidor. Ao ser inicializada, esta tarefa implementa uma

instância da classe `java.net.DatagramSocket`, a qual implementa um socket de datagramas. Dois métodos desta instância implementam as funções de recebimento e de envio de datagramas. Os datagramas são transmitidos através da rede Internet como vetores de bytes. A tarefa executa em um laço infinito e utiliza o método `receive` para o recebimento de datagramas contendo instruções vindas do servidor. Como o método `receive` atua apenas como um recipiente para um datagrama ser recebido, deve-se extrair os dados do datagrama recebido com o método `getData` da classe `java.net.DatagramPacket`. Para o envio de dados de datagrama contendo instruções ao servidor, a tarefa utiliza o método `send`. Ao contrário da classe `ClienteThread`, esta tarefa utiliza o protocolo UDP para transmissão das mensagens. O protocolo de comunicação de instruções utilizado por esta tarefa é o LMCP, descrito anteriormente.

3.3.1.3. Aplicativo Servidor Java

O aplicativo servidor Java é um aplicativo executado no "host" onde a unidade objeto de experimentação e/ou controle à distância se localiza. O servidor Java foi desenvolvido visando oferecer a máxima funcionalidade possível para o adequado cumprimento das funções pretendidas. A Figura 3.7 mostra a estrutura de classes desenvolvida para a implementação do aplicativo servidor.

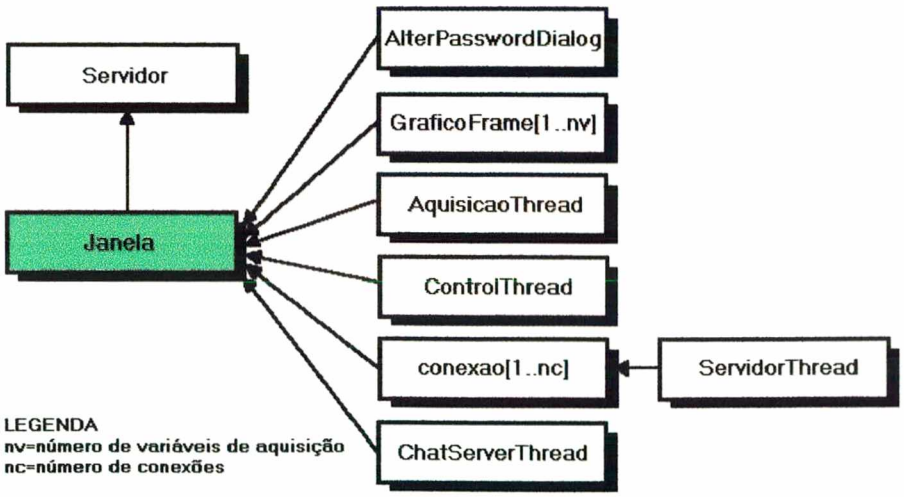


Figura 3.7 – Estrutura das classes do aplicativo servidor Java

Classe `Servidor` – A classe `Servidor` é implementada como um aplicativo Java e possui apenas uma instrução, para a criação de uma instância da classe `Janela`.

Classe `Janela` – A classe `Janela` é implementada como uma classe derivada da classe `java.awt.Frame` e é a responsável pela gerência do aplicativo Servidor. Nesta classe são definidas

e criadas instâncias das demais classe que compõem o aplicativo servidor. Além de ser responsável pela definição da interface gráfica com o usuário, esta classe é responsável pelo coordenação das demais classes e tarefas que compõem o sistema. Esta classe implementa grande parte da funcionalidade do aplicativo servidor, como:

- definição da interface gráfica com o usuário e gerenciamento de eventos gerados na mesma;
- leitura e salvamento de arquivos de configuração (formato ASCII) e salvamento de arquivos com dados de aquisição (formato ASCII ou binário);
- criação, incorporação e atualização dos gráficos com resultados da aquisição (vetor de instâncias da classe `GraficoFrame`). É criada uma instância da classe `GraficoFrame` para cada variável de aquisição do processo;
- criação e inicialização das tarefas de aquisição de dados do processo (instância da classe `AquisicaoThread`), de controle do processo (instância da classe `ControlThread`) e de comunicação de mensagens (instância da classe `ChatServerThread`);
- criação e gerenciamento do vetor de conexões (instâncias da classe `conexao`);
- criação do socket servidor (instância da classe `ServerSocket`). Esta instância implementa um socket para aceite de conexões de clientes vindos da rede Internet. A cada nova solicitação de conexão uma nova instância da classe conexao é criada e adicionada ao vetor de conexões.
- gerenciamento do cronograma de experimentos.

Classe `conexao` – a classe `conexao` é a classe responsável pelo armazenamento das informações relacionadas com cada cliente conectado ao servidor. Nesta classe ficam armazenadas informações como o endereço Internet, a porta de comunicação de dados do processo, a porta de comunicação de mensagens, o nome do usuário, o tipo de acesso (visitante ou administrativo), os fluxos de entrada e saída e o status da conexão com o servidor. Uma nova instância desta classe é criada sempre que uma nova conexão é realizada no servidor. O conjunto de conexões ativas é armazenado em um vetor na classe `Janela`. Cada instância desta classe implementa uma instância da classe `ServidorThread`, tarefa responsável pelo recebimento e envio de dados do processo.

Classe **`AlterPasswordDialog`** – classe derivada da classe `symantec.itools.awt.util.dialog.ModalDialog`, e que implementa uma caixa de diálogo modal utilizada para alteração da senha administrativa.

Classe **`GraficoFrame`** – classe derivada da classe `java.awt.Frame`, e que implementa uma caixa de diálogo responsável pela exibição gráfica do resultado da aquisição de dados do processo. Cada variável de aquisição possui uma instância desta classe para exibição de seus valores. Esta classe cria uma instância da classe `labore.graph.LineGraph`, a qual implementa um gráfico de linhas. A classe `LineGraph` foi a única classe gráfica desenvolvida durante este trabalho, e utiliza o pacote gráfico warhol.charts. Dois vetores são implementados nesta classe com o objetivo de facilitar a obtenção dos valores das ordenadas e abscissas. Os pontos adquiridos do processo são facilmente adicionados ao gráfico através do método `addPoint` definido nesta classe.

Classe **`ServidorThread`** - A classe `ServidorThread` é implementada como uma classe derivada da classe `java.lang.Thread` e é a classe responsável pela tarefa de transmissão de dados relacionados à operação da Unidade. Através desta classe é realizado o envio dos dados adquiridos do processo aos clientes conectados e o recebimento de parâmetros enviados pelo cliente ao servidor. A cada nova instância da classe `conexao` criada e adicionada ao vetor de conexões da classe `Janela` é criada uma nova instância da classe `ServidorThread`. A tarefa é executada como um laço infinito e utiliza o método accept da instância da classe `ServerSocket` criada pela classe `Janela` para ficar aguardando na porta 8190 pela solicitação de conexão de algum cliente. Ao receber uma requisição de conexão, o método `accept` retoma uma instância da classe `Socket`, a qual é utilizada para obtenção do endereço Internet, da porta de comunicação e dos fluxos de entrada e saída utilizados pelo cliente. Após a tarefa definir estes parâmetros para a instância da classe `conexao` à qual pertence e comunicar à classe `Janela` a existência de uma nova conexão, uma nova instância da classe `conexao` é criada e adicionada ao vetor de conexões. Esta instância permanece com seus parâmetros nulos até que um novo cliente faça a solicitação de conexão ao servidor. Assim como a classe `ClienteThread` descrita anteriormente, esta tarefa utiliza o método `readLine` e o método `println` para obtenção e envio de instruções, respectivamente. Após receber alguma instrução, no formato de um string, a tarefa realiza uma série de testes para determinação da ação apropriada para a instrução recebida. Esta tarefa utiliza o protocolo de transmissão TCP e o protocolo de comunicação de instruções LDAP.

Classe **`ChatServerThread`** - A classe `ChatServerThread` é implementada como uma classe derivada da classe `java.lang.Thread` e é a classe responsável pela tarefa de transmissão de dados

relacionados com o canal de mensagens. É a classe responsável ainda pelo controle de conexões no canal de mensagens e pelo gerenciamento de mensagens vindas de e para o aplicativo servidor. Ao ser inicializada, a tarefa cria um objeto da classe `java.net.DatagramSocket` e o coloca aguardando na porta 8180 por alguma instrução vinda do cliente. O procedimento para envio e recebimento de datagramas é realizado da mesma forma como descrito para a classe `ChatClienteThread`. As instruções são transmitidas como datagramas binários (instâncias da classe `java.net.DatagramPacket`) utilizando o protocolo de transmissão UDP. Diferentemente do protocolo TCP, o protocolo UDP não garante a entrega dos datagramas enviados. Se por um lado os datagramas podem ser perdidos durante a transmissão, por outro a transferência das informações ganha muito em desempenho. Para o caso específico de um servidor de comunicação para conversação o mais importante é o desempenho. Caso algum datagrama seja perdido durante a transmissão o prejuízo ocasionado é muito menor do que quando algum dado adquirido do processo ao algum parâmetro enviado ao servidor seja perdido. O protocolo de comunicação de instruções utilizado por esta tarefa é o LMCP.

Classe `AquisicaoThread` - A classe `AquisicaoThread` é implementada como uma classe derivada da classe `symantec.itools.util.Timer` e é responsável pela tarefa de aquisição de dados do processo. Tem a função de gerenciar o início e término da aquisição e de realizar a obtenção de dados do processo em tempos programados pelo operador. A tarefa possui um temporizador interno que controla o início e a duração da experimentação em andamento. Quando a duração do experimento atinge o mínimo intervalo entre leituras consecutivas (definido neste estudo como 100 ms), a tarefa realiza a leitura das variáveis de aquisição e as armazena em acumuladores até que o intervalo de aquisição definido pelo operador seja atingido. Neste momento, é realizada a média dos dados armazenados nos acumuladores. As médias obtidas para as variáveis de aquisição são então adicionadas ao gráfico correspondente e transmitidos para os usuários remotos. Nesta classe se faz necessária a utilização da metodologia de Métodos Nativos da linguagem Java para interação com a interface de aquisição. Os métodos responsáveis pela interação com a interface de aquisição são declarados nativos usando o modificador `native`. O método `loadLibrary` da classe `java.lang.System` deve ser utilizado para o carregamento da DLL contendo a implementação dos métodos responsáveis pela interação com os canais de entrada e saída da interface de aquisição.

Classe `ControlThread` – a classe `ControlThread` é implementada como uma classe derivada da classe `java.lang.Thread` e é responsável pela implentação da estratégia de controle do processo.

Nesta classe estão definidos os métodos para controle do processo, bem como a definição das variáveis que influenciam na ação de controle. Os set-points das variáveis de controle também são definidos e alterados nesta classe.

3.3.2. Implantação do Sistema na Unidade de Microgravimetria LABORE/UFSC

A Unidade de Microgravimetria utilizada neste trabalho se encontra instalada no Laboratório de Cinética, Catálise e Reatores Químicos – LABORE – do Departamento de Engenharia Química da Universidade Federal de Santa Catarina.

No início deste trabalho, a unidade já se encontrava montada e operante. A aquisição de dados da unidade era realizada pelo aplicativo SuperHet for Windows versão 1.0 (1995), desenvolvido no próprio laboratório.

Para a aquisição de dados da unidade utiliza-se uma interface MQI12/8PC fabricada pela Microquímica Indústria e Comércio, a qual está conectada a uma unidade amplificadora de sinal MQLM-01. Essa unidade por sua vez possui um canal de entrada para a microbalança Cahn 2000.

A única variável de automação da unidade é a massa medida pela microbalança. A calibração desta variável foi realizada utilizando-se pesos padrão pesados em balança analítica.

A automatização da unidade inclui, portanto, a leitura de uma variável, conforme especificado na Tabela 3.6.

Tabela 3.6 – Variáveis de automatização da Unidade de Microgravimetria LABORE/UFSC

Variável	Tipo de automatização
Massa (mg)	Leitura

O sistema Cliente/Servidor Java foi instalado em um computador equipado com processador 486 de 66 MHz, com 16MB de memória RAM e rodando Windows 95. O interpretador utilizado para execução do aplicativo servidor foi o Symantec® Java! Bytecode Compiler versão 2.00b21 (1996), o qual acompanha o pacote de desenvolvimento Visual Café Pro 1.0.

Em virtude desta unidade possuir apenas variáveis de leitura, o sistema Cliente/Servidor Java foi implementado sem a utilização da classe `ControlThread`, descrita anteriormente.

Para avaliação do sistema na Unidade de Microgravimetria foram realizados as seguintes medidas e experimentos:

- Medida do tempo de resposta do servidor: utilizando o mesmo computador utilizado para avaliação dos simuladores matemáticos via Internet, foi avaliado o tempo de resposta do servidor. Foram amostradas medidas do tempo de resposta do servidor a cada dez segundos.

- Experimentos para determinação da taxa de evaporação: foram realizados experimentos na Unidade de Microgravimetria LABORE/UFSC com vários compostos voláteis com o objetivo de determinação da curva de evaporação dos mesmos. Um pedaço de papel Copimax Alcalino Laser, com área de 1,0 cm^2 e gramatura de 75g/m^2, foi utilizado para embeber os solventes. A microbalança foi inicialmente zerada com o papel seco. Após a saturação por imersão do papel com o composto em estudo durante cinco minutos, o mesmo foi inserido na microbalança e iniciada a aquisição da variação de massa com o tempo.

3.3.3. Automatização e Implantação do Sistema na Unidade de Ultrafiltração Tangencial LABSEM/UFSC

A Unidade de Ultrafiltração Tangencial utilizada neste trabalho se encontra instalada no Laboratório de Separação por Membranas – LABSEM – do Departamento de Engenharia Química da Universidade Federal de Santa Catarina.

A unidade foi fabricada sob encomenda pela empresa Netzsch do Brasil Indústria e Comércio Ltda. e instalada em dezembro de 1997. A Figura 3.8 mostra o fluxograma da unidade de ultrafiltração tangencial.

A unidade é montada sobre uma base móvel com armação em aço inoxidável com rodízios, na qual são fixados todos os componentes, e se destina a filtrar um volume de 5 a 10 l/h. A pressão máxima de trabalho da unidade é de 4 bar. A unidade é equipada com tubulações e conexões em aço inoxidável AISI 304 e vedações EPDM.

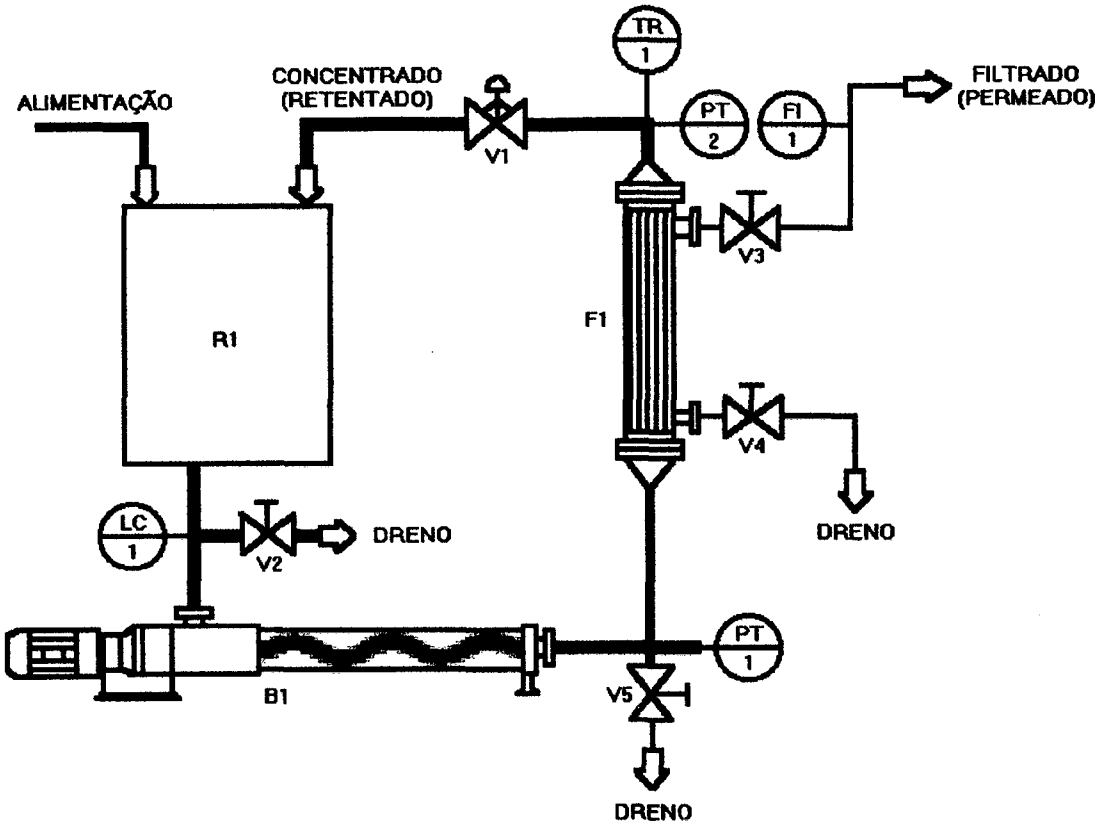


Figura 3.8 – Fluxograma da Unidade de Ultrafiltração LABSEM/UFSC

A Unidade de Ultrafiltração Netzsch é constituída dos seguintes componentes:

– Bomba Nemo (B1) – bomba de alimentação/recirculação com potência instalada de 1,5 CV (motor trifásico de corrente alternada 220-380 V, 60 Hz, 850 RPM, IP-54 fabricado pela Weg Motores S/A) e vazão de recirculação de 2,4 m^3/h.

– Reservatório (R1) – reservatório em aço inoxidável AISI 304 equipado com camisa dupla e com capacidade útil de 50 litros.

– Módulo de Filtração (F1) – módulo com corpo cilíndrico em aço inoxidável AISI 304, com capacidade para uma membrana de filtração. A fixação do módulo a tubulação é realizada por meio de uniões sanitárias, permitindo fácil desmontagem. O módulo foi instalado com uma membrana cerâmica alfa-óxido de alumínio de 1,0 m de comprimento, área de filtragem de 0,2 m^2, 19 canais com diâmetro de 3,3 mm cada e porosidade de 0,05 μm.

– Válvulas Manuais (V2/3/4/5) – válvulas utilizadas principalmente para limpeza da unidade.

- Dispositivo contra trabalho a seco (LC1) – sonda de proteção da bomba, modelo TSP 010805, equipada com microcontrolador de nível.

- Termoresistência (TR1) – sensor tipo PT-100 para medida da temperatura do retentado, fabricada em aço inoxidável 304. Sinal de saída de 4-20 mA.

- Rotâmetro (FI) – indicador/transmissor elétrico contínuo cuja medida é proporcional ao nível (vazão) ocupado pelo flutuador no rotâmetro. Sensor utilizado para medida do fluxo permeado, modelo R0867, com transmissor 1273, com sinal de saída de 4-20 mA e vazão de 4-40 l/h.

- Manômetros eletrônicos (PT1/2) – sensores para medida da pressão do sistema antes e após o módulo de filtração, fabricado pela Danfoss A/S, modelo MBS 4010. Sinal de saída de 4-20 mA.

- Válvula Pneumática (V1) – válvula borboleta reguladora de pressão equipada com atuador pneumático com posicionador eletro-pneumático modelo SRI 986. Sinal de entrada de 4-20 mA e sinal de saída para o atuador de 0-100% da alimentação. O posicionador é alimentado pela linha de ar comprimido do Departamento, cuja pressão máxima é de 7 bar.

- Conversor de Freqüência – conversor de freqüência do tipo PWM, fabricante Weg Automação Ltda., modelo CFW-07, que permite a variação da velocidade de motores trifásicos padrões. Comando por painel ou através de entrada analógica. Sinal de entrada de 0-10 V, 0-20 mA ou 4-20 mA.

Para a automatização da unidade utilizou-se um cartão multifunção ISA modelo CIO-DAS08/Jr-AO, fabricado pela ComputerBoards Inc., com 8 entradas analógicas, 2 saídas analógicas, resolução de 12 bits, taxa de aquisição de 20kHz, 31 canais digitais e 3 contadores rápidos de 1 MHz. A automatização da unidade inclui a leitura de quatro variáveis e a escrita de uma variável, conforme especificado na Tabela 3.7. A unidade possibilita ainda a variação manual da freqüência do motor que alimenta a bomba, através do conversor CFW-07.

Para as variáveis de leitura foram realizados testes de calibração com o objetivo de se obter as constantes de conversão.

Tabela 3.7 – Variáveis de automatização da Unidade de Ultrafiltração LABSEM/UFSC

Variável	Tipo de automatização
Temperatura (°C)	Leitura
Fluxo Permeado (l/h)	Leitura
Pressão na entrada do módulo de filtração (bar)	Leitura
Pressão na saída do módulo de filtração (bar)	Leitura
Abertura da válvula pneumática (0-100% aberta)	Escrita

Para a temperatura, a calibração da termoresistência foi realizada com o auxílio de um sensor de temperatura analógico, calibrado previamente. Para o fluxo permeado, a calibração do rotâmetro foi realizada utilizando-se uma proveta graduada de 500 ml e um cronômetro digital. Para as pressões antes e após o módulo de filtração, a calibração foi auxiliada com um manômetro analógico, calibrado previamente.

Para a variável de escrita foi determinado quais os valores possíveis nos quais seria possível variar a abertura da válvula pneumática, para determinada freqüência aplicada à bomba, sem que fosse ultrapassada a pressão máxima de operação da unidade – 4 bar. Para tanto, os experimentos consistiam em fixar a freqüência da bomba em determinado valor e variar a abertura da válvula pneumática entre totalmente aberta e totalmente fechada, anotando-se a pressão do sistema obtida através dos sensores de pressão.

O sistema Cliente/Servidor Java foi instalado em um computador equipado com processador Pentium de 200 MHz, com 32MB de memória RAM e rodando Windows 95. O interpretador utilizado para execução do aplicativo servidor foi o mesmo utilizado para a Unidade de Microgravimetria - Symantec® Java! Bytecode Compiler versão 2.00b21 (1996).

Devido à ausência de estratégias de controle nesta Unidade, durante esta etapa do trabalho, o sistema Cliente/Servidor Java foi implementado com a utilização da classe `ControlThread` somente para definição e alteração do set-point da variável de escrita, e para aplicação de restrições quanto a valores possíveis em função da freqüência utilizada pela bomba.

Para avaliação do sistema na Unidade de Ultrafiltração foram realizados as seguintes medidas e experimentos:

- Medida do tempo de resposta do servidor: o procedimento para medida do tempo de resposta do servidor é equivalente ao utilizado para a Unidade de Microgravimetria LABORE/UFSC.

- Experimentos para avaliação da Unidade de Ultrafiltração LABSEM/UFSC em operação: foram realizados experimentos utilizando-se água destilada para determinação das condições operacionais da Unidade.

Capítulo 4

RESULTADOS E DISCUSSÃO

4.1. APLICATIVOS DE ENSINO

Para analisar a aplicabilidade do pacote matemático Java labore.math e da linguagem Java no ensino para Engenharia Química, cinco simuladores matemáticos implementados como applets Java foram desenvolvidos.

As interfaces gráficas com o usuário apresentadas pelos simuladores são semelhantes e são compostas por três seções principais:

Parâmetros: interface que possibilita a alteração dos parâmetros do problema e do método matemático utilizado, através de caixas de edição.

Simulação: interface que possibilita o controle do processo de simulação matemática e a visualização do progresso da mesma e dos resultados obtidos, na forma de gráficos. Nesta interface pode-se determinar um dos parâmetros de avaliação dos simuladores desenvolvidos: o tempo de resolução.

Resultados Tabulares: interface com a apresentação dos resultados obtidos na simulação em um formato tabular.

Quatro dos cinco simuladores desenvolvidos apresentam esta estrutura em sua interface gráfica. A única exceção é o primeiro simulador demostrado neste capítulo, o Simulador de um Reator CSTR Encamisado, que está dividido em seis seções. Além das três seções comuns aos outros simuladores, este apresenta mais três seções adicionais:

Problema: interface com a descrição do problema, incluindo a apresentação gráfica do "layout" do reator e da reação a serem simulados. No "layout" do reator são demonstrados os fluxos de entrada e saída do processo.

Dados: interface com a exibição de parâmetros físicos pré-determinados do problema e considerações fenomenológicas realizadas para obtenção do modelo matemático.

Modelagem: interface com a descrição das equações de conservação que compõem o modelo matemático a ser simulado.

Para a avaliação dos applets simuladores serão analisados dois parâmetros: o tempo de transferência e tempo de resolução.

O tempo de transferência depende basicamente de dois fatores: o tamanho das classes utilizadas pelo applet e o tempo de transmissão de dados entre o host do usuário e o host onde o applet reside. Para determinação do tamanho total das classes utilizadas pelo applet não foram consideradas as classes que compõem a interface de programação API Java, pois as mesmas são parte integrante do interpretador incorporado no navegador e não são transmitidas pela rede. O tempo médio de transmissão de dados entre os host local e remoto durante os experimentos foi de 184,92 ms, para pacotes de dados de 32 bytes utilizando o aplicativo TCP/IP ping[1]. Um terceiro fator é a existência de classes utilizadas pelo applet na memória cache do navegador. No entanto, a influência deste fator foi eliminada com a limpeza da memória cache antes do início dos experimentos.

O tempo de resolução é obtido do próprio applet, em sua seção Simulação.

4.1.1. Simulador de um Reator CSTR Encamisado

Este applet simula um reator químico no qual ocorre uma reação irreversível exotérmica de 1ª ordem. Uma camisa de resfriamento mantém a temperatura sob condições desejadas. O sistema de equações diferenciais ordinárias resultante do balanço molar para o componente A, o balanço de energia no reator e na camisa de resfriamento é resolvido por uma rotina de integração Runge-Kutta de 4ª ordem, com monitoramento do erro e passo de integração auto-ajustável. As Figura 4.1 a 4.6 mostram a interface gráfica com o usuário desenvolvida para o applet simulador, destacando suas várias seções.

[1] O programa PING opera enviando uma solicitação ICMP (Internet Control Message Protocol) para uma máquina na Internet. Se o software IP da máquina de destino recebe uma solicitação ICMP, ela enviará uma resposta imediatamente. Após sua execução o programa PING exibe as estatísticas de transmissão dos dados, indicando o status do canal de comunicação (Porta Digital Internet, 1996).

