UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

# UM AMBIENTE DE SIMULAÇÃO PARA PROCESSOS INDUSTRIAIS E SISTEMAS DE CONTROLE

Dissertação submetida à Universidade Federal de Santa Catarina para a obtenção do grau de Mestre em Engenharia Elétrica

**Juan Antonio Salvatierra Gimenes**

Florianópolis, 16 de Setembro de 1993

# UM AMBIENTE DE SIMULAÇÃO PARA PROCESSOS INDUSTRIAIS E SISTEMAS DE CONTROLE

JUAN ANTONIO SALVATIERRA GIMENES

Esta dissertação foi julgada adequada para a obtenção do título de

**MESTRE EM ENGENHARIA**

especialidade Engenharia Elétrica, área de concentração Sistemas de Controle e Automação Industrial, e aprovada na sua forma final pelo programa de Pós-graduação.

Prof. Julio Elias Normey Rico, MSc.
Orientador

Prof. Roberto de Souza Salgado, Ph.D.
Coordenador do Curso de Pós-graduação em Engenharia Elétrica

Banca examinadora:

Prof. Julio Elias Normey Rico, MSc.
Orientador

Prof. Edson Roberto De Pieri, Dr.
Co-orientador

Prof. Ariovaldo Bolzan, Dr.

Prof. Ivanil S. Bonatti, Dr.

A Laura,
minha companheira, minha melhor amiga.

# AGRADECIMENTOS

À CAPES, pelo apoio financeiro na realizacão deste trabalho.

Aos professores Julio Elias Normey Rico e Edson Roberto De Pieri, pela orientação e amizade.

Aos amigos Fernando Mussoi, Marcos, Mauricio, Ricardo, Márcio, André Medeiros, Thair, Glaucio, Zique, Cristina, Carla, Daniel, Alexandre, Romeu, Otavio, Keiko, Arão, Vinicius, João, Leonardo, Eliane, Elvio, Claudine, Osair, André Novotny, Yuji, Xiao-Bing, Paulo Valim, Aldebaro, Idmilson, Paulo André, Sergio, Fernando Busch, Elizete, Jose Eduardo, Denise, Cézar, Raimundo, Jaqueline, Luciana, Iberê, Tristão, Yolando, Roberto, Elieser, Marcelo, Alberto, Lucinéia e Eduardo, pela amizade e pelas discussões construtivas, e a todos aqueles que de uma ou de outra forma colaboraram para a realização deste trabalho.

Agradeço sobretudo, a minha família, pelo constante apoio e incentivo que souberam transmitir apesar da distância que nos separa.

# SUMÁRIO

# RESUMO

O presente trabalho é resultado da demanda da área de sistemas de controle por pacotes assistidos por computador voltados à modelagem, simulação, análise e projeto de controladores associados a processos industriais, cuja natureza geralmente é multivariável e não-linear. Este pacote foi implementado utilizando-se a linguagem de programação C++ e as rotinas gráficas do XView[1], sobre o ambiente UNIX[2] em estações de trabalho.

Este ambiente surgiu principalmente devido à necessidade de pacotes capazes de simular processos lineares ou não-lineares e monovariáveis ou multivariáveis, através de uma interface gráfica atraente ao usuário.

O pacote é também um ambiente integrado porque utiliza uma biblioteca de tipos abstratos de dados relativa à área de controle clássico (tipos tais como função de transferência e variáveis de estado).

[1] XView é marca registrada da SUN Microsystems, Inc.
[2] UNIX é marca registrada da AT&T.

# *ABSTRACT*

*The present work is result of control systems needs for control aided design (CAD) systems to model, simulate, analyse and design controllers associated to industrial processes that, usually, are multivariable and non-linear. This CAD was implemented using C++ programming language and XView[1] graphic package, running under UNIX[2] on workstations.*

*This environment appeared because of the needs for computer packages able to simulate linear or non-linear and monovariable or multivariable processes, over an user-attractive graphic interface.*

*This CAD is also an integrated environment because it uses an abstract data type library relative to modern control (types such as transfer function and dynamic equation).*

[1] *XView is a trademark of SUN Microsystems, Inc.*
[2] *UNIX is a trademark of AT&T.*

# CAPÍTULO 1: INTRODUÇÃO

O desenvolvimento de sistemas complexos, devido a uma demanda, não só quantitativa mas também qualitativa, tornou necessária a adoção de uma metodologia de trabalho para o desenvolvimento dos mesmos. A metodologia adotada, seja qual for, deve estabelecer procedimentos e técnicas que tornem eficientes e mais confiáveis os sistemas produzidos e permitam uma melhor evolução destes [Fairley 85, Farines 86, Pressman 87].