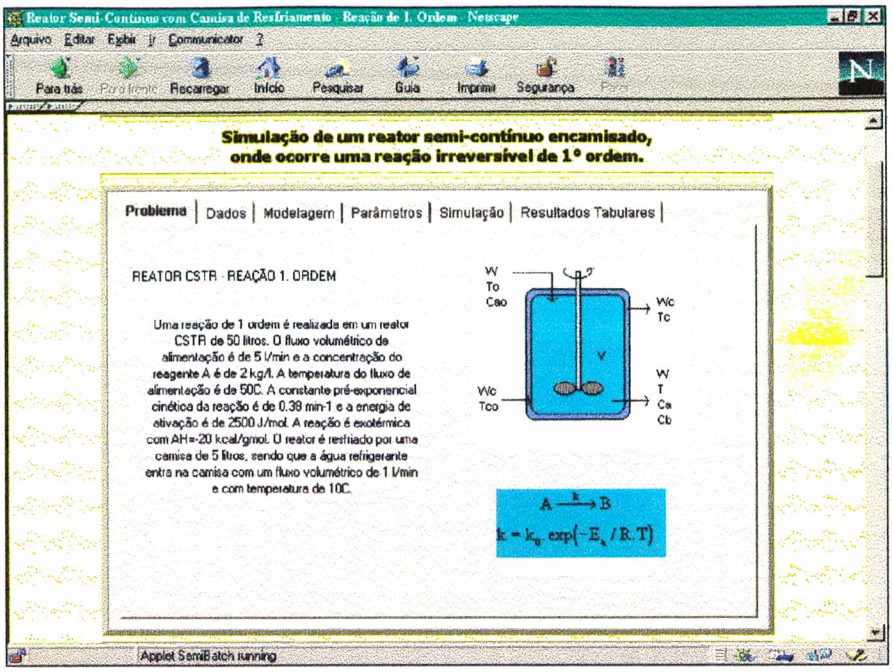


Figura 4.1 – Interface gráfica com o usuário desenvolvida para o Simulador de um Reator CSTR Encamisado - seção Problema

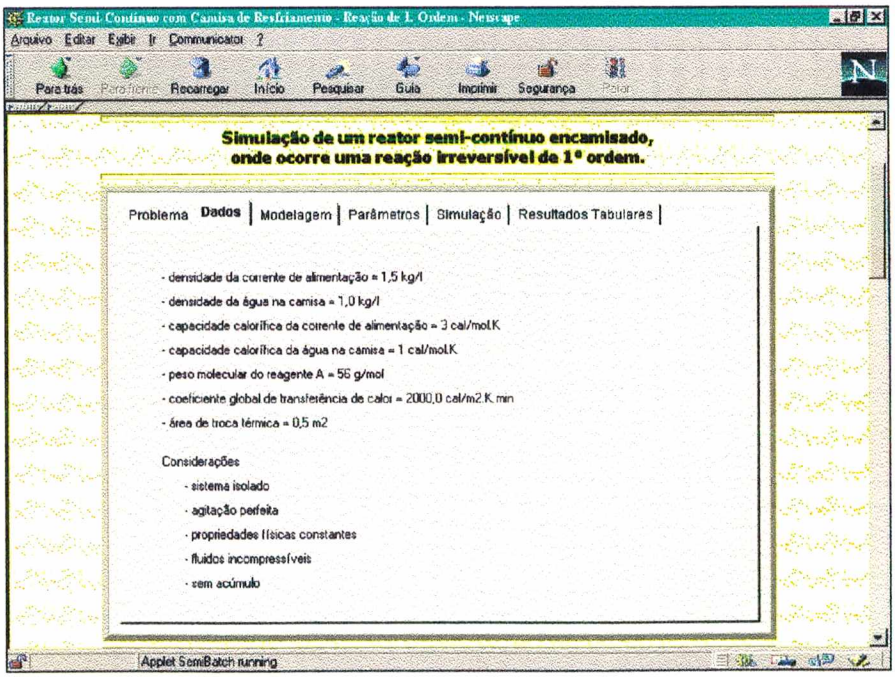


Figura 4.2 – Interface gráfica com o usuário desenvolvida para o Simulador de um Reator CSTR Encamisado - seção Dados

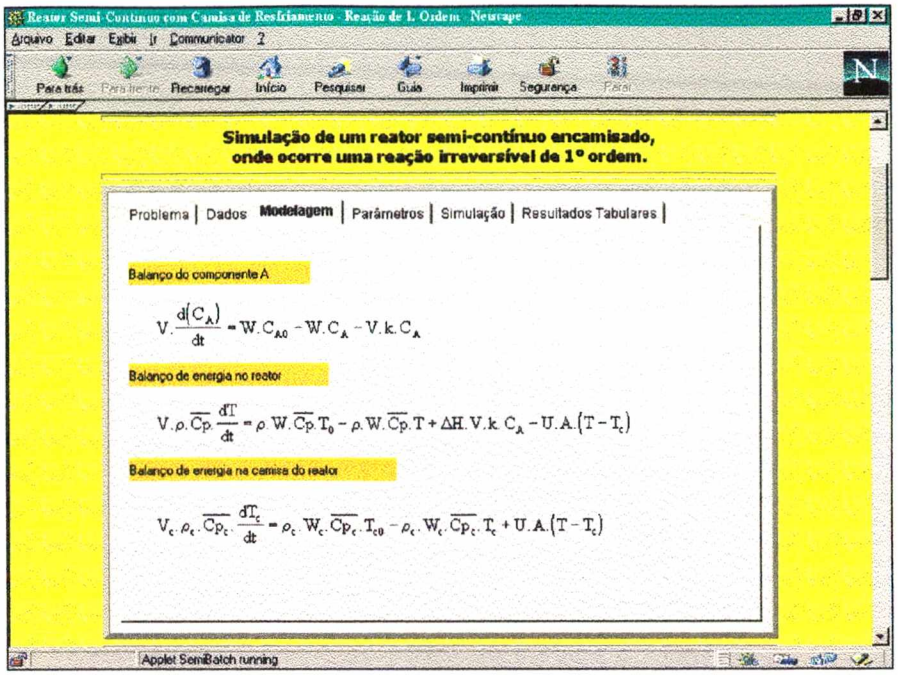


Figura 4.3 – Interface gráfica com o usuário desenvolvida para o Simulador de um Reator CSTR Encamisado - seção Modelagem

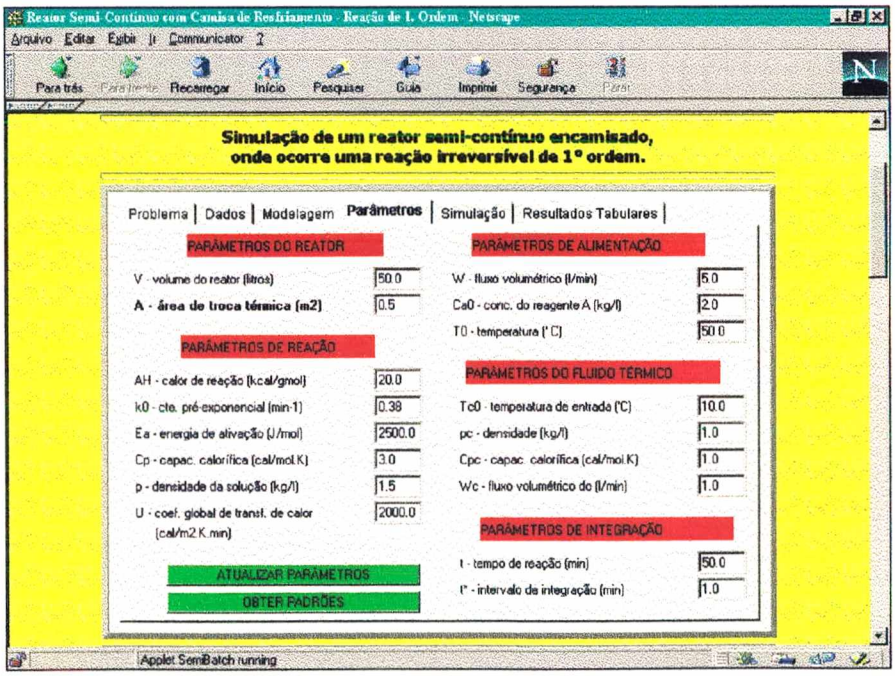


Figura 4.4 – Interface gráfica com o usuário desenvolvida para o Simulador de um Reator CSTR Encamisado - seção Parâmetros

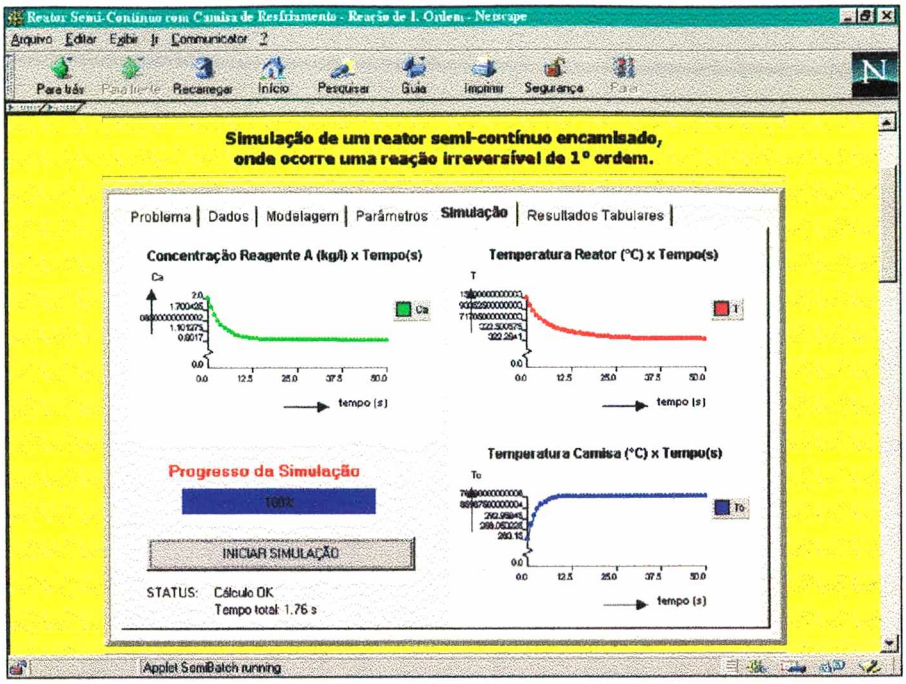
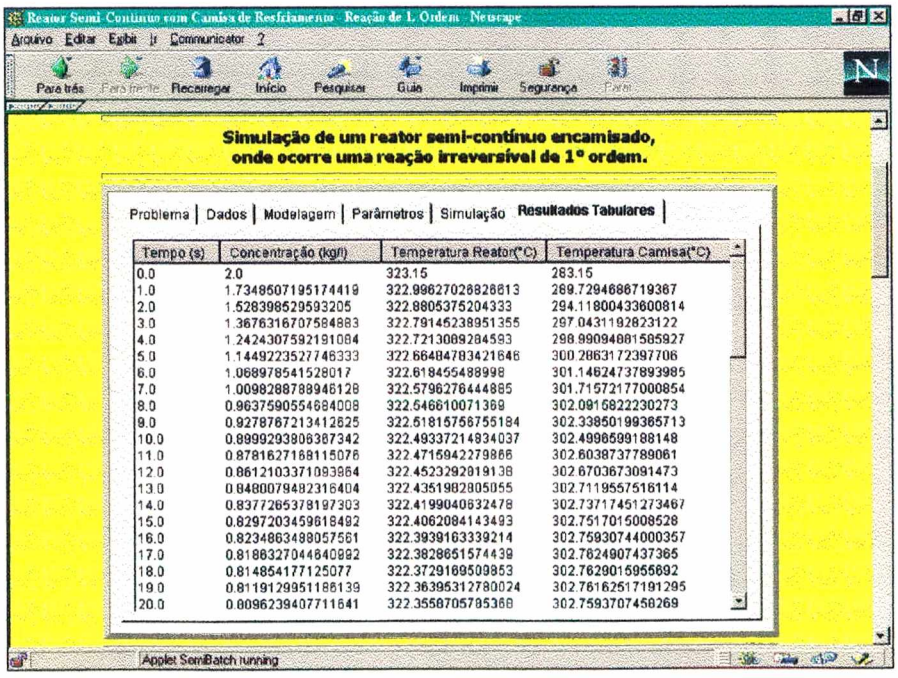


Figura 4.5 – Interface gráfica com o usuário desenvolvida para o Simulador de um Reator CSTR Encamisado - seção Simulação



Tempo (s)	Concentração (kg/l)	Temperatura Reator(°C)	Temperatura Camisa(°C)
0.0	2.0	323.15	283.15
1.0	1.7348507195174419	322.99627028828613	289.7294686719367
2.0	1.528398529593205	322.8805375204333	294.11800433600814
3.0	1.3676316707584883	322.79145238951355	297.0431192823122
4.0	1.2424307592191084	322.7213089284593	298.99094681585927
5.0	1.1449223527746333	322.66484783421646	300.2863172397706
6.0	1.068978541528017	322.61845488998	301.14624737893985
7.0	1.0098288788946128	322.5796276444885	301.71572177000854
8.0	0.9637590554884008	322.546610071369	302.0915822230273
9.0	0.9278767213412625	322.51815756755184	302.33850199365713
10.0	0.8999293806387342	322.49337214834037	302.4996599188148
11.0	0.8781627168115078	322.4715942279866	302.6038737789061
12.0	0.8612103371093864	322.4523292819138	302.6703673091473
13.0	0.8480079482316404	322.4351982805055	302.7119557516114
14.0	0.8377265378197303	322.41990406832478	302.73717451273467
15.0	0.8297203459618492	322.4062084143493	302.7517015008528
16.0	0.8234863488057561	322.3939163339214	302.75930744000357
17.0	0.8186327044840892	322.3828651574439	302.7624907437365
18.0	0.814854177125077	322.3729169509853	302.7629015956892
19.0	0.8119129951186139	322.36395312780024	302.76162517191295
20.0	0.8096239407711641	322.3558705785368	302.7593707456269

Figura 4.6 – Interface gráfica com o usuário desenvolvida para o Simulador de um Reator CSTR Encamisado - seção Resultados Tabulares

O tempo de transferência médio do simulador foi de 63,2 segundos. A Tabela 4.1 relaciona as classes que necessitam ser transferidas pela rede para execução do applet simulador.

Tabela 4.1 – Classes utilizadas pelo Simulador de um Reator CSTR Encamisado

CLASSE	TAMANHO (Kbytes)
`SemiBatch`	15,50
`JackedSemiBatch`	1,54
Classes do pacote labore.graph (1)	10,80
Classes do pacote labore.math (1)	4,00
Classes do pacote symantec.beans (1)	0,56
Classes do pacote symantec.itools.awt (13)	52,60
Classes do pacote symantec.itools.awt.util (1)	5,49
Classes do pacote symantec.itools.lang (1)	1,74
Classes do pacote symantec.itools.multimedia (1)	2,54
Classes do pacote symantec.itools.net (1)	0,68
Classes do pacote warhol.charts (6)	16,00
TOTAL – 28 classes	**111,45**

A Tabela 4.2 apresenta os resultados obtidos para o tempo de resolução do simulador. São descritos na tabela o número de pontos utilizados na simulação de 50 minutos do processo, a memória total utilizada pela Máquina Virtual Java, o percentual de memória livre e o tempo médio de resolução.

Tabela 4.2 – Tempo de resolução para o Simulador de um Reator CSTR Encamisado

Número de Pontos	Memória Total (Kbytes)	Memória Livre (%)	Tempo médio de Resolução (s)
50	2064	65	1,74
66	2064	64	2,40
100	784	30	4,83
200	2064	59	17,99
500	1808	40	81,22

A Figura 4.7 mostra o acréscimo no tempo de resolução necessário conforme o número de pontos utilizados na simulação.

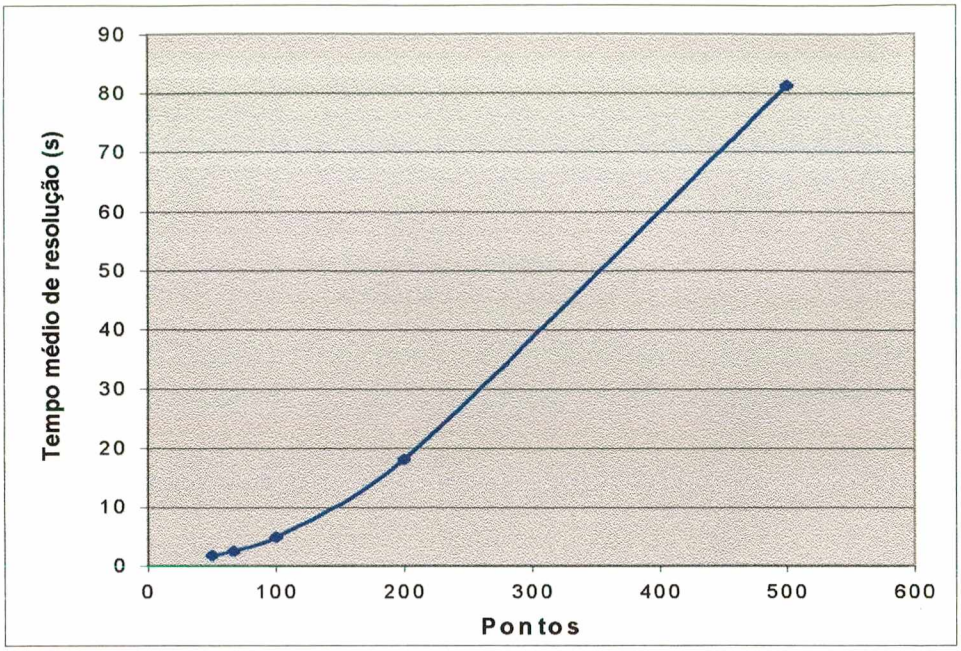


Figura 4.7 – Aumento do tempo de resolução em função do número de pontos para o Simulador de um Reator CSTR Encamisado

4.1.2. Simulador de um Reator Adiabático em Batelada

Como um exemplo de operação em batelada, a produção de propilenoglicol é realizada sob condições adiabáticas. Novamente o problema pode ser modelado como um sistema de equações diferenciais ordinárias e resolvido por uma rotina de integração Runge-Kutta.

As Figuras 4.8 a 4.10 mostram a interface gráfica com o usuário desenvolvida para o applet simulador, destacando as várias seções que o compõem.

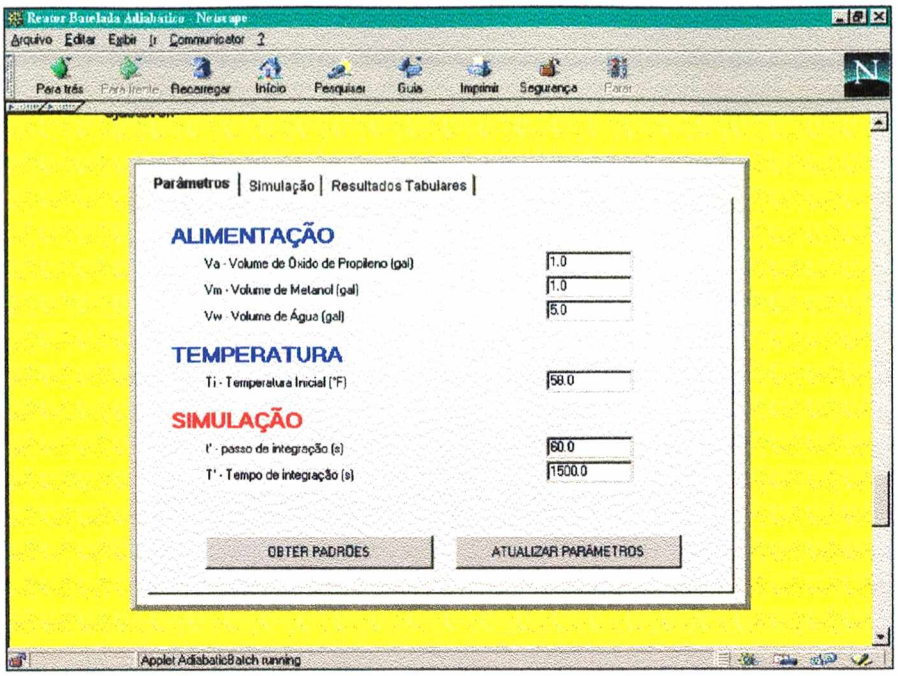


Figura 4.8 – Interface gráfica com o usuário desenvolvida para o Simulador de um Reator Adiabático em Batelada - seção Parâmetros

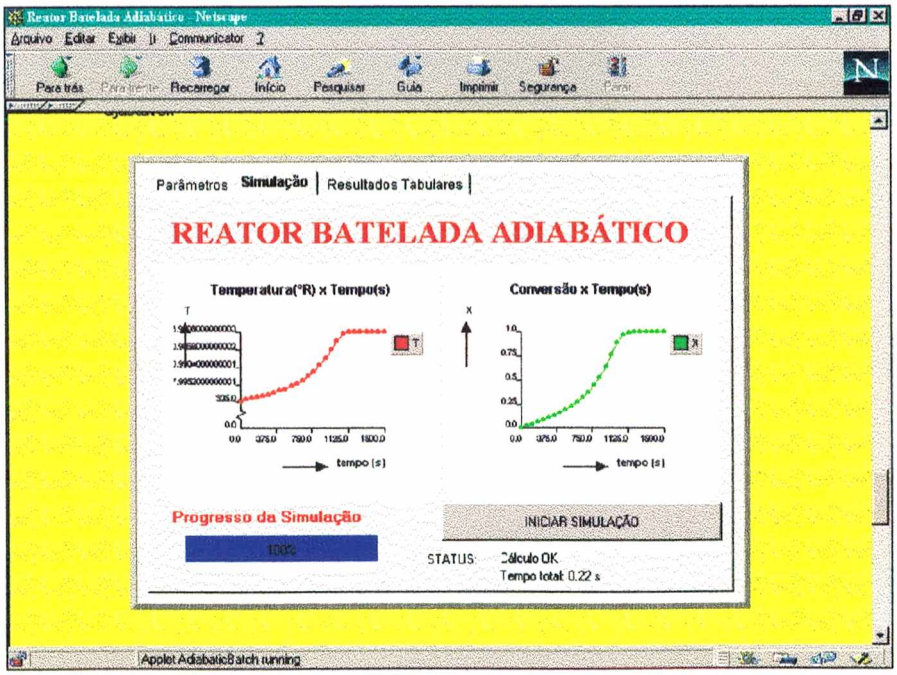


Figura 4.9 – Interface gráfica com o usuário desenvolvida para Simulador de um Reator Adiabático em Batelada - seção Simulação

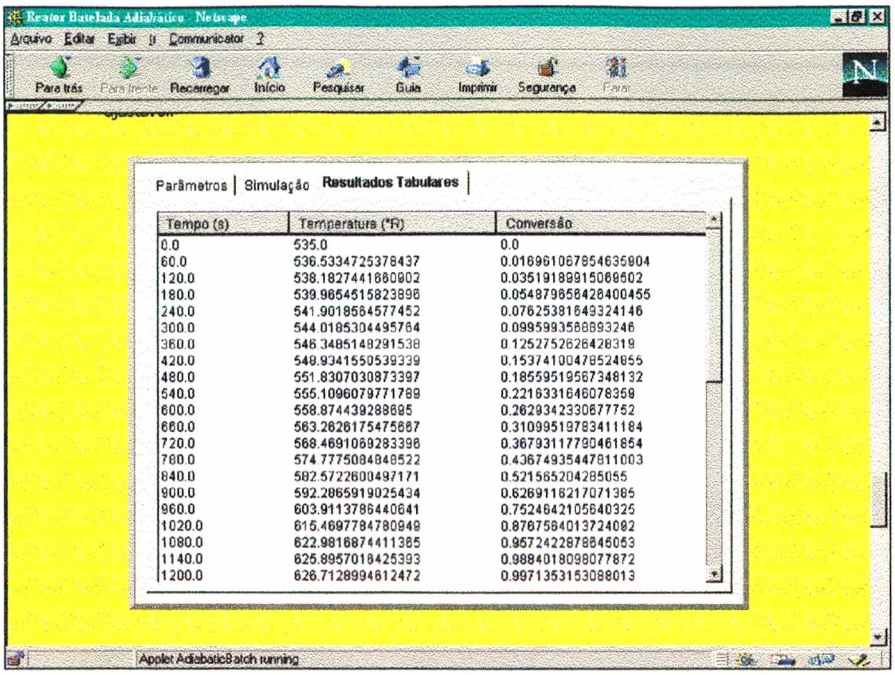


Figura 4.10 – Interface gráfica com o usuário desenvolvida para o Simulador de um Reator Adiabático em Batelada - seção Resultados Tabulares

O tempo de transferência médio do simulador foi de 62,8 segundos. A Tabela 4.3 relaciona as classes que necessitam ser transferidas pela rede Internet para execução do applet simulador.

Tabela 4.3 – Classes utilizadas pelo Simulador de um Reator Adiabático em Batelada

CLASSE	**TAMANHO (Kbytes)**
`AdiabaticBatch`	8,60
`AdiabaticBatchReactor`	1,99
`Constantes`	0,35
Classes do pacote labore.graph (1)	10,80
Classes do pacote labore.math (1)	4,00
Classes do pacote symantec.beans (1)	0,56
Classes do pacote symantec.itools.awt (12)	50,10
Classes do pacote symantec.itools.awt.util (1)	5,49
Classes do pacote symantec.itools.lang (1)	1,74
Classes do pacote warhol.charts (6)	16,00
TOTAL – 26 classes	**99,63**

A Tabela 4.4 apresenta os resultados obtidos para o tempo de resolução do simulador. São descritos na tabela o número de pontos utilizados na simulação de 1500 segundos do processo, a memória total utilizada pela Máquina Virtual Java, o percentual de memória livre e o tempo médio de resolução.