Uma atividade geral a toda metodologia de desenvolvimento de sistemas é o projeto. O projeto em engenharia pode ser definido como o conjunto de atividades que permitem propor uma alternativa concreta e viável de solução para um problema cujos objetivos foram pré-estabelecidos. O projeto necessita de um conhecimento profundo dos objetivos a serem alcançados, da sua análise em função de experiências anteriores e das técnicas e ferramentas existentes. A partir disto, poderá realizar-se um processo de tomada de decisão, o qual levará à escolha da alternativa mais adequada entre aquelas oriundas da criatividade dos projetistas. Neste processo, vários critérios podem ser adotados, como por exemplo, a avaliação do desempenho de cada alternativa proposta.

Uma técnica amplamente usada para avaliar o desempenho de sistemas é a simulação. O propósito da simulação é predizer o comportamento de um sistema uma vez desenvolvido um modelo do mesmo, possibilitando a pesquisa sem ter acesso ao sistema real.

A introdução do computador no projeto deu origem a novas concepções e técnicas que podem ser agrupadas no que convencionou-se chamar de projeto assistido por computador (PAC). Graças aos avanços tecnológicos, as técnicas de PAC evoluíram no sentido de tornar mais amigável a relação homem-máquina. Isto permitiu que a atenção do projetista seja dirigida a atividades mais criativas, resultando no aumento da qualidade e da confiabilidade dos sistemas desenvolvidos.

A introdução destas ferramentas de apoio ao projeto, implica na necessidade de realizar-se um esforço no sentido de adquirir uma competência tanto na utilização dos programas como na sua produção. Como resultado da demanda da área de sistemas de controle por pacotes assistidos por computador voltados à modelagem, simulação, análise e projeto de controladores, o Laboratório de Controle e Microinformática (LCMI) está desenvolvendo programas utilizáveis a nível acadêmico.

A modelagem matemática e a simulação são ferramentas indispensáveis para a análise e projeto de controladores para processos industriais não-lineares complexos. Para a realização da simulação, é necessário seguir três etapas:

- Desenvolver um modelo matemático do processo e seus controladores.
- Solucionar as equações do modelo.
- Analisar os resultados.

A idéia de implementar um pacote computacional capaz de simular processos lineares ou não-lineares e monovariáveis ou multivariáveis, utilizando uma interface gráfica atraente ao usuário, surgiu principalmente devido à necessidade de pacotes com estas características no LCMI.

Este trabalho não procura desenvolver um pacote com características novas; na verdade já existem vários pacotes, tais como Simulink (MathWorks) e CTI (Unicamp), que realizam as mesmas funções que se desejam implementar. Porém, comprar um destes pacotes está fora da realidade do LCMI onde há falta de recursos monetários.

Nestas circunstâncias, o desenvolvimento de um pacote básico voltado à modelagem e simulação tornou-se oportuno. Oportuno no sentido de prover ao LCMI de uma ferramenta que permita integrar-se a outras assim como facilitar a sua evolução, já que é requisito fundamental da implementação obedecer certos critérios de qualidade. Esta é uma contribuição importante deste trabalho, pois trabalhos anteriores devido principalmente à falta de documentação eram difíceis de serem aperfeiçoados e reutilizados.

Paralelamente ao desenvolvimento deste pacote, implementou-se outro pacote computacional, voltado à análise de sistemas lineares. Este pacote consiste numa biblioteca de tipos abstratos de dados tais como polinômio, função de transferência, vetor, matriz e variáveis de estado. Esta biblioteca é de grande utilidade para programadores que desenvolvem pacotes computacionais voltados à engenharia de controle, pois são estruturas e funções amplamente utilizadas: ela permite que o esforço do projetista seja direcionada a tarefas mais criativas, economizando-lhe tempo e energia.

Assim, o desenvolvimento de um ambiente de simulação tornou-se uma tarefa mais agradável e menos trabalhosa, no sentido de não ter sido necessário implementar uma versão "particular" destas estrututras e funções. Em termos de qualidade de software, a biblioteca apresenta boa modularidade, robustez, e principalmente, reusabilidade.

O propósito de desenvolver pacotes computacionais satisfazendo critérios de qualidade, deve-se, de fato, ao desejo de facilitar a integração destes produtos num único. Com esta visão,

os membros do LCMI procuram desenvolver pacotes cada vez mais poderosos sem que para isto seja necessário re-implementar o que já foi desenvolvido.