Tabela 4.4 – Tempo de resolução para o Simulador de um Reator Adiabático em Batelada

Número de Pontos	Memória Total (Kbytes)	Memória Livre (%)	Tempo médio de Resolução (s)
25	784	45	0,31
33	784	44	0,21
50	784	43	0,47
100	784	39	0,90
250	784	28	2,58

A Figura 4.11 mostra o aumento no tempo de resolução necessário conforme o número de pontos utilizados na simulação.

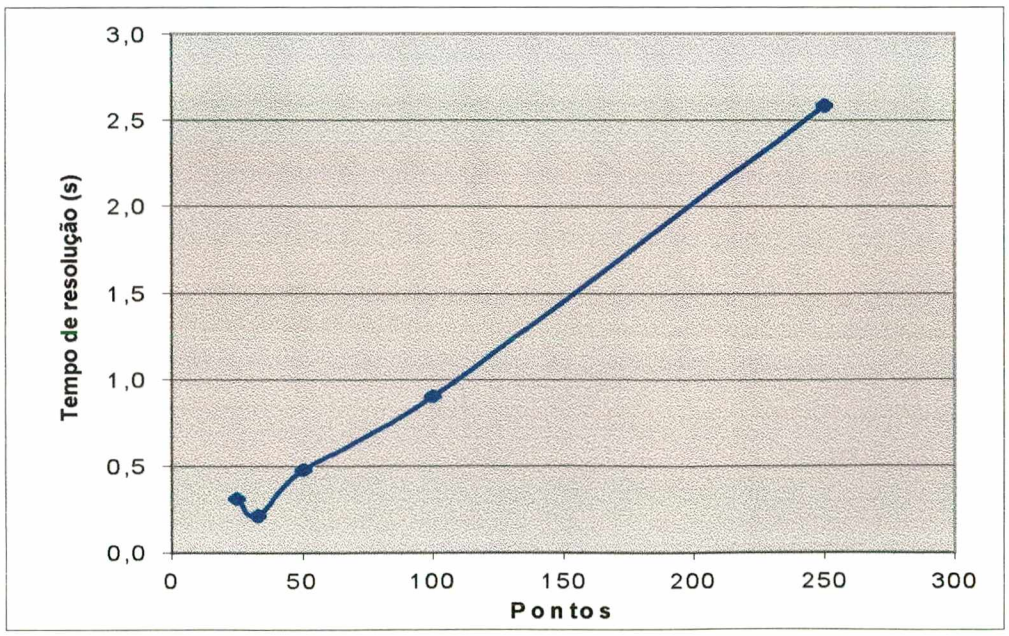


Figura 4.11 – Aumento do tempo de resolução em função do número de pontos para o Simulador de um Reator Adiabático em Batelada

4.1.3. Simulador de um Conversor de SO_2

Este applet realiza a simulação de uma oxidação industrial de SO_2 em um conversor de leito fixo não-adiabático que é parte do processo de produção de ácido sulfúrico a partir do enxofre. O catalisador é empacotado em tubos e operação do tipo fluxo pistonado é assumida. O modelo matemático consiste de um sistema de três equações diferenciais derivadas da equações de projeto do PFR (balanço molar), balanço de energia e equações de queda de pressão (balanço de momento). O sistema de equações diferenciais ordinárias é resolvido numericamente por uma rotina Runge-Kutta de 4ª ordem.

As Figuras 4.12 a 4.14 mostram a interface gráfica com o usuário desenvolvida para o applet simulador, destacando as várias seções que o compõem.

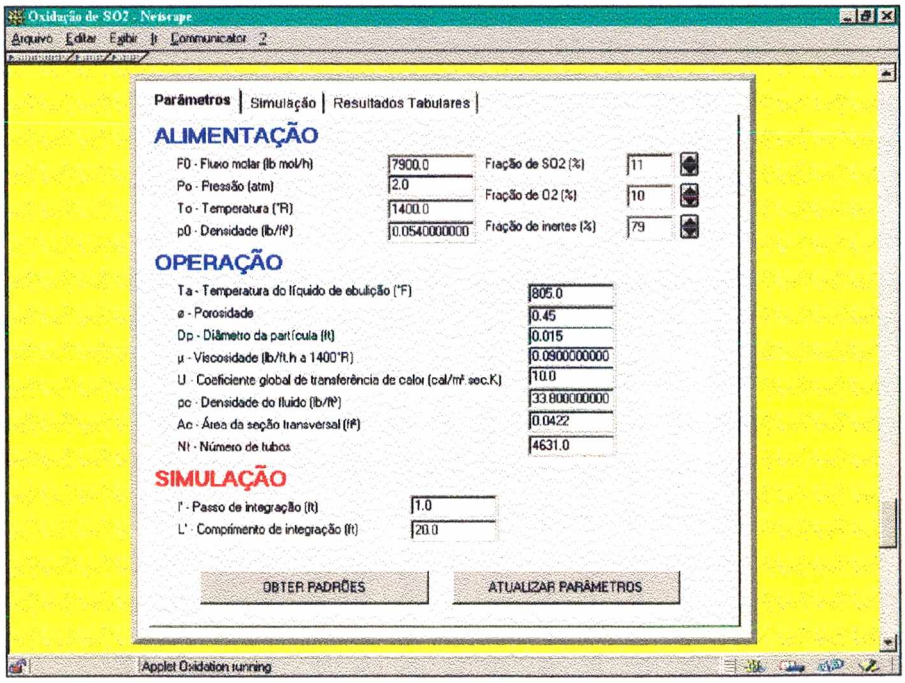


Figura 4.12 – Interface gráfica com o usuário desenvolvida para o Simulador de um Conversor de SO_2 - seção Parâmetros

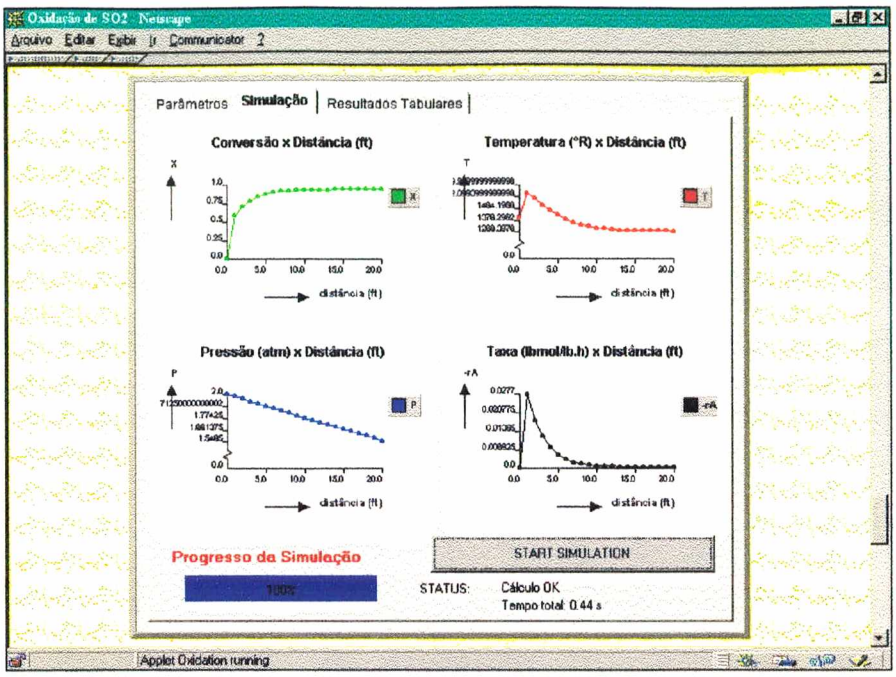
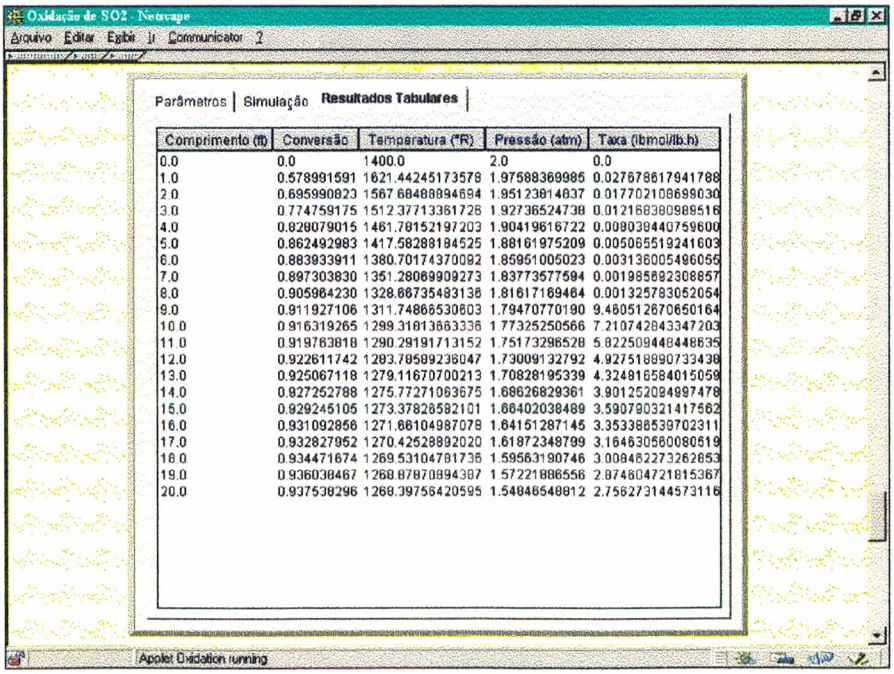


Figura 4.13 – Interface gráfica com o usuário desenvolvida para o Simulador de um Conversor de SO_2 - seção Simulação



Comprimento (ft)	Conversão	Temperatura (°R)	Pressão (atm)	Taxa (lbmol/lb.h)
0.0	0.0	1400.0	2.0	0.0
1.0	0.578991591	1621.44245173578	1.97588369985	0.027678617941788
2.0	0.695990823	1567.68488894694	1.95123814037	0.017702108698030
3.0	0.774759175	1512.37713361726	1.92736524738	0.012168380988516
4.0	0.829079015	1461.78152197203	1.90419616722	0.008038440759600
5.0	0.862492983	1417.58288184525	1.88161975209	0.005065519241603
6.0	0.883933911	1380.70174370092	1.85951005023	0.003136005496055
7.0	0.897303830	1351.28069909273	1.83773577594	0.001985692308857
8.0	0.905964230	1328.66735483138	1.81617169464	0.001325783052054
9.0	0.911927106	1311.74866530803	1.79470770190	9.460512670650164
10.0	0.916319265	1299.31813863336	1.77325250566	7.210742843347203
11.0	0.919763818	1290.29191713152	1.75173296528	5.822509448448835
12.0	0.922611742	1283.70589236047	1.73009132792	4.927518890733438
13.0	0.925067118	1279.11670700213	1.70828195339	4.324816584015059
14.0	0.927252788	1275.77271063675	1.68626829361	3.901252084997478
15.0	0.929245105	1273.37828582101	1.66402038489	3.590790321417562
16.0	0.931092856	1271.66104987078	1.64151287145	3.353386539702311
17.0	0.932827952	1270.42528892020	1.61872348799	3.164630560080519
18.0	0.934471674	1269.53104781736	1.59563190746	3.008482273262853
19.0	0.936038467	1268.87870894387	1.57221886556	2.874604721815367
20.0	0.937538296	1268.39756420595	1.54846548812	2.758273144573116

Figura 4.14 – Interface gráfica com o usuário desenvolvida para o Simulador de um Conversor de SO_2 - seção Resultados Tabulares

O tempo de transferência médio do simulador foi de 74,8 segundos. A Tabela 4.5 relaciona as classes que necessitam ser transferidas pela rede Internet para execução do applet simulador.

Tabela 4.5 – Classes utilizadas pelo Simulador de um Conversor de SO_2

CLASSE	TAMANHO (Kbytes)
`Oxidation`	12,00
`OxidationSO2`	2,67
`Constantes`	0,35
Classes do pacote labore.graph (1)	10,80
Classes do pacote labore.math (1)	4,00
Classes do pacote symantec.beans (1)	0,56
Classes do pacote symantec.itools.awt (18)	57,90
Classes do pacote symantec.itools.awt.util (1)	5,49
Classes do pacote symantec.itools.lang (1)	1,74
Classes do pacote warhol.charts (6)	16,00
TOTAL – 32 classes	**111,51**

A Tabela 4.6 apresenta os resultados obtidos para o tempo de resolução do simulador. São descritos na tabela o número de pontos utilizados na simulação de um reator de 20 pés de comprimento, a memória total utilizada pela Máquina Virtual Java, o percentual de memória livre e o tempo médio de resolução.

Tabela 4.6 – Tempo de resolução para o Simulador de um Conversor de SO_2

Número de Pontos	Memória Total (Kbytes)	Memória Livre (%)	Tempo médio de Resolução (s)
20	784	38	0,45
26	784	37	0,34
40	784	35	0,81
80	784	30	1,43
200	1040	35	2,40

A Figura 4.15 mostra o aumento no tempo de resolução necessário conforme o número de pontos utilizados na simulação.

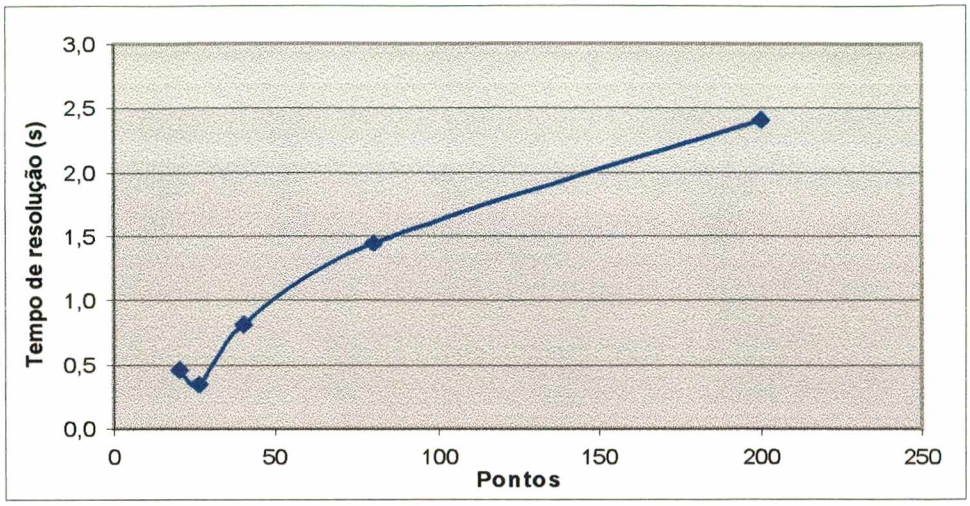


Figura 4.15 – Aumento do tempo de resolução em função do número de pontos para o Simulador de um Conversor de SO_2

4.1.4. Simulador de um Reator CSTR Fluidizado sujeito a Decaimento Catalítico

O decaimento catalítico em um reator CSTR pode ser analisado com este applet. O craqueamento catalítico de um gasóleo leve em fase gasosa foi modelado assumindo-se que o catalisador e a mistura gasosa estão bem misturados no tanque. Como um resultado, uma descrição matemática do sistema, composta da reação de desativação acoplada com a reação de craqueamento principal, conduz a um sistema de equações diferenciais ordinárias que pode ser resolvido por uma rotina Runge-Kutta de 4ª ordem.

As Figura 4.16 a 4.18 mostram a interface gráfica com o usuário desenvolvida para o applet simulador, destacando as várias seções na que o compõem.

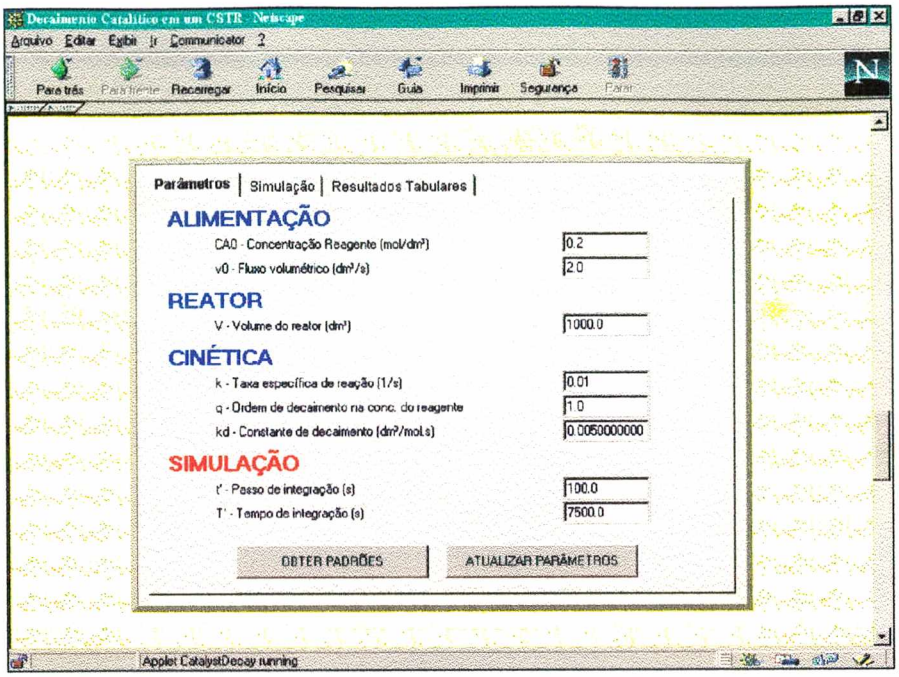


Figura 4.16 – Interface gráfica com o usuário desenvolvida para Simulador de um Reator CSTR Fluidizado sujeito a Decaimento Catalítico - seção Parâmetros

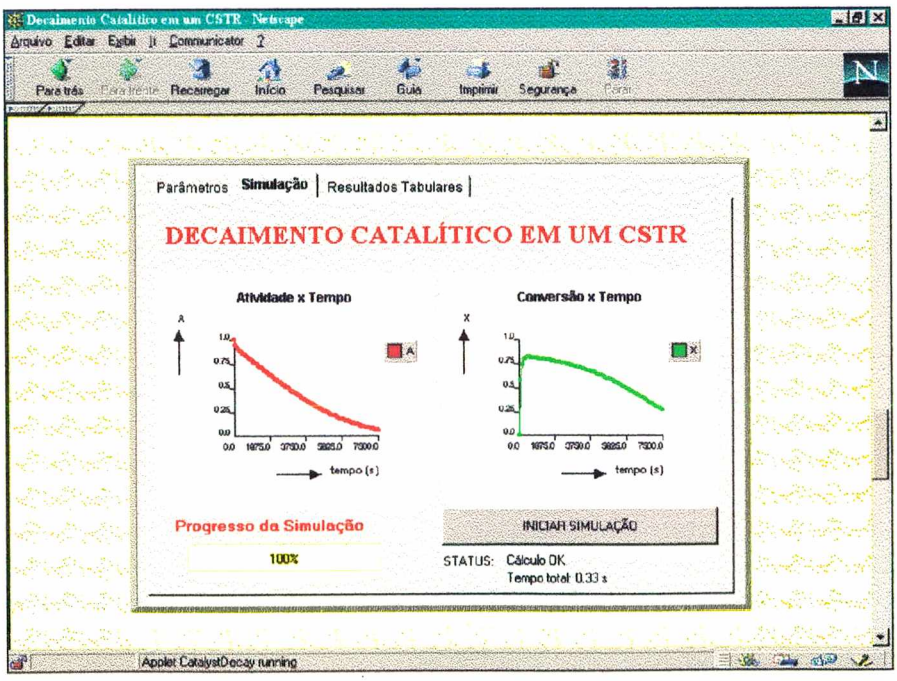


Figura 4.17 – Interface gráfica com o usuário desenvolvida para o Simulador de um Reator CSTR Fluidizado sujeito a Decaimento Catalítico - seção Simulação

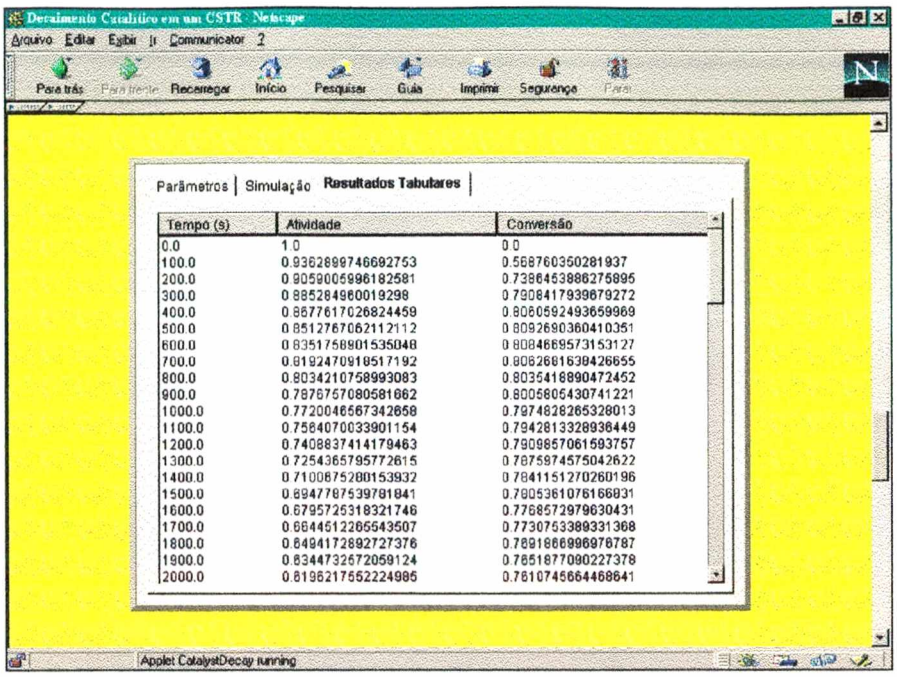


Figura 4.18 – Interface gráfica com o usuário desenvolvida para o Simulador de um Reator CSTR Fluidizado sujeito a Decaimento Catalítico - seção Resultados Tabulares

O tempo de transferência médio do simulador foi de 59,0 segundos. A Tabela 4.7 relaciona as classes que necessitam ser transferidas pela rede Internet para execução do applet simulador.

Tabela 4.7 – Classes utilizadas pelo Simulador de um Reator CSTR Fluidizado sujeito a Decaimento Catalítico

CLASSE	**TAMANHO (Kbytes)**
`CatalystDecay`	9,50
`CatalystDecayCSTR`	0,94
Classes do pacote labore.graph (1)	10,80
Classes do pacote labore.math (1)	4,00
Classes do pacote symantec.beans (1)	0,56
Classes do pacote symantec.itools.awt (12)	50,10
Classes do pacote symantec.itools.awt.util (1)	5,49
Classes do pacote symantec.itools.lang (1)	1,74
Classes do pacote warhol.charts (6)	16,00
TOTAL – 25 classes	**99,13**

A Tabela 4.8 apresenta os resultados obtidos para o tempo de resolução do simulador. São descritos na tabela o número de pontos utilizados na simulação de 125 minutos do processo, a memória total utilizada pela Máquina Virtual Java, o percentual de memória livre e o tempo médio de resolução obtido.

Tabela 4.8 – Tempo de resolução para o Simulador de um Reator CSTR Fluidizado sujeito a Decaimento Catalítico

Número de Pontos	Memória Total (Kbytes)	Memória Livre (%)	Tempo médio de Resolução (s)
75	528	25	0,36
100	784	32	0,46
150	784	28	0,71
300	1040	38	1,38
750	1808	49	5,61

A Figura 4.19 mostra o crescimento no tempo de resolução necessário conforme o número de pontos utilizados na simulação.

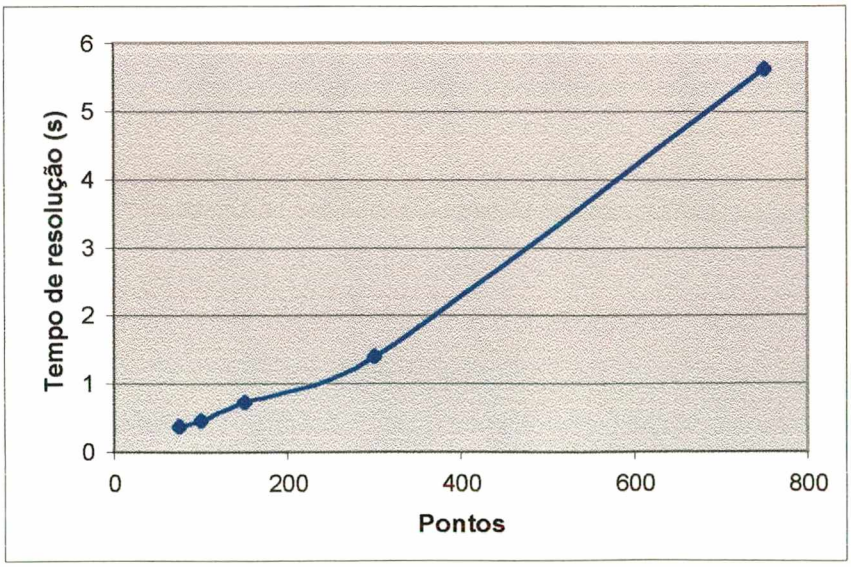


Figura 4.19 – Aumento do tempo de resolução em função do número de pontos para o Simulador de um Reator CSTR Fluidizado sujeito a Decaimento Catalítico

4.1.5. Simulador de um Reator de Leito Fixo com Dispersão Axial, utilizando um Modelo Estacionário Isotérmico

Outro tipo comum de problema encontrado na engenharia de reações químicas é o de problemas de valor de contorno. Problemas de valor de contorno são usados para modelar reações nos níveis de partícula e de reator. Neste simulador foi utilizado o método da colocação ortogonal para resolver um modelo de reator tubular empacotado com dispersão axial de massa. O problema é limitado ao caso de um reator isotérmico operando em estado estacionário. O método da colocação ortogonal foi utilizado para resolver a equação diferencial resultante do balanço molar com condições apropriadas de contorno.

As Figuras 4.20 a 4.22 mostram a interface gráfica com o usuário desenvolvida para o applet simulador, destacando as várias seções na que o compõem.

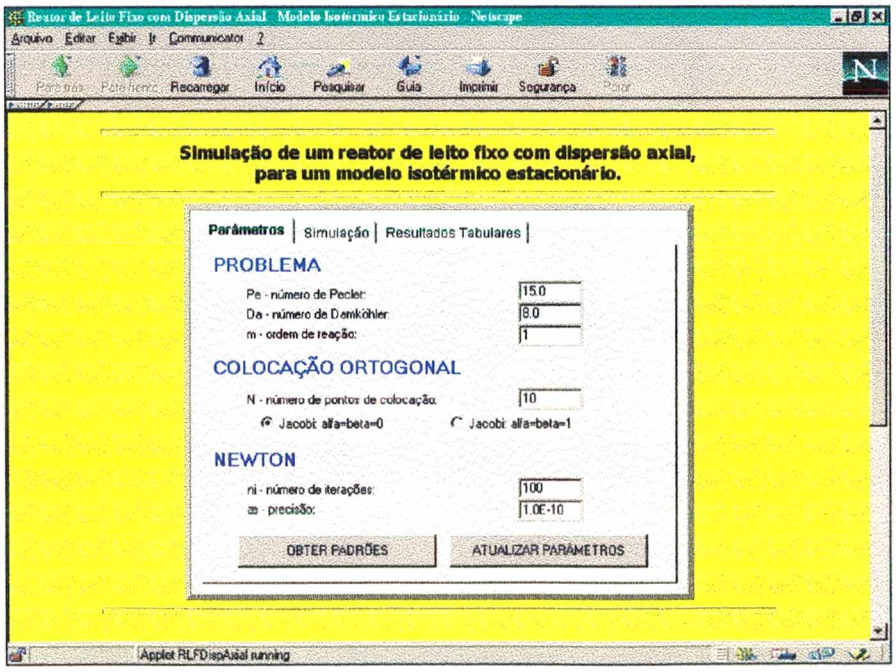


Figura 4.20 – Interface gráfica com o usuário desenvolvida para o Simulador de um Reator de Leito Fixo com Dispersão Axial - seção Parâmetros

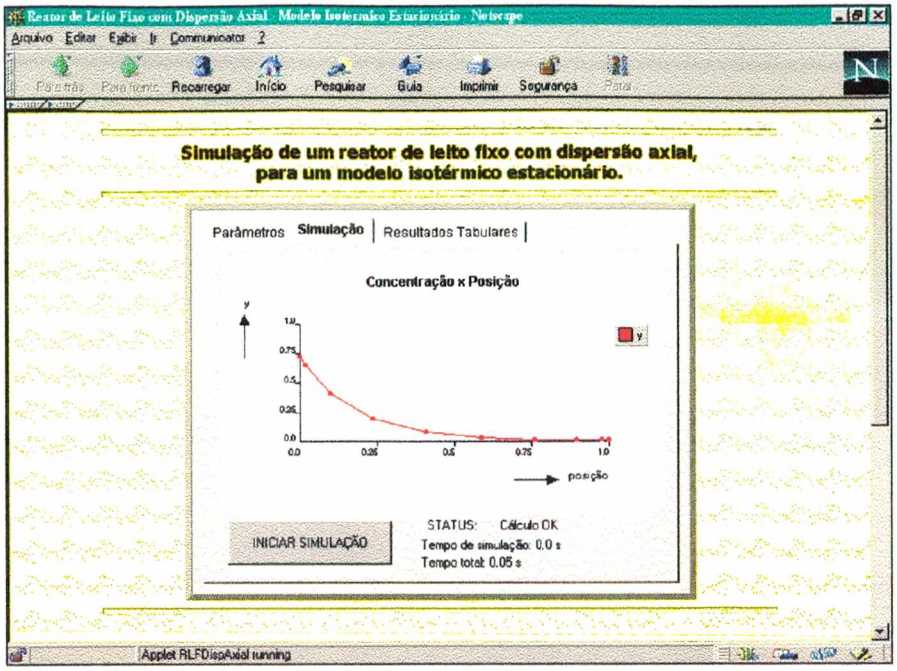


Figura 4.21 – Interface gráfica com o usuário desenvolvida para o Simulador de um Reator de Leito Fixo com Dispersão Axial - seção Simulação

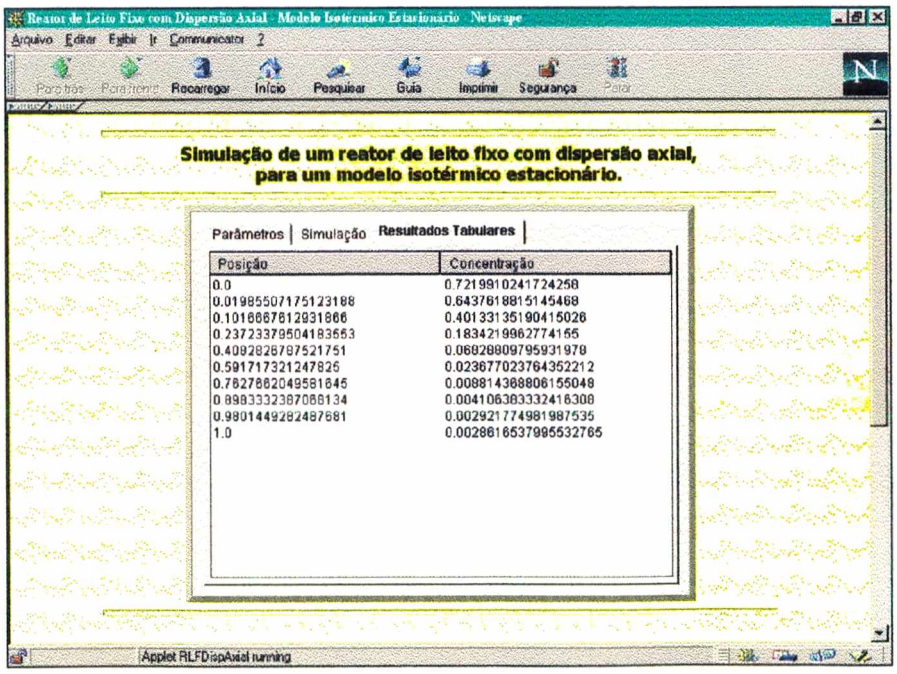


Figura 4.22 – Interface gráfica com o usuário desenvolvida para o Simulador de um Reator de Leito Fixo com Dispersão Axial - seção Resultados Tabulares

O tempo de transferência médio do simulador foi de 62,7 segundos. A Tabela 4.9 relaciona as classes que necessitam ser transferidas pela rede Internet para execução do applet simulador.

Tabela 4.9 – Classes utilizadas pelo Simulador de um Reator de Leito Fixo com Dispersão Axial

CLASSE	TAMANHO (Kbytes)
`RLFDispAxial`	8,62
`LeitoFixoCO`	1,45
Classes do pacote labore.graph (1)	10,80
Classes do pacote labore.math (3)	7,02
Classes do pacote symantec.beans (1)	0,56
Classes do pacote symantec.itools.awt (10)	49,40
Classes do pacote symantec.itools.lang (1)	1,74
Classes do pacote warhol.charts (6)	16,00
TOTAL – 24 classes	**95,59**

A Tabela 4.10 apresenta os resultados obtidos para o tempo de resolução do simulador. São descritos na tabela o número de pontos de colocação utilizados na simulação, a memória total utilizada pela Máquina Virtual Java, o percentual de memória livre e o tempo médio de resolução.

Tabela 4.10 – Tempo de resolução para o Simulador de um Reator de Leito Fixo com Dispersão Axial

Número De Pontos	Memória Total (Kbytes)	Memória Livre (%)	Tempo médio de Cálculo (s)	Tempo médio Total de Resolução (s)
10	528	28	0,01	0,02
50	784	39	0,05	0,10
75	1040	43	0,12	0,22
100	1296	43	0,26	0,40
250	11024	77	3,52	4,14

A Figura 4.23 montra o aumento no tempo de cálculo e no tempo total de resolução, conforme o número de pontos de colocação utilizados na simulação.

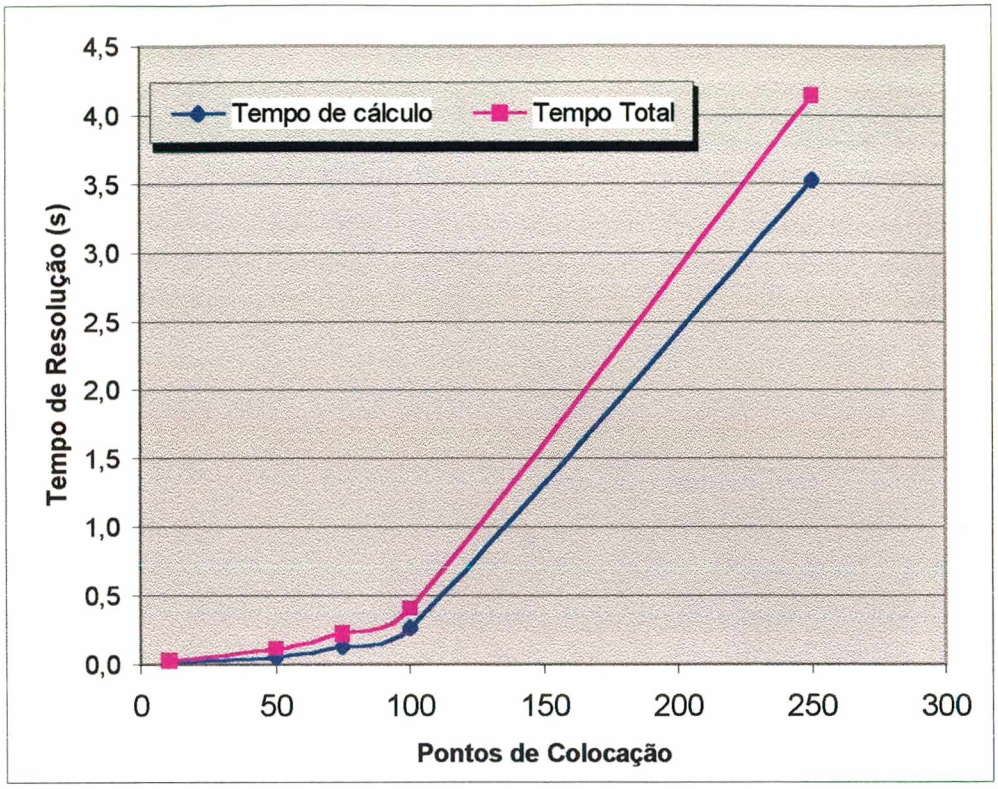


Figura 4.23 – Aumento do tempo de cálculo e no tempo total de resolução em função do número de pontos de colocação para o Simulador de um Reator de Leito Fixo com Dispersão Axial

4.1.6. Avaliações de Desempenho dos Applets Simuladores

A Tabela 4.11 compara os applets simuladores desenvolvidos quanto ao número de classes utilizado, tamanho total de arquivos transferidos e tempo de transferência médio.

Tabela 4.11 – Comparação entre os applets simuladores desenvolvidos

Applet Simulador	Classes Utilizadas	Tamanho dos Arquivos (Kbytes)	Tempo de Transferência (s)
SemiBatch v1.1	28	111,45	63,2
AdiabaticBatch v1.1	26	99,63	59,1
Oxidation v1.1	32	111,51	62,8
CatalystDecay v1.1	25	99,13	74,8
RLFDispAxial v1.1	24	95,59	72,7
MÉDIA	**27**	**103,46**	**64,9**

Visto o tamanho total das classes utilizadas pelos applets simuladores ser praticamente o mesmo, tem-se uma boa correspondência entre os tempos de transmissão e pode-se dizer que o tempo médio de transferência dos applets simuladores, acessando os mesmos de um host com tempo de resposta do servidor de aproximadamente 185 ms, é de cerca de um minuto, mesmo com a memória cache do navegador desabilitada.

Com a utilização da memória cache, após realizar a transferência de qualquer um dos simuladores, a transferência dos demais será em geral praticamente instantânea. Isto porque a maior parte das classes utilizadas é comum a todos os applets (pacotes labore, symantec e warhol).

O tempo de transmissão obtido pelos applets simuladores é bastante influenciado pelas classes responsáveis pelo suporte à interface gráfica com o usuário. Em média, cada applet simulador utiliza 79,3 Kbytes de classes implementadas apenas para este fim – pacotes labore.graph, symantec.itools.awt e warhol.charts – , representando em média 76,7 % do tamanho total dos arquivos utilizados.

O tempo de resolução obtido para os simuladores transferidos pela Internet foi idêntico ao obtido executando-se os simuladores a partir do disco rígido local, para as mesmas condições de simulação.

Deve-se notar também o reduzido tempo despendido pelos applets simuladores para a resolução dos sistemas de equações propostos. Mesmo com a solicitação de grande quantidade de pontos em determinado intervalo de simulação, o tempo despendido para resolução do problema em geral é de alguns segundos. Esses resultados demostram o bom desempenho apresentado pelos applets simuladores via Internet, e a eficiência da linguagem Java na solução de problemas matemáticos.

Um verificação detalhada dos valores obtidos para as variáveis simuladas, ao final do intervalo total de integração, mostra uma variação média de apenas 10^{-8} entre o valor obtido utilizando o menor número de pontos e a simulação realizada com o maior número de pontos. Por exemplo, na Tabela 4.12 são mostrados os valores finais médios obtidos pelo applet simulador SemiBatch v1.1, utilizando os parâmetros padrão, com o menor e o maior número de pontos (ver Tabela 4.2), bem como a variação observada.

Tabela 4.12 – Comparação entre os valores finais da simulação obtidos pelo applet SemiBatch

Variável	**Valor Final da Simulação**		**Variação**
Simulada	**menor nº de pontos**	**maior nº de pontos**	
C_A (kg/l)	0,801653197141361	0,8016531971481395	$6{,}8.10^{-12}$
T (°C)	322,28411297769674	322,28411297771896	$2{,}2.10^{-11}$
T_C (°C)	302,71766572325356	302,7176657232674	$1{,}4.10^{-11}$

Os valores obtidos para o tempo de resolução correspondem à versão final dos simuladores, implementados com prioridade 3 na tarefa de exibição dos gráficos de resultados. A primeira versão dos simuladores foi desenvolvida com a prioridade padrão, ou seja, prioridade 5. No entanto, utilizando prioridade com valor 5 para os simuladores que utilizavam o método Runge-Kutta, no qual a atualização gráfica dos resultados ocorre concomitante com a resolução do sistema de equações, a mesma só ocorria após o término dos cálculos. Desta forma, foram realizados testes alterando-se a prioridade da tarefa responsável pela simulação. O valor ideal obtido para a prioridade foi 3, valor no qual a atualização do gráfico acompanha o progresso do cálculo. A conseqüência da redução da prioridade da exibição gráfica ocasiona um aumento no tempo final de resolução. Para a alteração implementada nos applets simuladores desenvolvidos, esse aumento foi de aproximadamente 20%. Essa perda no desempenho do simulador é compensada claramente pelo efeito obtido na atualização do gráfico de resultados.

Um comportamento inesperado observado nas Figuras 4.11 (Simulador de um Reator Adiabático em Batelada) e 4.15 (Simulador de um Conversor de SO_2) sugere, para esses simuladores envolvidos, uma inversão na tendência das curvas de tempo de resolução em função do número de pontos solicitados em uma simulação. Este comportamento se manifestou com a utilização de poucos pontos de simulação (< 50 pontos) e cuja simulação decorria menos que 0,5 s. Novos experimentos foram realizados com os dois simuladores com o objetivo de propor uma explicação para esse

comportamento. Além de novos experimentos com a implementação padrão dos simuladores, os mesmos foram submetidos a implementação da funcionalidade de fornecimento dos tempos necessário para a geração de cada ponto proposto. A Figura 4.24 mostra os resultados obtidos para o Simulador de um Reator Adiabático em Batelada com e sem a implementação da funcionalidade de tempos individuais de geração dos pontos de simulação, respectivamente. A Figura 4.25 mostra os resultados para o Simulador de um Conversor de SO_2.

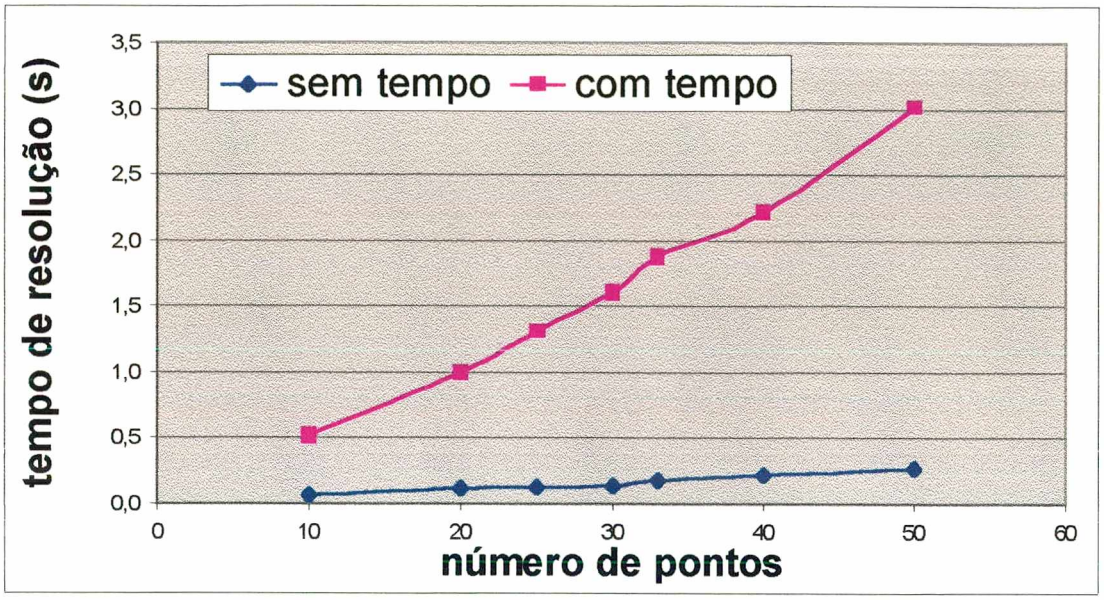


Figura 4.24 – Tempo de resolução em função do número de pontos para o Simulador de Reator Adiabático em Batelada [2]

[2] **OBSERVAÇÃO:**

sem tempo - sem rotina para impressão na saída padrão do sistema dos tempos gastos para gerar cada ponto de simulação.

com tempo - com rotina para impressão na saída padrão do sistema dos tempos gastos para gerar cada ponto de simulação.

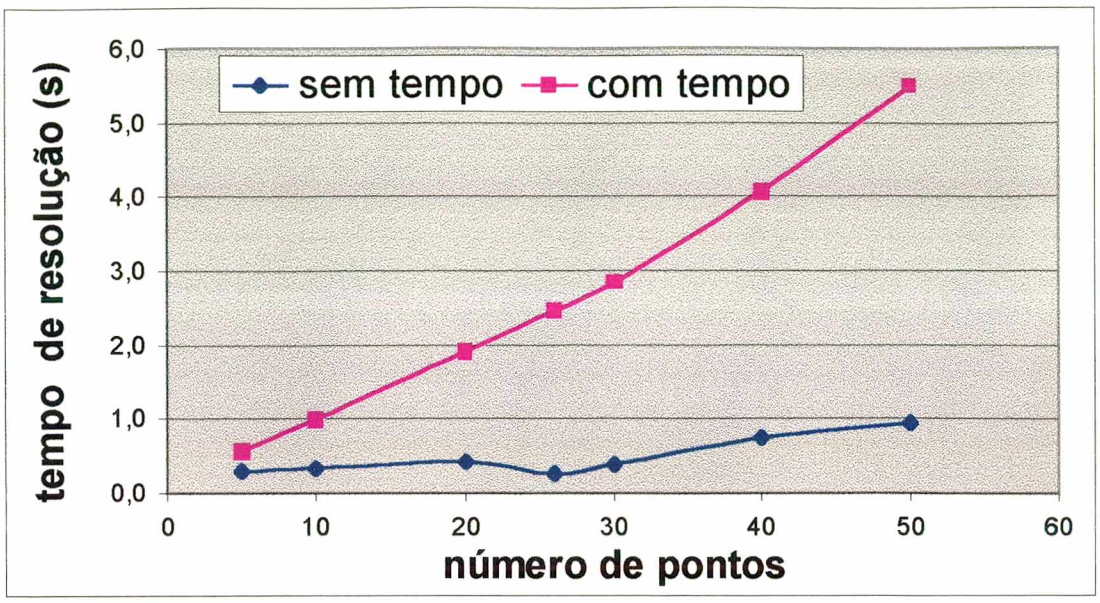


Figura 4.25 – Tempo de resolução em função do número de pontos para o Simulador de um Conversor de SO_2

Pelos resultados apresentados na Figura 4.24 nota-se a não reprodutibilidade do comportamento observado anteriormente no Simulador de um Reator Adiabático em Batelada, tanto utilizando a implementação padrão, quanto utilizando a implementação para obtenção do tempo despendido na geração de cada ponto de simulação. Observa-se também o aumento no tempo de resolução do simulador em função da implementação dessa rotina para obtenção dos tempos intermediários. Esse aumento também pode ser observado na Figura 4.25, para o Simulador de um Conversor de SO_2. Para este simulador, o comportamento de redução do tempo de resolução para um determinado aumento no número de pontos de simulação manifestou-se novamente na implementação padrão do simulador. Com a inclusão da rotina para obtenção dos tempos intermediários o comportamento foi o esperado. Para o caso específico desse simulador, acredita-se que a utilização de mais pontos de simulação (26) resulte em um menor tempo de resolução do que quando utilizando-se menos pontos (20) devido a convergência do problema para a obtenção dos resultados em pontos específicos. Como a rotina de Runge-Kutta utilizada neste simulador implementa monitoramento de erro e passo de integração auto-ajustável, é possível que a simulação com mais pontos necessite uma quantidade menor de ajustes no passo de integração para a obtenção de convergência em determinados pontos de simulação, ocasionando um tempo de resolução global inferior. Note-se que a diferença que ocasionou o estudo desse comportamento envolve uma variação de 10 pontos de simulação e de menos de 0,2 s no tempo de resolução. Acredita-se que esse comportamento é influenciado por outras variáveis e deve-se realizar novos estudos para uma avaliação concreta do mesmo.

4.2. APLICATIVOS DE PESQUISA

4.2.1. Sistema Cliente/Servidor de Aquisição e Controle à Distância

Com o objetivo de implementar o Sistema de Aquisição e Controle à Distância foram desenvolvidos dois programas em linguagem Java: o applet cliente Java e o aplicativo servidor Java. O sistema foi implementado em duas unidades experimentais: Unidade de Microgravimetria LABORE/UFSC e Unidade de Ultrafiltração LABSEM/UFSC.

4.2.1.1. Applet Cliente Java

O applet cliente Java desenvolvido tem o objetivo de permitir, a usuários utilizando um navegador Java-compatível, o acesso ao aplicativo servidor Java. Através do applet cliente o usuário pode acompanhar a aquisição de dados, alterar parâmetros de aquisição e/ou controle, informar-se sobre os experimentos programados e interagir diretamente com o operador do equipamento ou com outros usuários utilizando o sistema. A interface gráfica com o usuário desenvolvida para o applet cliente Java consiste de cinco seções principais:

Conexão – seção que permite ao usuário o envio de pedidos de conexão ou desconexão com o servidor. Para efetivar a conexão o usuário deve fornecer um nome e uma senha de acesso. O nome é utilizado para identificação do usuário, enquanto que a senha define o tipo de acesso que o mesmo terá ao servidor. O acesso visitante é realizado com a utilização da senha 'guest' e possibilita apenas a visualização dos dados sendo adquiridos do processo. O acesso administrativo é realizado utilizando-se uma senha específica, definida pelo operador do servidor. Essa modalidade de acesso permite, além da visualização de dados, a edição de parâmetros do processo.

Parâmetros – seção que permite a usuários com acesso administrativo a alteração de parâmetros do processo. A quantidade e o tipo dos parâmetros passíveis de alteração pela rede dependerão da unidade na qual o sistema estiver implementado. Ambas as unidades onde o sistema cliente/servidor foi implementado neste trabalho possibilitam a alteração do intervalo de aquisição, sendo que a Unidade de Ultrafiltração LABSEM permite ainda a alteração do set-point da válvula pneumática. A utilização desta seção por usuários com acesso visitante gerará uma mensagem de erro no quadro de status do applet cliente.

Cronograma – seção que permite o conhecimento de datas e horários nos quais a unidade estará em funcionamento.

Aquisição – seção que permite a visualização dos dados sendo adquiridos do processo, através de gráficos de linhas.

Canal de Comunicação – seção que possibilita a interação direta do usuário com o operador da unidade ou com outros usuários conectados ao sistema.

Além dessas cinco seções, o applet cliente reserva um espaço de sua interface gráfica para o Quadro de Status, local que possibilita a visualização de mensagens de retorno a ações realizadas pelo usuário. Esta seção é responsável ainda pela exibição do tempo de resposta do servidor, parâmetro atualizado a cada dez segundos.

As Figuras 4.24 a 4.28 mostram as várias seções que compõem a interface gráfica com o usuário desenvolvida para o applet cliente Java.