Esta dissertação apresenta as características principais de um PAC voltado à modelagem e simulação de processos industrias, que, geralmente, são de natureza multivariável e não-linear. Este pacote visa, exatamente, contribuir como ferramenta de apoio ao projeto.

No capítulo 2 apresentam-se alguns conceitos importantes relacionados à simulação de sistemas, com o propósito de facilitar uma especificação funcional do ambiente de simulação.

No capítulo 3 descrevem-se detalhes da implementação, permitindo um conhecimento acerca da estrutura dos principais algoritmos implementados no pacote.

No capítulo 4 apresenta-se um exemplo, no qual aplicam-se os principais recursos e ferramentas implementadas no ambiente. Os resultados obtidos são comparados com os de outros pacotes, para comparar a exatidão dos mesmos.

No capítulo 5 apresentam-se as conclusões deste trabalho, assim como algumas perspectivas para o desenvolvimento de pacotes computacionais.

# CAPÍTULO 2: ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA

## 2.1 Introdução

O controle de processos envolve uma gama ampla e diversificada de problemas a resolver. A complexidade destes varia de acordo com a natureza do processo e com as funções de controle que se deseje implementar. Os processos podem ser lineares ou não-lineares, monovariáveis ou multivariáveis, contínuos ou discretos e variantes ou invariantes no tempo.

Existe um vasto conjunto de metodologias, algoritmos, técnicas e ferramentas para o tratamento dos processos anteriormente citados. A escolha do procedimento mais adequado dependerá da natureza do processo e do ambiente em que este se encontra.

A simulação de sistemas é um assunto que ao longo dos anos tem despertado interesse não só dos engenheiros como também de pesquisadores em todas as áreas da ciência. Embora as equações que regem os fenômenos relacionados com processos industrias sejam há muito tempo conhecidas, sua solução analítica é restrita a casos específicos bem conhecidos [Franks 72, Luyben 73]. Desta forma, a tendência atual na solução destes fenômenos é o uso de técnicas numéricas.

As técnicas numéricas, quando implementadas em computadores adequados, permitem a visualização gráfica imediata dos resultados. Desta forma, a sua essência consiste em determinar as grandezas de interesse a um certo fenômeno com maior exatidão, propiciando não só um projeto mais racional dos dispositivos, mas também uma análise mais realista. Eles atualmente representam uma ferramenta poderosa e indispensável em áreas de projeto e instituições de pesquisa e ensino.

Este capítulo apresenta uma breve especificação de requisitos do ambiente, assim como as principais características da modelagem e simulação de sistemas e os recursos escolhidos para o desenvolvimento de um pacote computacional voltado ao estudo de processos industriais e sistemas de controle.

## 2.2 Modelagem de sistemas

Para estudar um sistema é possível, obviamente, realizar experimentos sobre o próprio sistema. O objetivo de muitos estudos de sistemas, entretanto, é predizer como será o

desempenho do mesmo antes de construí-lo. Claramente, não é prático experimentar com um sistema enquanto este se encontra na forma hipotética. Uma alternativa algumas vezes utilizada é construir um número de protótipos, e testá-los, mas isto pode consumir muito tempo e ser dispendioso. Mesmo com um sistema real disponível, não é prático experimentar com o próprio sistema. Por exemplo, não é possível estudar uma planta química pela mudança arbitrária dos parâmetros dos controladores que a compõem. Conseqüentemente, o estudo de sistemas é geralmente conduzido por um modelo do mesmo. Para o propósito da maioria dos estudos, não é necessário considerar todos os detalhes do sistema. Assim, o modelo não é apenas um substituto do sistema, é também uma simplificação dele.

Definimos um modelo como sendo o corpo de informação de um sistema, deduzido para o propósito de estudo do mesmo. Desde que o propósito do estudo determina a natureza da informação que será adquirida, não há apenas um único modelo para o sistema. Diferentes modelos do mesmo sistema seriam produzidos por diferentes analistas interessados em diferentes aspectos do sistema, ou pelo mesmo analista, assim que seu entendimento do sistema muda.

A tarefa de definir um modelo para o sistema pode ser amplamente subdividido em duas subtarefas: estabelecimento da estrutura do modelo e suprimento de dados. O estabelecimento da estrutura determina as fronteiras do sistema e identifica as entidades, atributos e atividades do sistema. Os dados proveêm os valores que os atributos podem ter e define as relações envolvidas nas atividades. Estas duas subtarefas são definidas como partes de uma tarefa ao invés de dois trabalhos separados, porque elas são usualmente tão intimamente relacionadas que uma não pode ser realizada independentemente da outra.