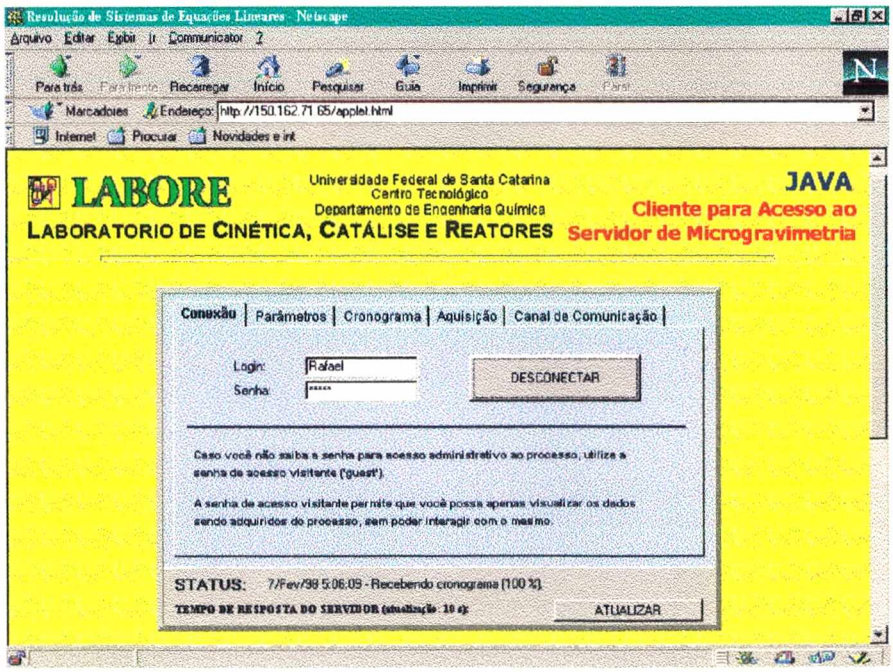


Figura 4.24 – Interface gráfica com o usuário desenvolvida para o applet cliente Java - seção Conexão

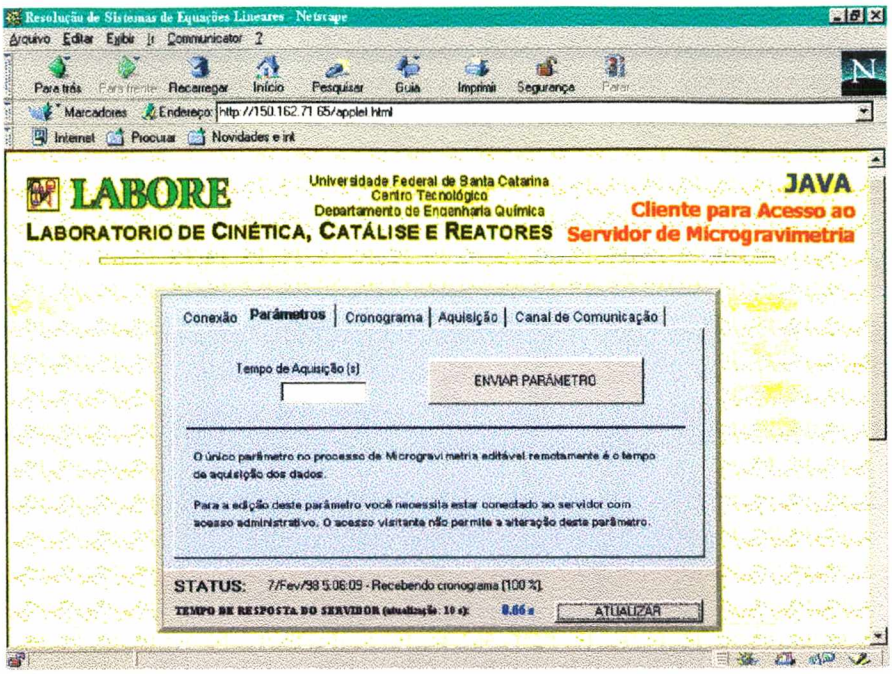


Figura 4.25 – Interface gráfica com o usuário desenvolvida para o applet cliente Java - seção Parâmetros

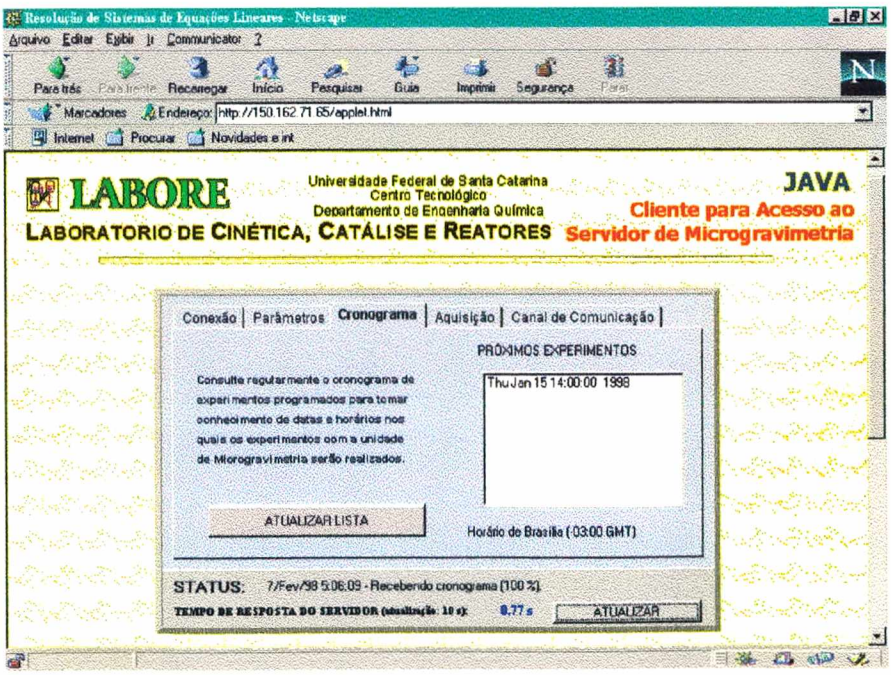


Figura 4.26 – Interface gráfica com o usuário desenvolvida para o applet cliente Java - seção Cronograma

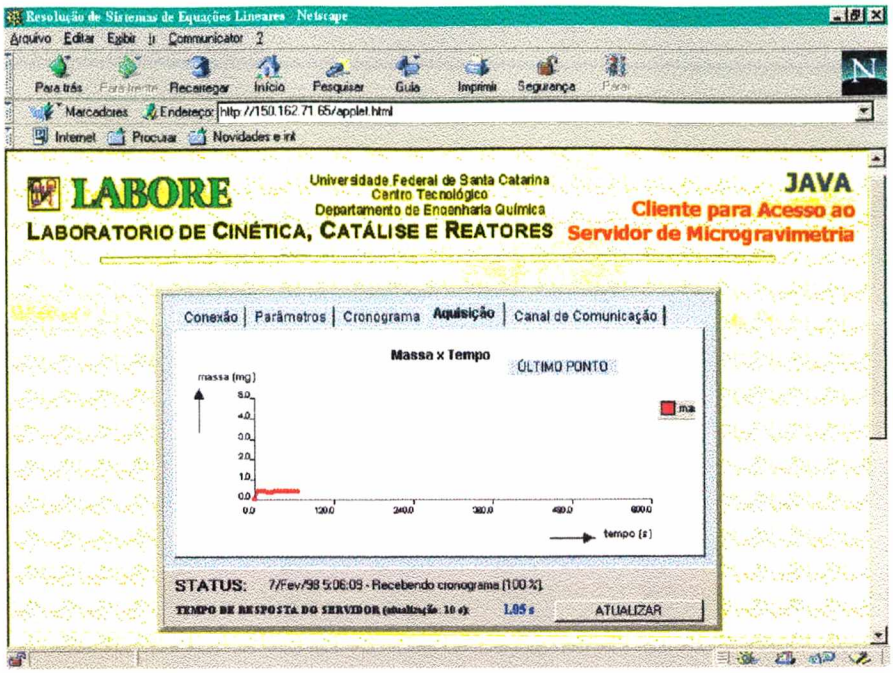


Figura 4.27 – Interface gráfica com o usuário desenvolvida para o applet cliente Java - seção Aquisição

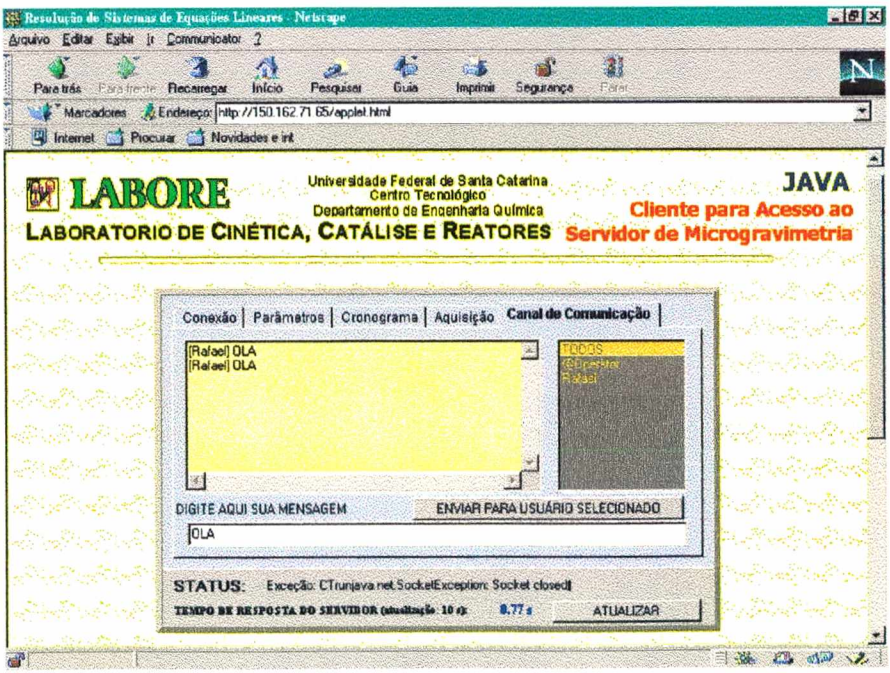


Figura 4.28 – Interface gráfica com o usuário desenvolvida para o applet cliente Java - seção Canal de Comunicação

O tempo de transferência médio do applet cliente instalado na Unidade de Microgravimetria LABORE/UFSC foi de 73,1 segundos (resposta ao ping = 165,1 ms). Já para o applet cliente instalado na Unidade de Ultrafiltração LABSEM/UFSC o tempo de transferência médio foi de 92,4 segundos (resposta ao ping = 163,8 ms).

A Tabela 4.13 relaciona as classes que necessitam ser transferidas pela rede Internet para execução dos applets clientes em cada unidade experimental.

Tabela 4.13 – Classes utilizadas pelo applet cliente em cada unidade experimental

CLASSE	**TAMANHO (Kbytes)**	
	Cliente LABORE	**Cliente LABSEM**
`ChatClienteThread`	5,50	5,50
`Cliente`	12,20	13,30
`ClienteThread`	4,62	4,66
Classes do pacote labore.graph (1)	7,30	10,80
Classes do pacote symantec.itools.awt (5)	24,50	24,50
Classes do pacote symantec.itools.awt.shape (3)	2,68	2,68
Classes do pacote symantec.itools.lang (1)	1,74	1,74
Classes do pacote symantec.itools.util (1)	2,15	2,15
Classes do pacote warhol.charts (6)	16,00	16,00
TOTAL – 20 classes	**76,69**	**81,33**

O tempo médio de resposta do servidor para o applet cliente instalado na Unidade de Microgravimetria LABORE/UFSC foi de 0,83 segundos (resposta ao ping = 158,7 ms). Já para o applet cliente instalado na Unidade de Ultrafiltração LABSEM/UFSC o tempo médio de resposta do servidor foi de 0,96 segundos (resposta ao ping = 170,6 ms).

4.2.1.2. Aplicativo Servidor Java

O aplicativo servidor Java desenvolvido tem o objetivo de implementar a interface para aquisição e/ou controle de unidades industriais ou de laboratório e fornecer suporte para que usuários localizados à distância da unidade possam ter acesso as informações geradas pelo aplicativo. Através do aplicativo servidor Java o operador da unidade pode ter o controle das conexões realizadas no servidor, agendar experimentos programados, interagir com a unidade através da aquisição e/ou do controle do

processo e utilizar um canal de comunicação para interagir diretamente com os usuários utilizando o sistema.

A interface gráfica com o usuário desenvolvida para o aplicativo servidor Java consiste de cinco seções principais:

Controle de Conexões Remotas – seção responsável pela administração das conexões remotas realizadas no servidor. Essa seção possibilita ao operador realizar a atualização da senha de acesso administrativo, o bloqueio de novas conexões, a visualização e desconexão de conexões ativas e a visualização do "status" do servidor. O "status" envolve o registro de todos as ações realizadas no servidor, como conexões, desconexões, envio de cronogramas e recebimento de parâmetros, bloqueio de conexões e alteração da senha administrativa. O controle de conexões remotas no servidor pode ser realizada através do bloqueio de novas conexões e da desconexão de conexões ativas. O bloqueio de conexões permite que solicitações de novas conexões sejam recusadas, enquanto que a desconexão permite desconectar determinadas (ou todas) conexões existentes no servidor.

Controle de Cronograma de Experimentos – seção que permite ao operador responsável pela unidade a administração de experimentos programados, incluindo a adição, alteração e exclusão de datas/horários. O objetivo do cronograma de experimentos é possibilitar aos usuários acessando o servidor remotamente o conhecimento prévio de datas e horários nos quais a unidade estará em operação.

Controle de Aquisição de Dados – seção que permite ao operador a definição dos parâmetros de aquisição de dados do processo. Esses parâmetros incluem a definição das variáveis de aquisição, a definição do intervalo de tempo no qual será realizada a média das aquisições dessas variáveis, entre outros. Essa seção possibilita ainda a visualização da data e horário de início e término do último experimento realizado ou em andamento, bem como sua duração. As opções para gravação dos dados de aquisição também são realizadas nessa seção, e incluem o intervalo de tempo de gravação e o nome e tipo (ASCII ou binário) do arquivo com os resultados da aquisição.

Canal de Comunicação – seção que permite ao operador interagir diretamente com os usuários conectados remotamente. Através dessa interface de comunicação, o operador pode esclarecer dúvidas sobre o aplicativo, sobre o processo e sobre os dados sendo adquiridos. A seção contém

uma caixa de status, onde as conexões, desconexões e mensagens públicas são registradas. Uma outra caixa relaciona os usuários conectados no servidor.

Resultados de Aquisição – para cada variável de aquisição é definida uma janela para exibição dos resultados de aquisição. O resultado da aquisição é apresentado através de gráficos de linhas, atualizados conforme os dados são adquiridos do processo. Além do gráfico de linhas com os resultados da aquisição, cada janela possui um status indicando o último ponto adquirido e adicionado ao gráfico.

As Figuras 4.29 a 4.34 mostram as várias seções que compõem o aplicativo servidor Java implementado na Unidade de Microgravimetria LABORE/UFSC.

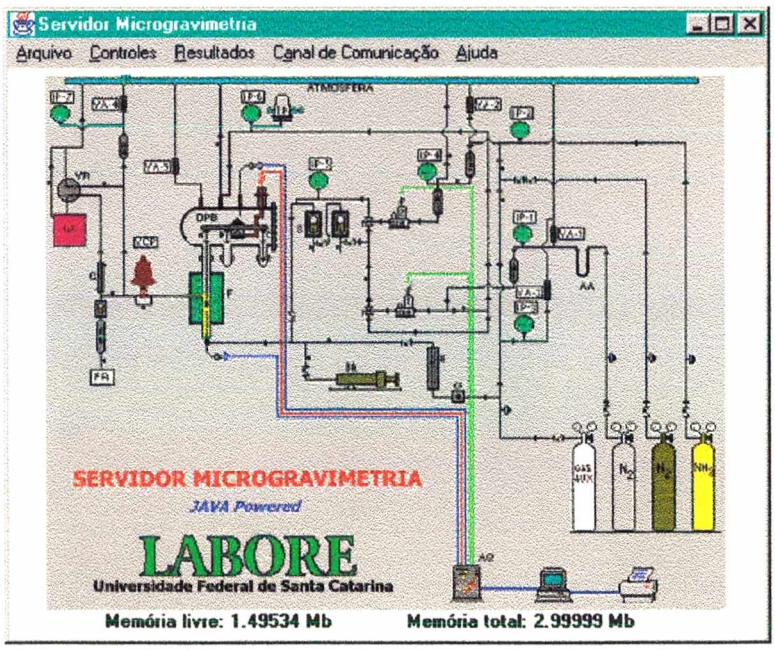


Figura 4.29 – Interface gráfica com o usuário desenvolvida para o aplicativo servidor Java implementado na Unidade de Microgravimetria LABORE/UFSC– Tela Inicial do Aplicativo

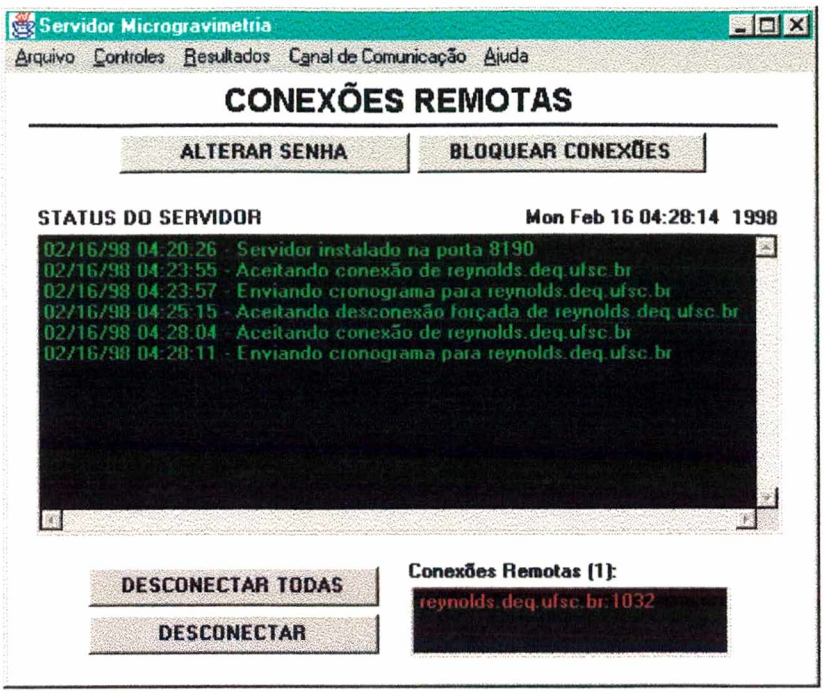


Figura 4.30 – Interface gráfica com o usuário desenvolvida para o aplicativo servidor Java implementado na Unidade de Microgravimetria LABORE/UFSC - seção Controle de Conexões

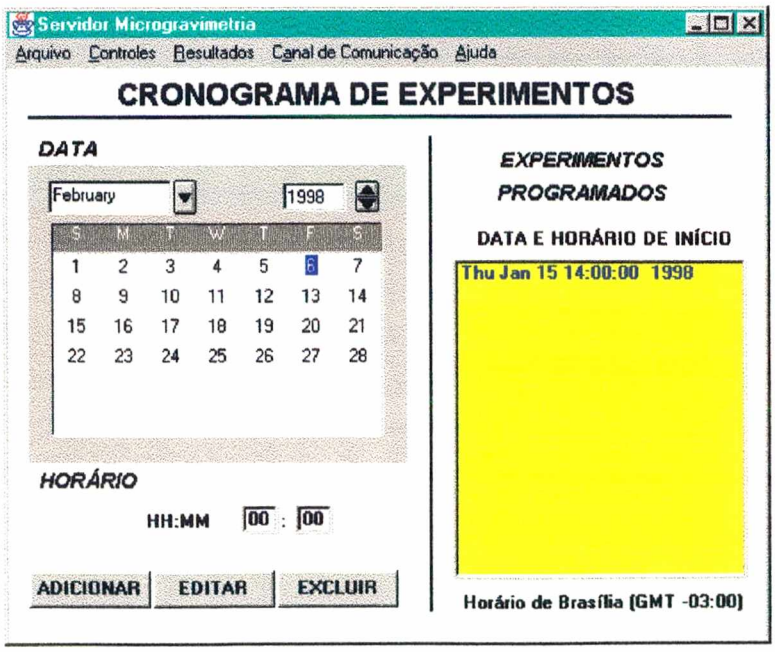


Figura 4.31 – Interface gráfica com o usuário desenvolvida para o aplicativo servidor Java implementado na Unidade de Microgravimetria LABORE/UFSC - seção Controle de Cronograma de Experimentos

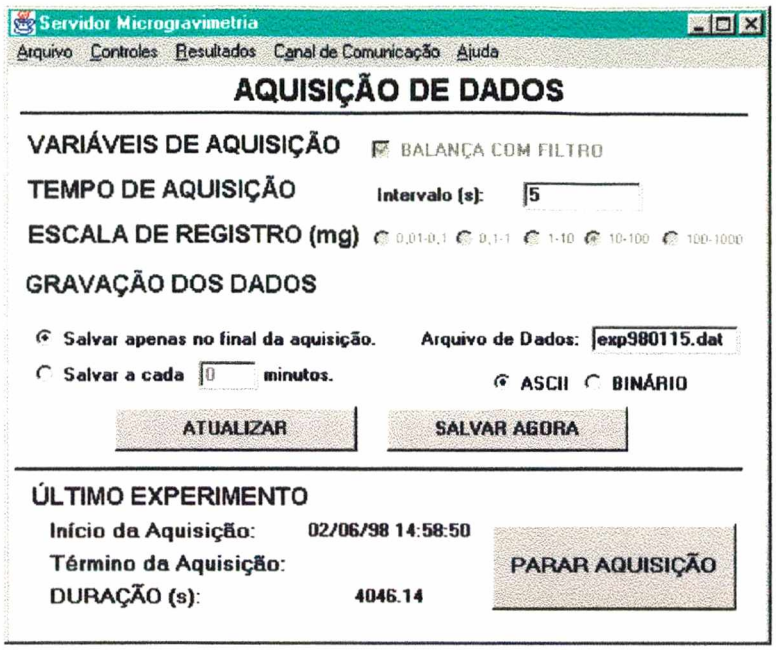


Figura 4.32 – Interface gráfica com o usuário desenvolvida para o aplicativo servidor Java implementado na Unidade de Microgravimetria LABORE/UFSC - seção Controle de Aquisição de Dados

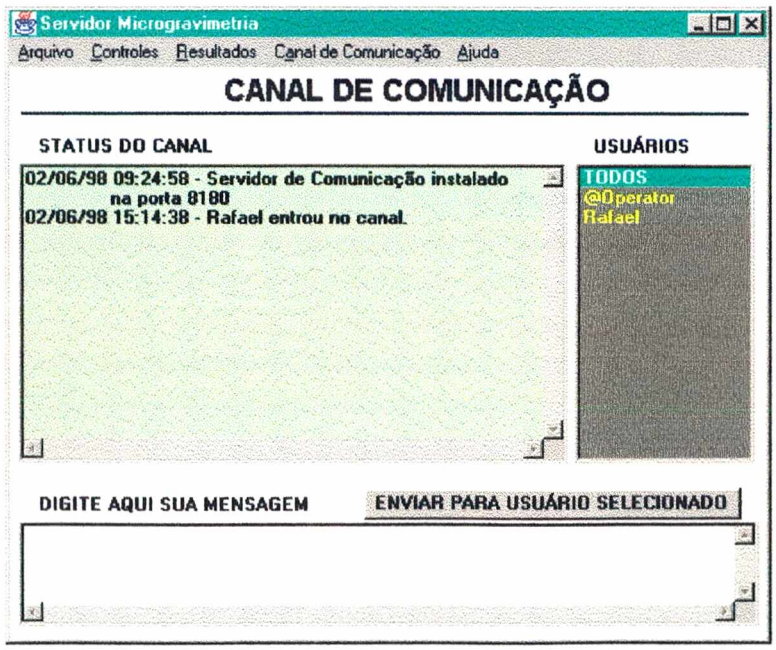


Figura 4.33 – Interface gráfica com o usuário desenvolvida para o aplicativo servidor Java implementado na Unidade de Microgravimetria LABORE/UFSC - seção Canal de Comunicação

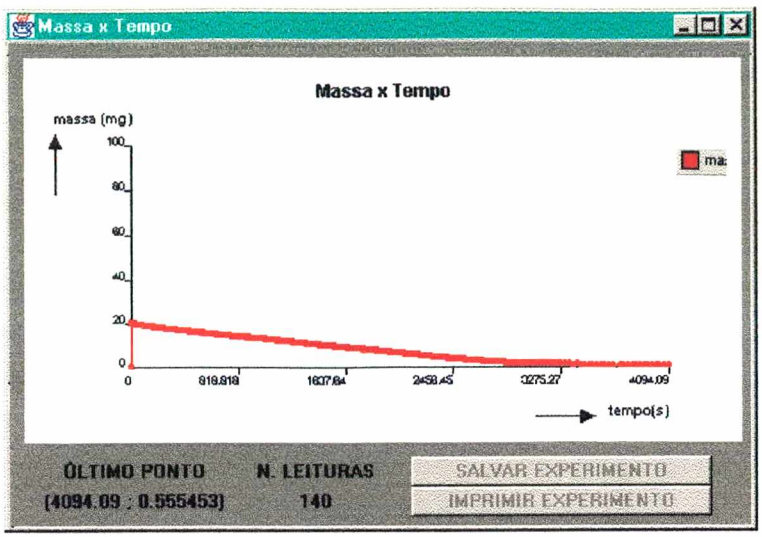


Figura 4.34 – Interface gráfica com o usuário desenvolvida para o applet cliente Java implementado na Unidade de Microgravimetria LABORE/UFSC - seção Resultados de Aquisição

A interface gráfica definida para o aplicativo servidor Java implementado na Unidade de Ultrafiltração LABSEM/UFSC é semelhante à interface apresentada para a Unidade de Microgravimetria LABORE/UFSC. As únicas exceções importantes dizem respeito às seções de Tela Inicial do Aplicativo e de Controle de Aquisição de Dados, seções estas mostradas na Figura 4.35 e 4.36, respectivamente.

Figura 4.35 – Interface gráfica com o usuário desenvolvida para o applet cliente Java implementado na Unidade de Ultrafiltração LABSEM/UFSC – Tela Inicial do Aplicativo

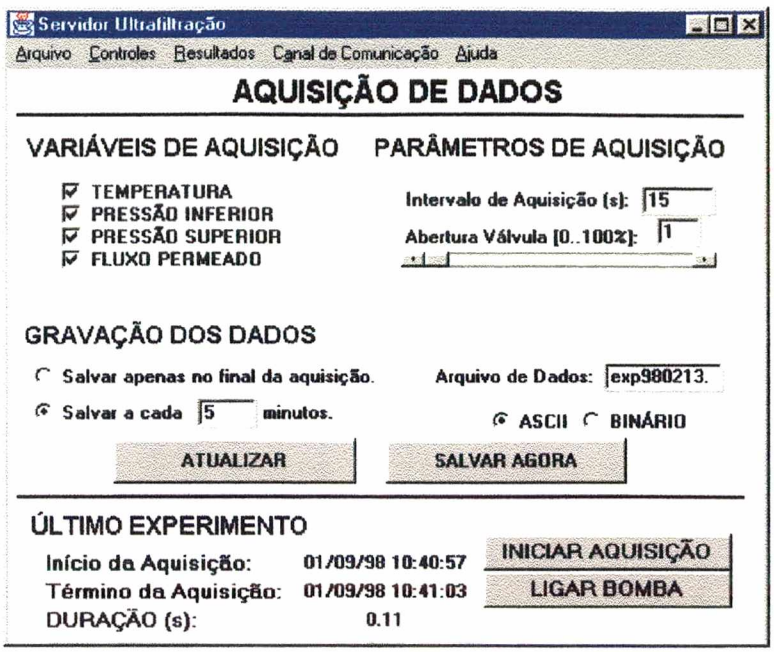


Figura 4.36 – Interface gráfica com o usuário desenvolvida para o applet cliente Java implementado na Unidade de Ultrafiltração LABSEM/UFSC - seção Controle de Aquisição de Dados

Com relação ao desvio entre os tempos de aquisição obtidos pelo aplicativo servidor Java e aqueles esperados idealmente em uma aplicação em tempo real, temos uma variação positiva média de 0,055 s na Unidade de Microgravimetria LABORE/UFSC e de 0,077 s na Unidade de Ultrafiltração LABSEM/UFSC. Esses valores não são cumulativos. Isto significa que se desejarmos obter os dados de aquisição utilizando o aplicativo servidor Java a cada 30 segundos, na verdade obteremos os dados com tempos de aquisição acrescidos em média por 0,07 segundos.

Há de se registrar o desempenho inferior do aplicativo servidor Java com relação a aplicações executáveis nativas da plataforma utilizada. Enquanto as aplicações executáveis, como o Microsoft Word, foram desenvolvidas especificamente para essa plataforma, o aplicativo servidor Java necessita ser interpretado antes de sua execução. Esse é o preço pago pela interoperabilidade e independência de plataforma suportada pela linguagem Java. O efeito da interpretação dos bytecodes no desempenho do aplicativo servidor Java é observado mais profundamente quando há a necessidade de atualização de sua interface gráfica, ou quando outras aplicações executáveis nativas estão sendo utilizadas no sistema. Observa-se, contudo, que o interpretador utilizado é relativamente "antigo" (1996), sendo que a utilização de versões mais recentes, ou compiladores "Just-In-Time" mais eficientes, possivelmente resultaria em um desempenho superior ao observado.

4.2.2. Implementação na Unidade de Microgravimetria LABORE/UFSC

Com o objetivo de avaliar o sistema em funcionamento na Unidade de Microgravimetria LABORE/UFSC foram realizados experimentos utilizando compostos embebidos em um pedaço de papel para determinação da taxa de evaporação dos mesmos.

Os resultados obtidos foram adimensionalizados para uma melhor análise e comparação. Definiu-se a massa residual adimensional como o valor obtido da divisão da massa de composto presente na amostra pela massa inicial, e o tempo adimensional como sendo o tempo decorrido dividido pelo tempo necessário para que a massa residual adimensional atingisse o valor de 5% da massa inicial, chamado de tempo de secagem. A Figura 4.37 mostra os resultados obtidos para os quatro compostos utilizados: álcool etílico, n-butilamina, acetona e hidróxido de amônio.

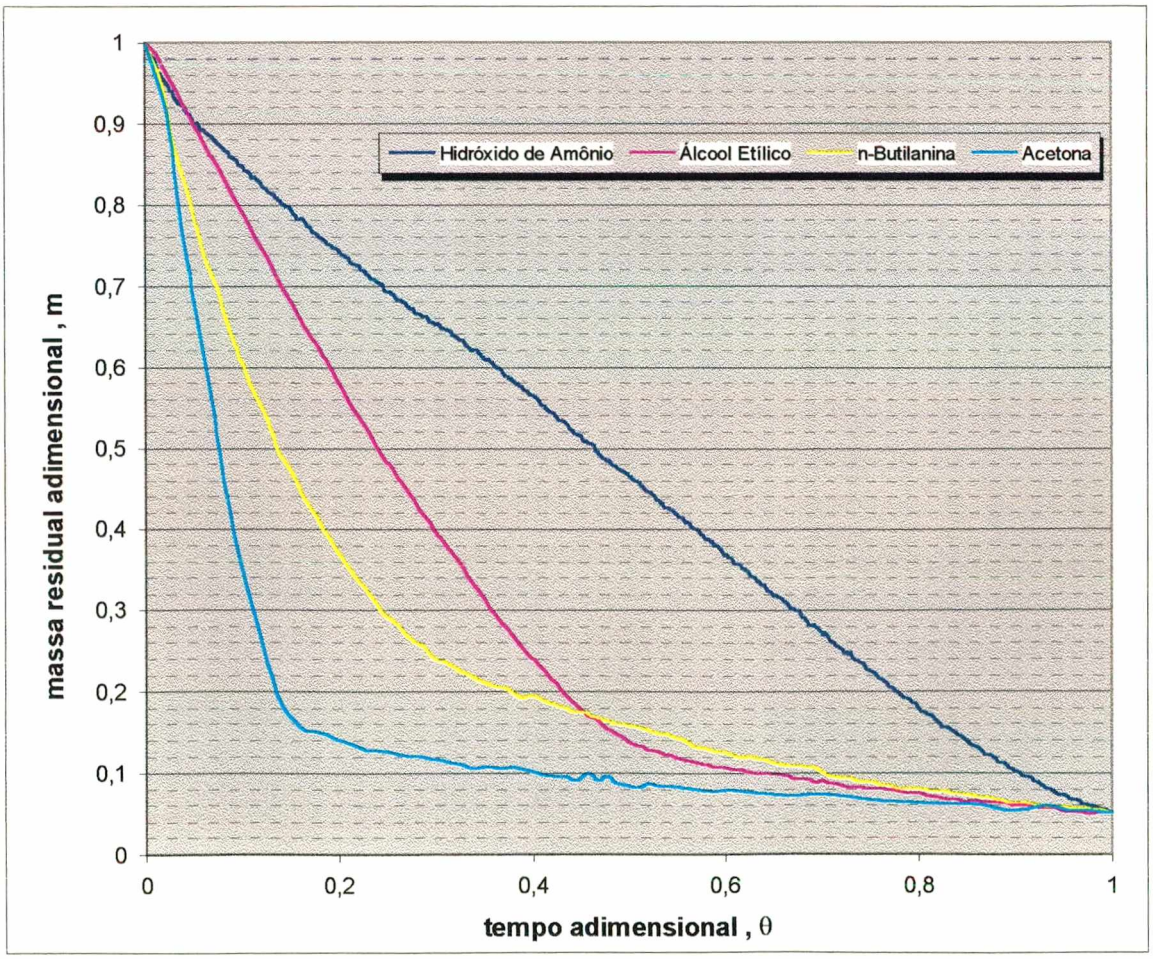


Figura 4.37 – Resultados obtidos para os experimentos de evaporação de compostos voláteis na Unidade de Microgravimetria LABORE/UFSC

A Tabela 4.14 mostra os resultados obtidos para as assíntotas inicial e final obtidas do Figura 4.37, para cada composto utilizado. As assíntotas inicial (primeiros 10 pontos) e final (últimos 20 pontos) foram associadas às taxas de evaporação e difusão, respectivamente. O coeficiente linear da assíntota final foi associado à massa de solvente ocluída durante a imersão.

Tabela 4.14 – Assíntotas inicial e final obtidas para os experimentos de evaporação na Unidade de Microgravimetria LABORE/UFSC

Composto	Assíntota inicial	R-quadrado	Assíntota final	R-quadrado
Álcool Etílico	m=-2,244*t+1,0121	0,9997	m=-0,1120*t+0,1604	0,9258
n-Butilamina	m=-4,503*t+1,0197	0,9921	m=-0,1336*t+0,1844	0,9692
Acetona	m=-7,1677*t+1,0571	0,9957	m=-0,0680*t+0,1176	0,9616
Hidróxido de Amônio	m=-2,6937*t+1,0032	0,9947	m=-0,4350*t+0,4848	0,9830

A Tabela 4.15 mostra os resultados obtidos com os experimentos realizados na Unidade de Microgravimetria LABORE/UFSC.

Tabela 4.15 – Resultados obtidos para os experimentos de evaporação na Unidade de Microgravimetria LABORE/UFSC

Composto	Álcool Etílico	n-Butilamina	Acetona	Hidróxido de Amônio
Peso molecular (g/gmol)	46,07	73,14	58,08	34,05
Temperatura ambiente (°C)	29,5	29,9	32,5	32,8
Massa inicial (mg)	16,93	8,53	13,39	20,10
Tempo de secagem (s)	885,12	630,10	460,00	3005,08
Desvio do tempo real (s)	0,0533	0,0544	0,0525	0,0579
Massa absorvida (mg)	0,16	0,18	0,12	0,48
Taxa de evaporação (mg/min)	2,24	4,50	7,17	2,69
Taxa de difusão (mg/min)	0,11	0,13	0,07	0,43

Com relação à massa absorvida, a seguinte seqüência de compostos foi obtida: hidróxido de amônio >> n-butilamina > álcool etílico > acetona. Com relação a taxa de evaporação, a seqüência obtida foi: acetona > n-butilamina > hidróxido de amônio > álcool etílico. Por último, com relação à taxa de difusão a seqüência obtida foi: hidróxido de amônio >> n-butilamina > álcool etílico > acetona.

4.2.3. Implementação na Unidade de Ultrafiltração Tangencial LABSEM/UFSC

Inicialmente, o sistema foi utilizado para a calibração dos sensores da unidade. O experimento de calibração foi realizado utilizando água destilada com temperatura 30-32°C, com a válvula pneumática totalmente aberta e variando-se a freqüência da bomba de recirculação entre 0 e 60 Hz, através da utilização do Conversor de Freqüência. Sensores analógicos foram instalados junto aos sensores digitais de temperatura e de pressão na entrada do módulo de filtração. Para o fluxo permeado, utilizou-se tanto o sensor analógico integrado ao sensor digital quanto medidas utilizando provetas graduadas e cronômetro digital. Iniciando o experimento com a freqüência da bomba em 0 Hz, foram aplicadas cargas de 15 Hz até atingir a freqüência máxima da bomba, 60 Hz.

A Tabela 4.16 mostra os resultados obtidos através de leituras diretas das variáveis de automatização da Unidade de Ultrafiltração LABSEM/UFSC durante o experimento de calibração.

Tabela 4.16 – Resultados de leituras diretas das variáveis de automatização da Unidade de Ultrafiltração LABSEM/UFSC durante o experimento de calibração

Freqüência (Hz)	**0**	**15**	**30**	**45**	**60**
Temperatura (°C)	30,0	30,0	30,0	30,5	32,0
Pressão na entrada do módulo de filtração (bar)	0,00	0,27	0,97	2,05	3,36
Fluxo permeado-sensor analógico (l/h)	0,0	0,0	9,0	31,0	-
Fluxo permeado-medidas diretas (l/h)	0,0	9,9	53,7	96,7	157,2

A Figura 4.38 mostra os resultados obtidos através do aplicativo servidor Java para as leituras dos sensores digitais realizadas durante o experimento de calibração.

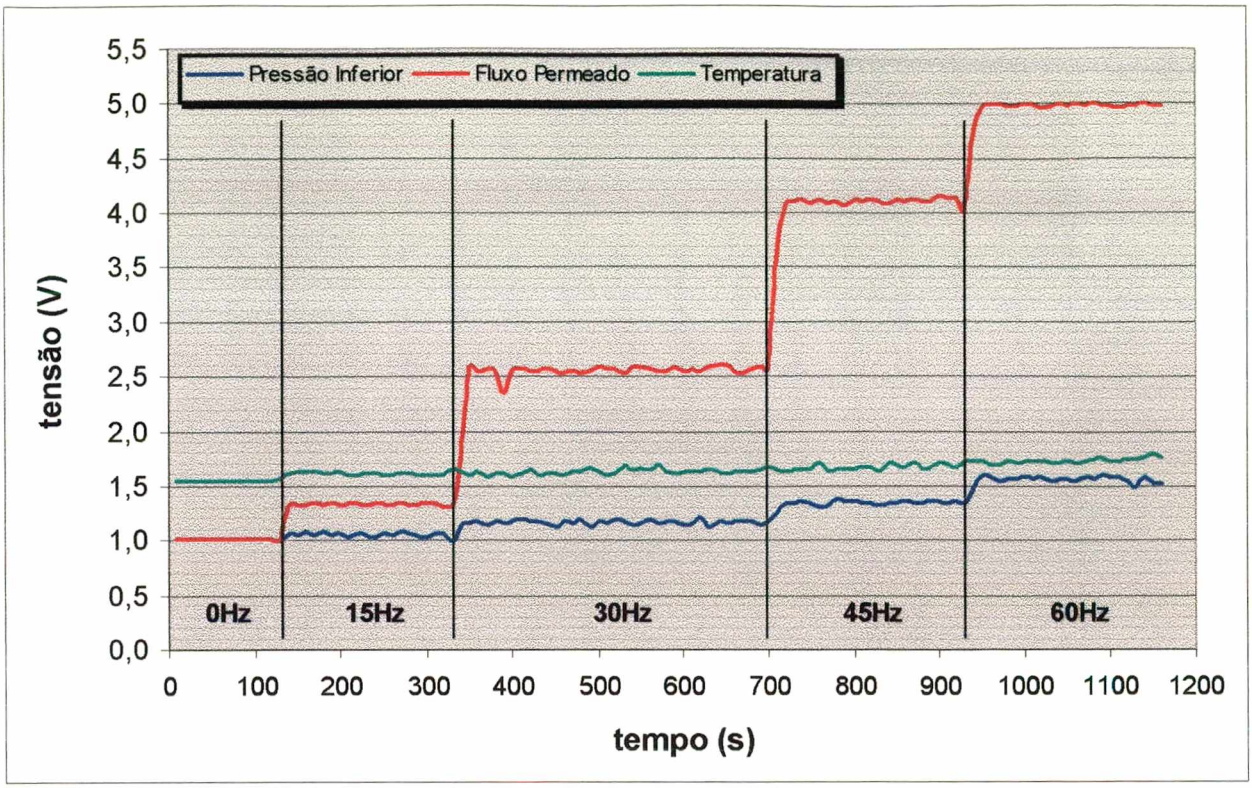


Figura 4.38 – Resultados obtidos para o experimento de calibração da Unidade de Ultrafiltração LABSEM/UFSC

Com os dados obtidos da Tabela 4.16 e da Figura 4.38, foram determinados as equações de calibração para os sensores de temperatura, de pressão e de fluxo permeado. A Tabela 4.17 descreve os sensores calibrados na unidade, bem como as equações de calibração e os valor do R-quadrado resultantes da linearização dos dados obtidos no experimento de calibração.

Tabela 4.17 – Resultados de calibração dos sensores de automatização da Unidade de Ultrafiltração LABSEM/UFSC

Sensor	Equação de calibração[3]	R-quadrado
Temperatura	T=18,689*V	0,9998
Pressão	P=6,0284*V-6,0225	0,9990
Fluxo Permeado (fluxímetro analógico)	F=13,947*V-26,753	0,9963
Fluxo Permeado (medidas diretas[4])	F=31,097*V-30,563	0,9961

[3] V = valor de leitura do sensor em volts.

[4] Medidas realizadas com o auxílio de provetas graduadas e cronômetro digital.

Posteriormente o sistema foi utilizado para realizar a aquisição de dados da unidade em operação. As Figuras 4.39 a 4.41 mostram os resultados obtidos para a unidade após a aplicação de duas cargas no sistema. Antes do início do experimento de operação, a freqüência da bomba foi fixada em 30 Hz e a abertura da válvula em 50%. A primeira carga aplicada corresponde à abertura da válvula para 100% (tempo de 90,08 s). A seguir aplicou-se a segundo carga, aumentando a freqüência da bomba para 40 Hz (tempo de 210,03 s).

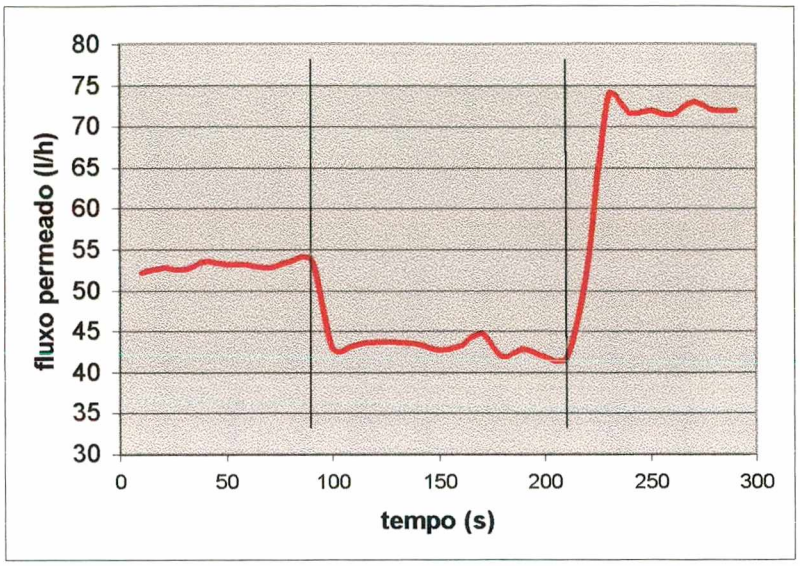


Figura 4.39 – Resultados obtidos para o experimento de operação da Unidade de Ultrafiltração LABSEM/UFSC – fluxo permeado

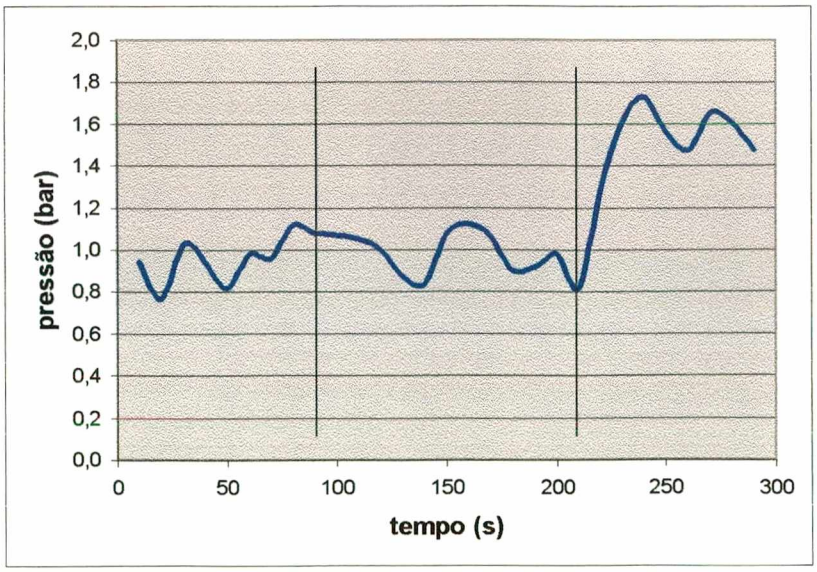


Figura 4.40 – Resultados obtidos para o experimento de operação da Unidade de Ultrafiltração LABSEM/UFSC – pressão na entrada do módulo de filtração

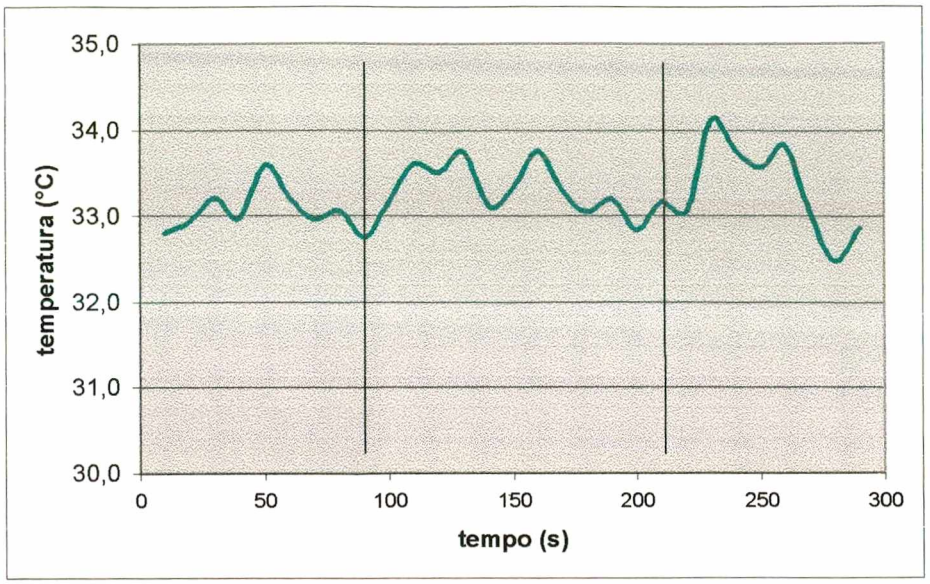


Figura 4.41 – Resultados obtidos para o experimento de operação da Unidade de Ultrafiltração LABSEM/UFSC – temperatura

Conforme Figuras apresentadas, observa-se que a aplicação da primeira carga ocasiona uma diminuição no fluxo permeado de 53 l/h para 43 l/h. A temperatura e a pressão na entrada do módulo de filtração permanecem inalterados. Com a aplicação da segunda carga, o fluxo de permeado sofre uma elevação de 43 l/h para 72 l/h, enquanto que a pressão na entrada do módulo de filtração passa de 0,9 bar para 1,5 bar. A temperatura permanece, em média, inalterada em 33,5°C.

Observa-se tanto no experimento de calibração quanto no experimento de operação o efeito causado pela utilização do Conversor de Freqüência. A utilização do Conversor de Freqüência gera uma série de harmônicos na corrente de alimentação, ocasionando oscilações indesejáveis nas leituras dos sensores.

Capítulo 5

CONCLUSÕES E SUGESTÕES

5.1. CONCLUSÕES

Como resultado final deste trabalho, pode-se afirmar que a utilização da linguagem Java para o desenvolvimento de applets e aplicativos de ensino e pesquisa em Engenharia Química proporciona condições ideais para a criação de sistemas eficientes, interativos, de fácil manutenção e atualização e acessíveis através da Internet. Esse acesso possibilita uma maior divulgação de tais sistemas entre usuários potenciais de programas para Engenharia. Além disto, implementa uma nova metodologia de distribuição de software, o qual pode potencialmente ser atualizado de uma única vez. Os usuários podem utilizar o sistema para atingir determinado objetivo, e quando for necessário, rapidamente incorporar novas capacidades ao sistema.

Na área específica de projetos auxiliados por computador, CAD, por exemplo, Java proporciona a oportunidade de vários engenheiros trabalharem no mesmo projeto simultaneamente, melhorando significativamente o processo de desenvolvimento. Pode-se ter acesso direto ao projeto atualizado, incluir novas informações e incorporar projetos feitos por terceiros.

Entre as ferramentas de desenvolvimento de programas Java testados, o pacote Symantec Visual Café Pro demonstrou ser uma boa escolha. Seu ambiente gráfico oferece ferramentas eficientes para o desenvolvimento rápido de programas Java. A tarefa de criação da interface gráfica é realizada visualmente, através da utilização de uma extensa (e extensível) biblioteca de componentes. Além de apresentar um conjunto de rápido desenvolvimento a nível de código, incluindo o navegador de classes e as ágeis procuras por expressão, a ferramenta contém um assistente para geração de código de tratamento de eventos e um depurador rápido e poderoso. A utilização da ferramenta Visual Café Pro possibilita a criação de programas Java com excelente interface gráfica e funcionalidade, de uma maneira rápida e eficiente.

O pacote matemático labore.math, projetado e desenvolvido neste trabalho, possibilita a rápida expansão do desenvolvimento de simuladores Web para Engenharia. A integração do pacote matemático é fácil e rápida. Após a definição do problema, basta criar-se uma classe derivada de uma das classes que compõem o pacote (`LUDecomposition`, `NRaphson`, `RungeKutta4` e `OrthogonalCollocation`) e integrá-la à interface gráfica no VisualCafe. A implementação do método matemático apropriado envolve poucas linhas de codificação, consistindo de no máximo seis etapas, envolvendo a declaração da modelagem matemática, a resolução do problema e a obtenção dos resultados.

O desempenho e o potencial da linguagem Java e do pacote labore.math podem ser observados nas avaliações realizadas em cinco applets simuladores de problemas de Engenharia de Reações Químicas. É digno de nota o pequeno tamanho dos applets, quando comparados com programas similares escritos em outras linguagens sendo que, em média, os applets simuladores não excederam 100 Kbytes. Quando os applets são transferidos pela rede, o tempo de sua transmissão, com a memória cache do navegador vazia, é de cerca de um minuto para o início de sua execução local. Esse tempo de transferência inicial dos applets simuladores é fortemente dependente da velocidade de conexão entre a máquina local e o host que armazena os applets.

A precisão dos resultados e o desempenho fornecidos pelas classes do pacote labore.math foram excelentes. Em geral, simulações completas com precisão da ordem de 10^{-8}, realizadas com os applets simuladores desenvolvidos, foram completadas em alguns segundos. O desempenho apresenta relação direta com o número de pontos que se deseja computar na simulação.

A linguagem Java pode manipular facilmente a arquitetura cliente/servidor, a comunicação de rede Internet, a multitarefa e a integração local com hardware especializado através de métodos nativos. Utilizando estas funcionalidades foram desenvolvidos neste trabalho sistemas do tipo cliente/servidor para a aquisição e controle à distância de processos. Os sistemas cliente/servidor Java possibilitam que usuários utilizando a Internet possam acessar um applet cliente, e utilizá-lo para se conectar com um aplicativo servidor Java, instalado no "host" de uma unidade química. Através do fornecimento de senhas para controle de níveis de acesso, o usuário pode visualizar a aquisição de dados online da unidade. Também pode interagir com o processo, enviando set-points e manipulando variáveis; pode obter informações sobre o cronograma de operação da unidade, e pode interagir diretamente com o operador ou com outro usuário através da utilização de um canal de comunicação.

Nas duas unidades experimentais nas quais o sistema cliente/servidor foi implementado, Unidade de Microgravimetria do LABORE/UFSC e Unidade de Ultrafiltração do LABSEM/UFSC, o tempo de resposta do servidor quando acessado remotamente foi inferior a um segundo! Esse tempo é influenciado pela situação do canal de conexão entre o usuário e o host da unidade. Para os testes realizados, a situação do canal de conexão pode ser considerada boa (resposta ao ping≈170ms). Mas, se o pedido de modificação de parâmetros vier de uma conexão ruim, esse tempo pode ser demasiado alto, impossibilitando a autorização de alteração de parâmetros da unidade remotamente. Dependendo da natureza do processo, a permissão para alteração de parâmetros da unidade só é possível se o tempo de resposta do servidor for pequeno. Situações imprevisíveis podem ser obtidas se o tempo de resposta do servidor for excessivamente alto. Por exemplo, em um processo em aquecimento, dependendo do tempo necessário para que um pedido de resfriamento seja recebido pelo servidor, o processo pode já ter mudado seu estado e estar resfriando.

Quanto a desvios com relação a aplicações de aquisição em tempo real, o aplicativo servidor apresentou um acréscimo médio no valor dos tempos de aquisição na ordem de 0,06 segundos. Como os resultados globais a serem extraídos dos processos aos quais o aplicativo servidor se destina a ser implementado não são influenciadas por essa pequena variação, o aplicativo servidor pode ser uma ferramenta eficiente tanto para a implantação em novas unidades de aquisição quanto para atualização de sistemas de aquisição e controle existentes.

O aplicativo servidor, assim como todos os programas descritos nesse trabalho, foi desenvolvido com base em uma estrutura de classes especializadas e bem definidas, tornando a implementação de sistemas similares em outras unidades experimentais ou industriais uma tarefa fácil e rápida.

A utilização do sistema cliente/servidor em unidades distantes ou localizadas em ambientes perigosos possibilita o acompanhamento e o controle desses processos sem a necessidade de presença física no local.

Os experimentos de calibração e de operação da Unidade de Ultrafiltração demonstraram a necessidade de substituição do sistema atual para variação de velocidade da bomba de recirculação. A utilização do sistema motor trifásico de corrente alternada e Conversor de Freqüência provoca oscilações indesejáveis nas leituras dos sensores, ocasionadas pela geração de harmônicos na corrente de alimentação.

5.2. SUGESTÕES

As sugestões para trabalhos futuros são abordadas em dois níveis: sugestões para complementação deste trabalho e sugestões para desenvolvimento de novos produtos Java.

Como sugestões para trabalhos que objetivem complementar os dados apresentados aqui, pode-se relacionar:

– Desenvolvimento de exercícios didáticos envolvendo os applets simuladores e os applets clientes, objetivando a aplicação prática desses programas no ensino de Engenharia Química, com avaliação da influência dessa metodologia no aprendizado das disciplinas relacionadas.

– Estudo da utilização dos applets simuladores e dos applets clientes em outros navegadores, como o Microsoft Internet Explorer, em outras plataformas, e em outros hosts utilizando conexões diferenciadas. Este estudo forneceria uma melhor avaliação desses programas Java acessados pela rede.

– Substituição do sistema de motor trifásico de corrente alternada e Conversor de Freqüência, utilizado para a variação de velocidade da bomba de recirculação, por um sistema composto por um motor trifásico de corrente contínua e um variador de velocidade. Esta substituição se faz necessária em virtude das oscilações indesejáveis causados pelo sistema atual.

Como um projeto de maior porte pode-se projetar o desenvolvimento de um serviço Web no qual os usuários da rede poderiam ter acesso a um Curso Completo de Engenharia Química totalmente dinâmico e interativo, envolvendo:

– Desenvolvimento de classes matemáticas implementando métodos numéricos para resolução de sistemas de equações diferenciais parciais, tais como o método da colocação ortogonal para EDPs, o método dos elementos finitos, o método dos volumes finitos, entre outros. Essas classes viriam complementar o pacote matemático labore.math. Paralelamente, applets simuladores para a aplicação desses métodos podem ser desenvolvidos.

– Desenvolvimento de applets simuladores aplicados a outras áreas da Engenharia Química, como Termodinâmica, Fenômenos de Transferência de Movimento e de Calor, entre outros.

– Desenvolvimento de "solvers" integrados, oferecendo um serviço de resolução de sistemas de equações de qualquer tipo. Através da utilização de uma classe que implemente um interpretador de

equações ("parser"), pode-se desenvolver uma interface na qual o usuário possa digitar suas próprias equações e definir as condições iniciais e de contorno de seu problema com facilidade.

– Desenvolvimento de tabelas virtuais de propriedades químicas e físicas de compostos e materiais, incluindo inclusive uma tabela periódica virtual de elementos. Juntamente com a linguagem Java pode-se utilizar o suporte a banco de dados JDBC para o armazenamento de grandes quantidades de informação.

– Implementação do sistema cliente/servidor de aquisição e controle à distância em outras unidades experimentais, possibilitando a interação direta de usuários remotos com as mais variadas áreas de pesquisa em desenvolvimento na universidade. Com isso há a possibilidade de estreitar ainda mais os laços entre a universidade e a comunidade, além da divulgação dos estudos realizados pela universidade para uma quantidade crescente de usuários.

BIBLIOGRAFIA

BASTA, N. *Chemical Engineering*, Vol. 103, N°3, **1996**, 149-152

BASTA, N. *Chemical Engineering*, Vol. 104, N°4, **1997**, 155-160

BASTA, N. e KIM, I. *Chemical Engineering*, Vol. 103, N°7, **1996**, 40-43

BASTA, N. *Chemical Engineering*, Vol. 103, N°2 , **1996**, 131-133

BHATTACHARJEE, S. "Unit Deamon", **1996**
http://www.thermal.sdsu.edu/java_applets/badRadiation/index.html
(06 Fev. 1998)

CANCELIER, A.; FOLETTO, E.; OGEDA, R. H. "Método de Runge-Kutta", Notas da disciplina ENQ3117-Métodos Numéricos Aplicados à Engenharia Química, **1996**

CHOWDHURY, J. *Chemical Engineering*, Vol. 102, N°3, **1995**, 30-35

CHOWDHURY, J. *Chemical Engineering*, Vol. 104, N°8, **1997**, 70-77

CHOWDHURY, J. *Chemical Engineering*, Vol. 104, N°9, **1997**, 54

CONNOLY, C. "Dynamic Simulation of a 2-Link Arm", **1996**
http://www.ai.sri.com/~connoly/robotics/dynsim.html.
(06 Fev. 1998)

Creative Engineering Software "Creative Engineering Software Solutions – StmP Steam and Water Properties, Steam Tables, Thermodynam", **1997**
http://www.eng-solutions.com/jcalc.htm
(06 Fev. 1998)

DEITZ, D. *Mechanical Engineering*, Vol. 118, N°4, **1996**, 68-72

DEITZ, D. *Mechanical Engineering*, Vol. 119, N°1, **1997**, 66-69

Developer.com. "developer.com – Directories". **1998**
http://www.developer.com/directories/pages/dir.java.educational.engineering.html
(16 Fev. 1998)

Developer.com. "developer.com – Directories". **1998**
http://www.developer.com/directories/pages/dir.java.educational.chemistry.html
(16 Fev. 1998)

DEVENPORT, W. J. "Ideal Flow Machine – The Applet", **1996**
http://aoe.vt.edu/aoe5104/ifm/ifm.html
(06 Fev. 1998)

FINLAYSON, B. A. *The Method of Weighted Residuals and Variational Principles with Application in Fluid Mechanics.* Academic Press, **1972**

FOGLER, H. S. *Elements of Chemical Reaction Engineering,* 2nd ed. Prentice-Hall, **1992**

FOUHY, K.; BASTA, N.; CHOWDHURY, J.; MATIC, J. *Chemical Engineering,* Vol. 104, N°8, **1997,** 137-142

GEROLD, J. S. *Control Engineering,* Vol. 44, N°4, **1997,** 64-78

GLASSCOCK, D. A . ; HALE, J. C.; SAMDAMI, S. *Chemical Engineering,* Vol. 101, N°11, **1994,** 82-89

GRAYCE, C. J. "The Second Law". **1996**
http://www.chem.uci.edu/instruction/applets/bounce.html
(06 Fev. 1998)

JULIANO, A. M. M. e PETRUS, J. C. C. *Recuperação por ultrafiltração das proteínas do soro para produção de queijos.* Ver. Lat. Cand. Tostes Vol. 42, N°306, **1987.**

KAZMER, D. "Injection Molding Cost Estimator". **1996**
http://www.ecs.umass.edu/mie/faculty/kazmer/imcost/imcost.html
(06 Fev. 1998)

LEMAY, L. e PERKINS, C. L. *Teach Yourself Java in 21 days,* 1s ed. Sams.net, **1996.**

MOROS, R. "Virtuelles Praktikum Technische Chemie | Versuch: VerweilZeite | ITC-Leipzig". **1996**
http://techni.tachemie.uni-leipzig.de/~vwz/index_e.html
(06 Fev 1997)

OGEDA, R. H. e PORTO, L. M. *Linguagem Java,* Porta Digital Internet, **1996**

OH, D. "The Java Virtual Wind Tunnel", **1997**
http://raphael.mit.edu/Java
(06 Fev. 1998)

Porta Digital Intenet, "Ferramenta Ping", **1996**
http://www.portadig.com.br/utilitarios/ping.html
(06 Fev. 1997)

PORTO, L. M; PETRUS, J. C. C.; JULIANO, A. M. M.; BOLZAN, A. *Montagem e Automatização de uma Unidade de Ultrafiltração para Produtos Alimentícios.* Projeto FINEP, Universidade Federal de Santa Catarina, **1994.**

PRESS, W. H.; FLANNERY, B. P.; TEUKOLSKY, S. A.; VETTRELING, W. T. *Numerical Recipes in C – The Art of Cientific Computing.* Cambridge University Press, **1988**

SHANK, P. e KIM, I. *Chemical Engineering,* Vol. 103, N°7, **1996,** 137-138

STEYN, M. M. DE V.; ALEXANDER, P. M.; RÖHM, D. *Computers Education,* Vol. 27, N°2, **1996,** 95-101

Symantec Corporation. *Symantec Visual Café Installation and Support*, Manual Online, **1996**

VAN RIENEN, I. "DigSim", **1997**
http://www.lookup.com/Homepages/96457/home.html
(06 Fev. 1998)

VANHELSUWÉ, I.; PHILIPS, G.; HSU, K.; SANKAR, E.; RIES, T.; RAHALY, J.; ZUKOSKI, J. *Mastering™ Java.* SYBEX, **1996**

VEDOVA, M. D. *Estudos de Adsorção e Dessorção de Piridina e n-Butilamina pelo Método Microgravimétrico.* Dissertação de Mestrado, *Universidade Federal de Santa Catarina*, **1996**

VILLADSEN, J. e MICHELSEN, M. L. *Solution of Differential Equations Models by Polinomial Approximation.* Prentice-Hall, **1978**

WIESER, B. "TBS-EOS Calculator", **1997**
http://www.octavian.com/tbs.html
(06 Fev. 1998)

WINTER, P. e SHELLEY, S. *Chemical Engineering*, Vol. 103, N°11, **1996**, 139

YOUNG, J. S. e SZOLLAR, S. "The Incredible Anneal-O-Matic", **1997**
http://www.cad.eecs.berkeley.edu/~jimy/classes/ee244/hw2/index.html
(06 Fev. 1998)

ZANETTI, R. J. *Chemical Engineering*, Vol. 103, N°11, **1996**, 46

ZANETTI, R. J. *Chemical Engineering*, Vol. 104, N°8, **1997**, 141

ZANETTI, R. J. *Chemical Engineering*, Vol. 104, N°9, **1997**, 195

APÊNDICE A - PROGRAMAÇÃO ORIENTADA A OBJETOS EM JAVA

A Programação Orientada a Objetos (POO) é uma das maiores novidades em termos de programação dos últimos anos. Usando programação orientada a objetos, os programas como um todo são construídos por diferentes componentes (objetos), os quais têm uma função específica no programa e podem se comunicar uns com os outros através de uma forma pré-definida.

A idéia da Programação Orientada a Objetos é que, assim como no mundo real, os objetos são compostos por muitos tipos de outros objetos menores. Desta forma, em um programa orientado a objetos teremos vários componentes menores agrupados para formar o sistema global. E cada um desses componentes é um objeto real, totalmente diferenciado dos demais.

Na programação orientada a objetos, cada objeto em um sistema é responsável por si próprio. Desta forma, cada objeto é responsável por alterar seu estado e executar ações específicas consigo relacionadas. Na verdade, cada objeto é geralmente composto por "variáveis", as quais definem seu estado e as características que o diferenciam de outros objetos semelhantes e por "métodos", os quais definem as ações realizadas pelo objeto.

Em Java, tudo que você fizer deve ser implementado via orientação a objetos. Para a utilização de determinado componente em seu sistema, você deve seguir dois passos distintos: declarar uma "classe" para o componente, e criar instâncias dessa classe, ou "objetos".

A "classe" é a representação abstrata de determinado componente, onde declara-se a estrutura do componente com relação às variáveis e métodos. A partir do momento em que você tenha definido a estrutura de um componente, através da declaração de uma classe que o representante, você pode criar quantas instâncias daquele componente forem necessárias. Então, você pode definir características diferentes para cada componente, simplesmente definindo valores diferentes para suas variáveis.

O "objeto" em si, ou "instância de classe", é a representação concreta de um determinado componente no sistema. Através de uma instância, pode-se manipular suas variáveis ou seus métodos. Em um sistema podemos ter quantas instâncias desejarmos de determinado componente, sendo que cada uma dessas instâncias pode ter estados e características diferentes.

Classes são utilizadas para definir novos tipos de dados em uma linguagem orientada a objetos. A declaração de uma classe em Java é realizada usando-se a sintaxe `class` seguida do nome da classe. Após a declaração da classe é feita a implementação da classe, especificando suas variáveis e métodos. Em Java os métodos são todos declarados e implementados dentro do próprio corpo da classe. Geralmente usa-se algum modificador precedendo a declaração da classe, para definir características especiais.

```
<modificador> class nome
{
  variáveis;
  ...
  métodos;
  ...
}
```

Os modificadores são utilizados para determinar o tipo de acesso que as variáveis e métodos terão em relação ao sistema como um todo.

Variáveis de instância são declaradas naturalmente dentro do bloco de uma classe, como uma variável local, mas com a possibilidade de incluir modificadores para controle de acesso com relação a outras classes no sistema. Já as variáveis de classe, que possuem um único valor para todas as instâncias da classe, além dos modificadores de acesso, possuem o modificador `static`.

Os métodos em Java parecem com funções em outras linguagens e se comportam como tal, mas são definidos dentro de uma classe. Os métodos podem ser usados para afetar um simples objeto, como também para realizar a comunicação com outros objetos. Uma classe ou objeto pode chamar métodos em outra classe ou objeto para lhe comunicar alterações em si ou verificar se aquele objeto foi alterado.

Assim como as variáveis, os métodos também podem ser de instância ou de classe. Métodos de instância são aqueles que distribuem uma cópia para cada instância da classe, e que possibilitam afetar as instâncias individualmente. Já os métodos de classe, definidos com a utilização do modificador

`static`, são utilizados para providenciar uma funcionalidade geral para todos os objetos da classe, e para a classe em si, sem afetar qualquer objeto específico.

A declaração de métodos consiste do tipo de retorno do método, o nome do método, a declaração dos argumentos de entrada e/ou saída do método e o bloco de declarações (corpo) do método em si. O tipo de retorno do método corresponde a um tipo de dado que será retornado pelo método quando o mesmo for executado. Este tipo de retorno pode ser qualquer tipo primitivo, qualquer tipo definido por uma classe, ou um valor vazio (`void`). Antes da declaração do método pode-se utilizar modificadores, principalmente para identificar o tipo de acesso que o mesmo oferecerá ao sistema.

```
<modificador> tipo-retorno nome(<arg1, arg2, ...>)

{

  declarações;

  ...

}
```

Os argumentos de um método são declarados entre os parênteses que seguem o nome do método. Os parênteses vazios indicam que o método não possui nenhum argumento. Os argumentos são declarados com um tipo e um nome e são separados por vírgulas. Você pode usar esses argumentos como variáveis locais em qualquer lugar dentro do método onde os mesmos estão declarados.

Um dos métodos definidos na declaração de uma classe merece atenção especial. Trata-se do "construtor" da classe. O construtor é o primeiro método chamado logo após ser criada uma nova instância de uma classe. No construtor da classe é determinado como o objeto é inicializado quando criado. Geralmente o processo de criação de um objeto é composto por três etapas: o programador declara o objeto e faz alocação de memória para o mesmo, através da utilização do operador new; as variáveis do objeto são inicializadas automaticamente dependendo de seu tipo (int=0; objetos=null; boolean=false, char='/0'); é realizada a chamada ao construtor da classe.

O construtor é um método que não possui tipo de retorno e tem o mesmo nome da classe. As principais funções do construtor são a alocação de memória para objetos auxiliares da classe, definição de valores iniciais para as variáveis, chamada de métodos baseados nessas variáveis ou de métodos em outros objetos, e cálculo de propriedades iniciais.

Lembre-se de que a linguagem Java não possui destrutores para desalocar memória, como em C++. Java possui um método chamado "finalizador", que pode ser usado para "limpar" um objeto antes do mesmo ser "garbage-collected" e de sua memória ser requisitada. O finalizador é chamado automaticamente quando todas as referências ao objeto são removidas, ou pode ser chamado a qualquer momento pelo programador. No entanto, a chamada direta ao método finalizador não apaga o objeto da memória. O objeto só será marcado para remoção quando todas as referências ao objeto forem removidas. O método finalizador é utilizado para melhorar a remoção de um objeto, através da eliminação de referências a outros objetos, etc.

Como vimos anteriormente, para termos acesso às variáveis e aos métodos declarados em uma classe, devemos geralmente criar uma instância desta classe e então referenciar às variáveis e métodos desejados. A criação de uma instância de determinada classe é feita utilizando-se o operador `new`, o qual é responsável pela alocação de memória para o objeto, juntamente com o construtor da classe. O construtor é chamado logo após a utilização do operador `new`, e pode ser utilizado para inicializar o objeto. O acesso às variáveis e aos métodos de determinada classe ou objeto é realizado usando o operador . (ponto).

Uma outra característica importante da Programação Orientada a Objetos é o polimorfismo, que se constitui em declarar vários métodos com o mesmo nome, mas com argumentos diferentes. A diferenciação nos argumentos pode ser quanto ao número ou ao tipo dos mesmos.

A idéia básica da declaração de classes em uma linguagem orientada a objetos é a de realizar uma verdadeira classificação. Deve-se pensar na implementação de um hierarquia de classes, indo desde uma estruturação mais geral, também chamada "super-classe", até a estruturação mais especializada, ou "sub-classe". A utilização de hierarquia de classes torna seu programa mais flexível, modular e reutilizável. Cria-se assim uma hierarquia, partindo-se de uma estrutura mais abrangente até uma estrutura mais particular. A construção da hierarquia, para definir uma classe como sendo derivada de outra classe mais geral, é realizada com a utilização da palavra reservada `extends` mais o nome da super-classe na declaração da sub-classe.

A grande vantagem da hierarquia de classes é a possibilidade de ter-se compartilhamento de variáveis e métodos de uma super-classe com uma sub-classe. Chama-se a isto "herança". Isto significa que uma sub-classe "herda" variáveis e métodos de sua super-classe, e pode utilizá-los como se os mesmos tivessem sido declarados para ela própria. Assim, conforme vai-se especializando uma classe através

de uma hierarquia, ela vai se tornando uma combinação de todas as características das classes acima dela na hierarquia. A linguagem Java também está organizada em uma hierarquia de classes, formando a interface de programação (API). Muita da funcionalidade que você pretende adicionar ao seu programa Java já está implementada nessa biblioteca de classes, através de uma série de classes específicas. Para a maioria dos casos não é preciso declarar classes novas, a não ser que se queira implementar um funcionalidade inexistente na interface de programação. Na biblioteca de classes da interface de programação Java, a classe que se encontra no topo da hierarquia é a classe `Object`, uma classe geral para objetos. A não ser que uma super-classe seja declarada, a herança é direta de `Object`.

Aqui verifica-se uma das diferenças observadas entre C++ e Java. A primeira permite herança múltipla, ou seja, "herdar" variáveis e métodos de mais de uma classe, enquanto que em Java só existe a herança simples. Java implementa herança múltipla através da utilização de interfaces, descritas mais adiante.

Outra característica da orientação a objetos são os "métodos sobrescritos". Com a utilização de métodos sobrescritos, pode-se definir métodos semelhantes para várias classes em uma hierarquia de classes. A utilização de métodos sobrescritos nos permite redefinir totalmente um método definido em uma super-classe, ou simplesmente acrescentar nova funcionalidade ao método, especializando-o a cada nível da hierarquia de classes. Métodos sobrescritos utilizam o mesma declaração tanto na super-classe quanto na sub-classe. A linguagem quando encontra a chamada a um método, inicia a busca pela classe onde se encontra a chamada e, se não a localizar, continua a busca subindo a hierarquia de classes. O acesso direto a métodos sobrescritos de super-classes por classes derivadas é feito com a sintaxe `super`.

```
super.metodo(<parametros>);
```

No caso de construtores, uma sub-classe pode acessar o construtor de uma super-classe através da sintaxe `super(<parametros>)`.

Uma característica existente em Java e inexistente em C/C++ é a presença de pacotes e interfaces, responsáveis pela organização mais geral de classes relacionadas. O conceito de interfaces está relacionado ao modo como duas classes distintas responderão ao mesmo método de forma diferente. Já a finalidade dos pacotes é agrupar classes e interfaces relacionadas. Toda a interface de programação Java (API) está organizada com pacotes e interfaces.

Os pacotes são utilizados para categorizar e agrupar classes relacionadas em larga escala. Para declarar que determinada classe, ou conjunto de classes, faz parte de determinado pacote, utiliza-se a sintaxe `package` no início do arquivo onde está definida a classe:

```
package nome_pacote;
```

Os pacotes podem ser organizados em uma hierarquia, sendo que os arquivos com as classes do pacote devem ser colocados na mesma hierarquia de diretórios. Deve-se fazer isso para que o interpretador e o compilador possam localizá-los. Pode-se definir o caminho no qual essas ferramentas localizarão as classes, através da variável de sistema CLASSPATH, a qual pode ser declarada no arquivo autoexec.bat. Existem duas alternativas para utilizar classes que estejam definidas em pacotes: referência direta ou importação de classe. Para utilizar a referência direta, deve-se declarar todo o caminho da hierarquia de pacotes onde a classe está definida. E isso deve ser feito toda vez que você for utilizar aquela classe. A importação de uma classe é feita uma única vez, geralmente antes da declaração da classe que vai utilizá-la. Após importar determinada classe, você pode utilizá-la referenciando seu nome diretamente. Pode-se importar todas as classes públicas em um pacote utilizando-se o * (asterisco) no lugar do nome de uma classe a ser importada.

As interfaces permitem uma forma de herança múltipla. Para implementar uma interface para uma classe, utiliza-se a palavra `implements` seguida da interface a ser implementada. Através do uso de interfaces, duas classes respondem ao mesmo método de forma distinta. As interfaces são declaradas em arquivos-fonte, uma interface por arquivo. Os arquivos com interfaces são compilados em arquivos *.class*. As interfaces complementam e estendem o poder de classes. Quase tudo que você utiliza em uma classe, você pode utilizar em uma interface. Uma das diferenças é que uma interface não pode ser instanciada. Outro detalhe é que, em uma interface, automaticamente todos os métodos são públicos e abstratos, e todas as variáveis são públicas, estáticas e finais. Assim como classes e métodos abstratos, as interfaces oferecem uma coleção de esqueletos de métodos que outras classes implementarão. Os métodos são passados para as sub-classes somente como descrições, sem nenhuma implementação ou existência de variáveis de instância. Isso ajuda a eliminar muita da complexidade envolvida com a herança múltipla completa.

APÊNDICE B - IMPLEMENTAÇÃO DO PACOTE MATEMÁTICO LABORE.MATH

B.1. Implementação da classe LUDecomposition

A primeira classe no pacote matemático é a classe LUDecomposition, a qual implementa o método da Decomposição LU para resolução de sistemas de equações algébricas lineares.

A implementação da classe LUDecomposition, para resolução de um sistema de equações algébricas lineares é realizada em quatro etapas básicas:

Etapa 1 - importação da classe LUDecomposition no início do código:

```
import labore.math.LUDecomposition;
```

Etapa 2 - criação de uma classe derivada da classe LUDecomposition:

```
public class MyClass extends labore.math.LUDecomposition
{
   ...
}
```

Etapa 3 - chamada ao método `defineInitialSystem`, na inicialização da classe, para definição dos parâmetros iniciais do sistema de equações algébricas lineares. Este método é polimorfo e oferece duas opções de utilização conforme os parâmetros conhecidos do sistema:

se é conhecido apenas o número de equações:

```
...
/* Define um sistema de 3 equações. */
defineInitialSystem(3);
...
```

se é conhecido o número de equações, a matriz de coeficientes e a matriz de resultados:

```
...

/* Define um sistema de 3 equações: -x1+12x2=43, -8x0+10x1+3x2=8 e 4x0-5x1+7x2=-
15. */

double a[3][3] = { 0., -1., 12, -8, 10, 3, 4, -5, 7 };

double b[3] = { 43., 8., -15 };

defineInitialSystem(3, a, b);

...
```

A qualquer momento durante a utilização do programa é possível alterar os coeficientes das equações através dos métodos para adição de um único coeficiente (método `addCoeff`), adição de uma linha de coeficientes (método `addLine`), adição de uma coluna de coeficientes (método `addColumn`), adição de uma matriz de coeficientes (método `addMatrixCoeff`) e adição conjunta da matriz de coeficientes e da matriz de resultados (método `addMatrix`).

```
...

/* Define o coeficiente da variável 0 na equação 0 com o valor de -1.5. */

addCoeff(0, 0, -1.5);

...

...

/* Define os coeficientes da variável 1 nas três equações do sistema como -6, 3
e 1. */

double c[3] = { -6., 3., 1. };

addColumn(1, c);

...
```

Etapa 4 - chamada ao método `calcSystem` para resolução do sistema. O método retorna uma string informando se o cálculo foi bem sucedido ou não. Os valores possíveis de retorno do método `calcSystem` são `getRetOK` e `getRetERROR`. A solução numérica do sistema pode ser obtida através da chamada ao método `getVariables`.

```
...

/* Resolve o sistema de equações e imprime na saída padrão do sistema o valor
das raízes. */

String s=calcSystem();

if(s.equals(getRetOK()))

{
    for(int i=0;i<n;i++)

    {

        System.out.println("RAIZ "+i+" = "+getVariables(i));

    }

}

...
```

B.2. Implementação da classe NRaphson

Para resolução de sistemas de equações algébricas não-lineares foi desenvolvida a classe NRaphson, que é derivada da classe LUDecomposition. A classe NRaphson implementa o método de Newton-Raphson, o qual reduz o sistema de equações algébricas não-lineares a um sistema de equações algébricas lineares, cujo método de resolução é implementado na classe LUDecomposition.

A implementação da classe NRaphson, para resolução de um sistema de equações algébricas não-lineares é realizada em cinco etapas:

Etapa 1 - importação da classe NRaphson no início do código

```
import labore.math.NRaphson;
```

Etapa 2 - criação de uma classe derivada da classe NRaphson:

```
public class MyClass extends labore.math.NRaphson

{

     ...

}
```

Etapa 3 - chamada ao método `defineInitialSystem`, na inicialização da classe, para definição dos parâmetros iniciais do sistema de equações algébricas não-lineares. O método é polimorfo e oferece três opções de utilização conforme os parâmetros conhecidos do sistema:

se é conhecido apenas o número de equações:

```
...

/* Define um sistema de 3 equações. */

defineInitialSystem(3);

...
```

se são conhecidos o número de equações, o número de tentativas e a tolerância na avaliação das variáveis e das equações:

```
...

/* Define um sistema de 3 equações, com até 100 tentativas para a obtenção da
tolerância de 10^-8 na avaliação das variáveis e das equações. */

defineInitialSystem(3, 100, 1e-8, 1e-8);

...
```

se são conhecidos o número de equações, o número de tentativas, o valor inicial das variáveis e a tolerância na avaliação das variáveis e equações:

```
...
```

```
/* Define um sistema de 3 equações, com até 100 tentativas para a obtenção da
tolerância de 10^-8 na avaliação das variáveis e das equações e cujos valores
iniciais das variáveis são: x0=0, x1=5 e x2=-1,5. */

double xi[3] = { 0., 5., -1.5 };

defineInitialSystem(3, 100, xi, 1e-8, 1e-8);

...
```

Etapa 4 - definição das equações que compõem o sistema de equações algébricas não-lineares através da declaração do método abstrato `system`. Pelo fato do método ser abstrato, deve-se sobrescrever o mesmo na implementação da classe com o modelo matemático. O sistema de equações é definido através da declaração das negativas e das derivadas parciais das equações. A declaração equivalente é realizada utilizando-se o método `addCoeff`. Para as negativas das equações deve-se definir o segundo argumento do método `addCoeff` como `getN` – o número de equações – , enquanto que para as derivadas parciais o segundo argumento é definido como o número da variável derivativa. O primeiro argumento do método `addCoeff` define o número da variável e o último argumento define a expressão da negativa ou derivada parcial da equação.

```
...

/* Define o sistema de equações: x1*x2-10=0, x3/x1+5=0 e x1*(x2+x3)-5=0. */

protected void system()

{
  /* negativa das equações.

     [-f0, -f1, ..., -fn-1] */

  double var[getN()];

  for(int i=0;i<getN();i++) { var[i]=getVariables(i); }

  addCoeff(0, getN(), -var[0]*var[1]+10);

  addCoeff(1, getN(), -(var[2]/var[0])-5);

  addCoeff(2, getN(),-var[0]*(var[1]*var[2])+5);
```

```
  /* derivadas parciais das equações.

     [df0/dx0, df0/dx1, ..., dfn-1/dxn-2, dfn-1/dxn-1] */

  addCoeff(0, 0, var[1]);

  addCoeff(0, 1, var[0]);

  addCoeff(0, 2, 0);

  addCoeff(1, 0, -var[2]/Math.pow(var[0],2));

  addCoeff(1, 1, 0);

  addCoeff(1, 2, 1/var[0]);

  addCoeff(2, 0, var[1]+var[2]);

  addCoeff(2, 1, var[0]);

  addCoeff(2, 2, var[0]);

}

...
```

Etapa 5 - chamada ao método `calcSystem` para resolução do sistema. O método retorna uma string indicando se o cálculo foi bem sucedido ou não. Os valores possíveis de retorno do método `calcSystem` são `getRetOK`, `getRetERROR` e `getRetERROR2`. A solução numérica do sistema é obtida através da chamada ao método `getVariables`.

```
...

/* Resolve o sistema de equações e imprime na saída padrão do sistema o valor
das raízes. */

String s=calcSystem();

if(s.equals(getRetOK()))

{
```

```
    for(int i=0;i<n;i++)
    {
        System.out.println("RAIZ "+i+" = "+getVariables(i));
    }
}
...
```

B.3. Implementação da classe RungeKutta4

A classe RungeKutta4 implementa o método de Runge-Kutta de 4ª ordem, com monitoramento de erro e passo de integração auto-ajustável, para resolução de sistemas de equações diferenciais ordinárias em problemas de valor inicial. O problema de valor inicial é um dos problemas mais comumente encontrados na Engenharia, fato este que tornou o método de Runge-Kutta de 4ª ordem amplamente difundido.

A implementação da classe RungeKutta4 para resolução de um sistema de equações diferenciais ordinárias é realizada em cinco etapas:

Etapa 1 - importação da classe RungeKutta4 no início do código:

```
import labore.math.RungeKutta4;
```

Etapa 2 - criação de uma classe derivada da classe RungeKutta4:

```
public class MyClass extends labore.math.RungeKutta4
{
    ...
}
```

Etapa 3 - chamada ao método `defineInitialSystem`, na inicialização da classe, para a definição inicial dos parâmetros do sistema de equações diferenciais ordinárias. Este método é polimorfo e oferece três opções de utilização conforme os parâmetros conhecidos do sistema:

se são conhecidos o número de equações, o passo de integração e a precisão:

```
...

/* Define um sistema de 3 equações com passo de integração inicial de 0,2 e
precisão de 10^-8. */

defineInitialSystem(3, 0.2, 1e-8);

...
```

se são conhecidos o número de equações, o passo de integração, o passo de integração mínimo e a precisão:

```
...

/* Define um sistema de 3 equações com passo de integração inicial de 0,2,
passo mínimo de integração de 10^-10 e precisão de 10^-8. */

defineInitialSystem(3, 0.2, 1e-10, 1e-8);

...
```

se são conhecidos o número de equações, o passo de integração, a precisão e o valor inicial das variáveis:

```
...

/* Define um sistema de 3 equações com passo de integração inicial de 0,2,
precisão de 10^-8 e valor inicial das variáveis de: y0=0, y1=1 e y2=350. */

double y[] = { 0.0, 1.0, 350.0 };

defineInitialSystem(3, 0.2, 1e-8, y);

...
```

A qualquer momento durante a utilização do programa é possível alterar o valor das variáveis através da utilização do método `setVariables`.

```
...
```

```
/* Define o valor das variáveis como: y0=0 e y1=1. */

double y[] = { 0.0, 1.0 };

setVariables(y);

...
```

Etapa 4 - definição das equações que compõem o sistema de equações diferenciais ordinárias através da declaração do método abstrato `derivs`. Pelo fato do método ser abstrato, deve-se sobrescrever o mesmo na implementação da classe com o modelo matemático. Deve-se declarar o lado direito das derivadas no vetor `dydx[0..n-1]`.

```
...

/* Define o sistema de equações: dy0/dx=1/x e dy1/dx=50*y1+x. */

public void derivs(double x, double[] y, double[] dydx)

{

    dydx[0] = 1/x;

    dydx[1] = 50*y[1]+x;

}

...
```

Etapa 5 - chamada ao método `calcSystem` para resolução do sistema. O método realiza a integração do sistema de equações definido no método `derivs`, entre os limites de integração definidos em seu argumento. O método retorna uma string indicando se o cálculo foi bem sucedido ou não. Os possíveis valores de retorno do método `calcSystem` são `getRetOK`, `getRetERROR` e `getRetERROR2`. A solução numérica do sistema é obtida através da chamada ao método `getVariables`.

```
...
```

```
/* Resolve o sistema de equações entre o intervalo de integração [0;10] e
imprime na saída padrão do sistema o valor das variáveis ao final do intervalo.
*/

String s = calcSystem(0.0, 10.0);

if(s.equals(getOK()))

{

   for(int i=0; i<getN(); i++)

   {

      System.out.println("Variável "+i+" = "+getVariables(i));

   }

}

...
```

B.4. Implementação da classe OrthogonalCollocation

A classe OrthogonalCollocation, que é derivada da classe NRaphson, implementa o método da colocação ortogonal para resolução de sistemas de equações diferenciais ordinárias em um problema de valor de contorno. O método da colocação ortogonal reduz um sistema de equações diferenciais ordinárias a um sistema de equações algébricas não-lineares, cujo método de resolução é implementado na classe NRaphson.

A implementação da classe OrthogonalCollocation para resolução de um sistema de equações diferenciais ordinárias é realizada em cinco etapas:

Etapa 1 - importação da classe OrthogonalCollocation no início do código:

```
import labore.math.OrthogonalCollocation;
```

Etapa 2 - criação de uma classe derivada da classe OrthogonalCollocation:

```
public class MyClass extends labore.math.OrthogonalCollocation
```

```
{
    ...
}
```

Etapa 3 - chamada ao método `defineInitialSystem`, na inicialização da classe, para definição dos parâmetros iniciais do sistema de equações diferenciais ordinárias. O método é polimorfo e oferece três opções de utilização conforme os parâmetros conhecidos do sistema:

se for conhecido somente o número de pontos de colocação:

```
...
/* Define um sistema de equações com 3 pontos de colocação. */
defineInitialSystem(3);
...
```

se forem conhecidos o número de pontos de colocação e os coeficientes alfa e beta do polinômio de Jacobi:

```
...
/* Define um sistema de equações com 3 pontos de colocação e com coeficientes
alfa e beta do polinômio de Jacobi iguais a 0 (método dos momentos). */
defineInitialSystem(3, 0, 0);
...
```

se forem conhecidos o número de pontos de colocação, os coeficientes alfa e beta, o número de tentativas e a tolerância:

```
...
/* Define um sistema de equações com 3 pontos de colocação, coeficiente alfa e
beta iguais a 0, e com até 100 tentativas para obtenção de uma tolerância de
10^-8. */
```

```
defineInitialSystem(3, 0, 0, 100, 1e-8);

...
```

Etapa 4 - definição das equações que compõem o sistema de equações diferenciais ordinárias através da declaração do método abstrato `system`. Pelo fato do método ser abstrato, deve-se sobrescrever o mesmo na implementação da classe com o modelo matemático. O sistema de equações é definido através da declaração das negativas e das derivadas parciais das equações de resíduo. Esta declaração é realizada utilizando-se o método `addCoeff`. Para as negativas das equações de resíduo deve-se definir o segundo argumento do método `addCoeff` como `getN` – o número total de pontos de colocação – , enquanto que para as derivadas parciais o segundo argumento é definido como o número do ponto de colocação [0 .. `getN`-1]. O primeiro argumento do método `addCoeff` define o número da equação de resíduo e o último argumento define a expressão da negativa ou derivada parcial da equação de resíduo. O procedimento para utilização do método `system` foi descrito anteriormente no B.2. deste Apêndice – Implementação da classe NRaphson.

Etapa 5 - resolução do sistema através de chamada ao método `calcSystem`. O método retorna uma string indicando se o cálculo foi bem sucedido ou não. Os possíveis valores de retorno do método `calcSystem` são `getRetOK`, `getRetERROR` e `getRetERROR2`. A solução do sistema é obtida através da chamada ao método `getVariables`, e os pontos de colocação são obtidos através da chamada ao método `getCollocationPoints`.

```
...

/* Resolve o sistema de equações e imprime na saída padrão do sistema o valor
das variáveis em cada ponto de colocação. */

String s=calcSystem();

if(s.equals(getRetOK()))

{

   for(int i=0;i<getN();i++)

   {
```

```
      System.out.println("RAIZ       EM       "+getCollocationPoints(i)+"       =
"+getVariables(i));

   }

}

...
```

APÊNDICE C - DOCUMENTAÇÃO DO PACOTE MATEMÁTICO LABORE.MATH

A documentação descrita a seguir foi gerada utilizando-se a ferramenta javadoc, disponível no Kit de Desenvolvimento Java (JDK), e pode ser obtida pela Internet no site Web do pacote labore.math (www.labore.ufsc.br/java/labore_math).

C.1. Índice de classes

Class Index

LUDecomposition
NRaphson
OrthogonalCollocation
RungeKutta4

C.2. Classe labore.math.LUDecomposition

```
java.lang.Object
   |
   +----labore.math.LUDecomposition
```

---

public class **LUDecomposition**

extends Object Class to solve a system of linear algebraic equations, using the LU Decomposition method.

Constructors

LUDecomposition

```
public LUDecomposition()
```

Methods

getRetERROR

```
public String getRetERROR()
```

Get the string representing the calculation failed.

getRetOK

```
public String getRetOK()
```

Get the string representing the calculation successfull.

getN

```
public int getN()
```

Get the number of equations.

getVariables

```
public double[] getVariables()
```

Get the array with the system solution.

getVariables

```
public double getVariables(int i)
```

Get the solution of determined variable i [0..n].

getEquationSolution

```
protected double getEquationSolution(int i)
```

Get the intermediary solution of determined equation.

setRetERROR

```
public void setRetERROR(String value)
```

Set the string representing the calculation failed.

setRetOK

```
public void setRetOK(String value)
```

Set the string representing the calculation successfull.

defineInitialSystem

```
protected void defineInitialSystem(int nEquations)
```

Routine to alter the number of equations. Arguments: the new number of equations.

defineInitialSystem

```
protected void defineInitialSystem(int nEquations,
                                   double ap[][],
                                   double bp[])
```

Routine to alter the number of equations. Arguments: the new number of equations, the array of coefficients and the array of results.

addCoeff

```
public void addCoeff(int i,
                     int j,
                     double value)
```

Routine to insert individual coefficients. Arguments: the number of the equation[0..n-1], the number of variable [0 ... n-1] (if equal to n, the variable is the result of equation) and the value.

addLine

```
public int addLine(int i,
```

```
                    double value[])
```

Routine to insert a line of coefficients. Arguments: the number of the equation [0 ... n-1] (if equal to n, the values are inserted in the array of results) and the array with values.

addColumn

```
public int addColumn(int i,
                     double v[])
```

Routine to insert a column of coefficients. Arguments: the number of the variable [0 ... n-1] (if equal to n, the values are inserted in the array of results) and the array with values.

addMatrixCoeff

```
public int addMatrixCoeff(double v[][])
```

Routine to insert a complete array of coefficients in the array of coefficients. Argument: the array with values.

addMatrix

```
public int addMatrix(double v[][])
```

Routine to insert a complete array of coefficients in the array of coefficients and in the array of results. Argument: the array with values.

calcSystem

```
public String calcSystem()
```

Routine to solve the system of equations. The return value is a string representing whether the calculation was successful or not (possible values are retOK and retERROR).

C.3. Classe labore.math.NRaphson

```
java.lang.Object
    |
    +----labore.math.LUDecomposition
            |
            +----labore.math.NRaphson
```

---

public class **NRaphson**

extends LUDecomposition Class to solve a system of non-linear algebraic equations, using the Newton-Raphson method.

Constructors

NRaphson

```
public NRaphson()
```

Methods

getRetERROR2

```
public String getRetERROR2()
```

Get the string representing a failed calculation.

getNtrial

```
public int getNtrial()
```

Get the number of trials.

getTolx

```
public double getTolx()
```

Get the tolerance in variables evaluation.

getTolf

```
public double getTolf()
```

Get the tolerance in equations evaluation.

getVariables

```
public double[] getVariables()
```

Get the array with solution of the system of equations.

Overrides:

getVariables in class LUDecomposition

getVariables

```
public double getVariables(int i)
```

Get the solution of determined variable [0..n-1].

Overrides:

getVariables in class LUDecomposition

setRetERROR2

```
public void setRetERROR2(String value)
```

Set the string representing a failed calculation.

setNtrial

```
public void setNtrial(int value)
```

Set the number of trials.

setVariables

```
public void setVariables(double array[])
```

Set the array with the initial values of variables.

setTolx

```
public void setTolx(double value)
```

Set the tolerance in variables evaluation.

setTolf

```
public void setTolf(double value)
```

Set the tolerance in equations evaluation.

defineInitialSystem

```
protected void defineInitialSystem(int nEquations)
```

Routine to define the initial system of equations. Argument: the number of equations.

Overrides:

defineInitialSystem in class LUDecomposition

defineInitialSystem

```
protected void defineInitialSystem(int nEquations,
                                   int nt,
                                   double tx,
                                   double tf)
```

Routine to define the initial system of equations. Arguments: number of equations, number of trials, tolerance in evaluation of variables and tolerance in evaluation of equations.

defineInitialSystem

```
protected void defineInitialSystem(int nEquations,
                                   int nt,
                                   double xi[],
                                   double tx,
                                   double tf)
```

Routine to define the initial system of equations. Arguments: number of equations, number of trials, array of initial value of variables, tolerance in evaluation of variables and tolerance in evaluation of equations.

calcSystem

```
public String calcSystem()
```

Routine to solve the system of equations. The return value is a string representing whether the calculation is successful or not (possible values are retOK, retERROR and retERROR2).

Overrides:

calcSystem in class LUDecomposition

system

```
protected abstract void system()
```

Routine that describes the system of non-linear algebraic equations. This routine is abstract and should be implemented in the subclass. The array b[0..n-1] contains the declaration of negative values fo the functions and the array a[0..n-1][0..n-1] contains the declaration of partial derivatives of the functions.

C.4. Classe labore.math.OrthogonalCollocation

```
java.lang.Object
    |
    +----labore.math.LUDecomposition
            |
            +----labore.math.NRaphson
                    |
                    +----labore.math.OrthogonalCollocation
```

---

public class **OrthogonalCollocation**

extends NRaphson Class to solve a system of partial differential equations, using the Orthogonal Collocation method.

Constructors

OrthogonalCollocation

```
public OrthogonalCollocation()
```

Methods

getAlpha

```
public int getAlpha()
```

Get the alpha coefficient.

getBeta

```
public int getBeta()
```

Get the alpha coefficient.

getCollocationPoints

```
public double[] getCollocationPoints()
```

Get the collocation points.

getCollocationPoints

```
public double getCollocationPoints(int i)
```

Get the determined collocation point i.

getAij

```
public double getAij(int i,
                     int j)
```

Get determined coefficient aij of residue equation. Arguments: the indexes i and j of the desired coefficient.

getBij

```
public double getBij(int i,
                     int j)
```

Get determined coefficient bij of residue equation. Arguments: the indexes i and j of the desired coefficient.

getVariables

```
public double[] getVariables()
```

Get the array with the system solution.

Overrides:

getVariables in class NRaphson

getVariables

```
public double getVariables(int i)
```

Get the solution of determined variable i.

Overrides:

getVariables in class NRaphson

setAlpha

```
public void setAlpha(int value)
```

Set the alpha coefficient.

setBeta

```
public void setBeta(int value)
```

Set the beta coefficient.

setAij

```
public boolean setAij(int i,
                      int j,
                      double value)
```

Set determined coefficients aij of residue equation. Return: true if operation Ok or false if illegal operation - index out of bounds.

setBij

```
public boolean setBij(int i,
                      int j,
                      double value)
```

Set determined coefficients bij of residue equation. Return: true if operation Ok or false if illegal operation - index out of bounds.

defineInitialSystem

```
protected void defineInitialSystem(int nP)
```

Routine to alter the number of equations. Argument: number of collocation points.

Overrides:

defineInitialSystem in class NRaphson

defineInitialSystem

```
protected void defineInitialSystem(int nP,
                                   int a,
                                   int b)
```

Routine to alter the number of equations. Arguments: number of collocation points, alpha coefficient and beta coefficient.

defineInitialSystem

```
protected void defineInitialSystem(int nP,
                                   int a,
                                   int b,
                                   int ntrial,
                                   double tol)
```

Routine to alter the number of equations. Arguments: number of collocation points, alpha coefficient, beta coefficient, number of tries and tolerance in results.

calcSystem

```
public String calcSystem()
```

Routine to solve the system of equations. The return value is a string representing whether the calculation is successful or not (possible values are retOK, retERROR and retERROR2).

Overrides:

calcSystem in class NRaphson

C.5. Classe labore.math.RungeKutta4

```
java.lang.Object
   |
   +----labore.math.RungeKutta4
```

---

public class **RungeKutta4**

extends Object Class to solve a system of ordinary differential equations (ODE's), using the 4th order Runge-Kutta method, whit error monitoring and adjustable steps.

Constructors

RungeKutta4

```
public RungeKutta4()
```

Methods

setNVar

```
public void setNVar(int n)
```

Set the number of variables.

setPrecision

```
public void setPrecision(double p)
```

Set the precision.

setH

```
public void setH(double h)
```

Set the integration step.

setHmin

```
public void setHmin(double h)
```

Set the minimum value for the integration step.

setRetOK

```
public void setRetOK(String s)
```

Set the string representing calculation successful.

setRetERROR1

```
public void setRetERROR1(String s)
```

Set the string representing calculation failed (step very small).

setRetERROR2

```
public void setRetERROR2(String s)
```

Set the string representing calculation failed (many steps).

setVariables

```
public void setVariables(double value[])
```

Set the n variables values [0..n-1].

setDerivatives

```
public void setDerivatives(double value[])
```

Set the n derivatives [0..n-1].

getNVar

```
public int getNVar()
```

Get the number of variables.

getPrecision

```
public double getPrecision()
```

Get the precision.

getH

```
public double getH()
```

Get the integration step.

getHmin

```
public double getHmin()
```

Get the minimum value for the integration step.

getRetOK

```
public String getRetOK()
```

Get the string representing calculation successful.

getRetERROR1

```
public String getRetERROR1()
```

Get the string representing calculation failed (step very small).

getRetERROR2

```
public String getRetERROR2()
```

Get the string representing calculation failed (many steps).

getVariables

```
public double[] getVariables()
```

Get the variables values.

getDerivatives

```
public double[] getDerivatives()
```

Get the derivatives.

defineInitialSystem

```
public void defineInitialSystem(int nEq,
                                double step,
                                double precision)
```

Define the initial system od ordinary differential equations. Arguments: number of equations, integration step and precision.

defineInitialSystem

```
public void defineInitialSystem(int nEq,
                                double step,
                                double precision,
                                double variables[])
```

Define the initial system of ordinary differential equations. Arguments: number of equations, integration step, step, precision and initial values of n variables [0..n-1].

defineInitialSystem

```
public void defineInitialSystem(int nEq,
                                double step,
```

```
                                    double minStep,
                                    double precision)
```

Define the initial system of ordinary differential equations. Arguments: number of equations, integration step, minimum integration step and precision.

calcSystem

```
public String calcSystem(double x1,
                         double x2,
                         double ystart[])
```

Routine to solve the system of equations. Arguments are: the initial values of variables, and the initial and final integration limit. The return is a string representing if the calculation is successful or not (possible values are retOK, retERROR1, retERROR2).

calcSystem

```
public String calcSystem(double x1,
                         double x2)
```

Routine to solve the system of equations. Arguments are: the initial values of variables, and the initial and final integration limit. The return is a string representing if the calculation is successful or not (possible values are retOK, retERROR1, retERROR2).

derivs

```
protected abstract void derivs(double x,
                               double y[],
                               double dydx[])
```

Routine where are described the system of ordinary diferential equations. This routine is abstract and should be implemented in the subclass. The array dydx[0..nvar-1] contain the declaration of the right side of differential equations.

C.6. Índice de todas variáveis e métodos

A

addCoeff(int, int, double). Method in class labore.math.LUDecomposition

Routine to insert individual coefficients.

addColumn(int, double[]). Method in class labore.math.LUDecomposition

Routine to insert a column of coefficients.

addLine(int, double[]). Method in class labore.math.LUDecomposition

Routine to insert a line of coefficients.

addMatrix(double[][]). Method in class labore.math.LUDecomposition

Routine to insert a complete array of coefficients in the array of coefficients and in the array of results.

C

D

G

Get the derivatives.

getEquationSolution(int). Method in class labore.math.LUDecomposition

Get the intermediary solution of determined equation.

getH(). Method in class labore.math.RungeKutta4

Get the integration step.

getHmin(). Method in class labore.math.RungeKutta4

Get the minimum value for the integration step.

getN(). Method in class labore.math.LUDecomposition

Get the number of equations.

getNtrial(). Method in class labore.math.NRaphson

Get the number of trials.

getNVar(). Method in class labore.math.RungeKutta4

Get the number of variables.

getPrecision(). Method in class labore.math.RungeKutta4

Get the precision.

getRetERROR(). Method in class labore.math.LUDecomposition

Get the string representing the calculation failed.

getRetERROR1(). Method in class labore.math.RungeKutta4

Get the string representing calculation failed (step very small).

getRetERROR2(). Method in class labore.math.NRaphson

Get the string representing a failed calculation.

getRetERROR2(). Method in class labore.math.RungeKutta4

Get the string representing calculation failed (many steps).

getRetOK(). Method in class labore.math.LUDecomposition

Get the string representing the calculation successfull.

getRetOK(). Method in class labore.math.RungeKutta4

Get the string representing calculation successful.

getTolf(). Method in class labore.math.NRaphson

L

N

O

R

S

setRetERROR2(String). Method in class labore.math.NRaphson

Set the string representing a failed calculation.

setRetERROR2(String). Method in class labore.math.RungeKutta4

Set the string representing calculation failed (many steps).

setRetOK(String). Method in class labore.math.LUDecomposition

Set the string representing the calculation successfull.

setRetOK(String). Method in class labore.math.RungeKutta4

Set the string representing calculation successful.

setTolf(double). Method in class labore.math.NRaphson

Set the tolerance in equations evaluation.

setTolx(double). Method in class labore.math.NRaphson

Set the tolerance in variables evaluation.

setVariables(double[]). Method in class labore.math.NRaphson

Set the array with the initial values of variables.

setVariables(double[]). Method in class labore.math.RungeKutta4

Set the n variables values [0..n-1].

system(). Method in class labore.math.NRaphson

Routine that describes the system of non-linear algebraic equation