Muitos tipos de modelos têm sido usados no estudo de sistemas e têm sido classificados de várias maneiras. A classificação é feita algumas vezes em termos da natureza do sistema modelado, tal como contínuo versus discreto ou determinístico versus estocástico. Para o propósito deste texto, os modelos serão tratados como modelos físicos ou modelos matemáticos.

Uma segunda distinção pode ser feita entre os modelos. Eles podem ser classificados como modelos estáticos ou modelos dinâmicos. No caso de modelos matemáticos, uma terceira distinção é a técnica empregada na solução do problema. Esta distinção é feita entre modelos analíticos e modelos numéricos.

Não é possível prover regras pelas quais modelos matemáticos possam ser construídos, mas alguns princípios podem ser ditados. Eles não descrevem uma série de passos que levam à construção de um modelo; eles descrevem diferentes pontos de vista que nos permitam julgar as informações a serem incluídas no modelo.

- *Construção de blocos.* A descrição de sistemas pode ser organizada numa série de blocos, ou subsistemas. O propósito de construir blocos é simplificar a especificação das interações dentro do sistema. Cada bloco descreve uma parte do sistema que dependa de umas poucas, preferivelmente uma, variáveis de entrada, e resulte em poucas variáveis de saída. O sistema como um todo pode ser descrito em termos das interconexões entre os blocos, correspondentemente, o sistema pode ser representado graficamente como um simples diagrama de blocos.

- *Relevância.* O modelo deve apenas incluir aqueles aspectos do sistema relevantes aos objetivos do estudo. Embora a informação irrelevante ao modelo possa não provocar dano algum, ela deve ser excluída porque ela aumenta a complexidade do modelo e causa maior trabalho para solucionar o problema.

- *Exatidão.* A exatidão da informação adquirida para o modelo deve ser considerada. Dados inexatos podem invalidar a simulação.

- *Agregação.* Um fator posterior a ser considerado é a extensão ao qual um número de entidades individuais podem ser agrupados em entidades maiores. Em alguns estudos, pode ser necessária a construção de entidades artificiais através do processo de agregação. Considerações similares de agregação devem ser dadas à representação de atividades.

## 2.3 Simulação de sistemas

A simulação de sistemas é considerada uma técnica numérica usando modelos matemáticos dinâmicos. Simulação é definido como a técnica de solução de problemas através do seguimento de mudanças no tempo de um modelo dinâmico de um sistema. A definição é suficientemente ampla para incluir o uso de modelos físicos dinâmicos, no qual as variáveis do modelo são avaliadas por medidas físicas ao invés de cálculos numéricos. Não se tentará discutir posteriormente o uso de modelos físicos, assim que as futuras referências à simulação serão feitas em termos de modelos matemáticos.

Nesta primeira versão do pacote consideram-se apenas processos contínuos. Coloca-se como perspectiva para a continuação deste trabalho a implementação de processos discretos (como por exemplo, PID digital).

A aplicação da técnica de simulação para pesquisa e análise de problemas tem suas vantagens. A seguir, listam-se algumas delas.

- Decisões relativas a futuros sistemas no estágio conceitual podem ser realizadas facilmente. Muitas respostas podem ser obtidas num sistema conceitual prioritariamente nas fases de projeto e desenvolvimento.

- O desempenho do sistema pode ser simulado e observado sobre todas as condições concebíveis. Os parâmetros do sistema e do ambiente podem facilmente ser mudados para simular qualquer condição desejada, e podem-se explorar condições que não possam ser controladas no ambiente natural.

- Os resultados do desempenho do sistema em campo podem ser extrapolados numa simulação para propósitos de predição. Dadas as informações de desempenho em campo, pode-se estender as informações num contexto probabilístico.

- Os eventos do sistema podem ser acelerados numa ordem grande de magnitude. O tempo real pode ser comprimido em velocidades eletrônicas, de modo que um processo real que pode tomar horas ou meses de operação pode ser simulado num computador em frações de segundo ou minutos. A única limitação é a capacidade do computador em si.

- Dados artificiais de um sistema, embora realistas, podem ser obtidos rapidamente em grandes quantidades sem ter que colocar em funcionamento o sistema real.

Questões específicas sobre o sistema o qual pode estar na sua fase conceitual, na sua fase de implementação e teste, ou mesmo já construído, freqüentemente podem só ser respondidas pela simulação. Para aplicar esta técnica, devem-se seguir alguns critérios.

- Devemos aplicar esta técnica sempre e quando as ferramentas analíticas estão indisponíveis ou são inapropriadas para a solução do problema. Sempre e quando hajam problemas demandando solução e não existam ferramentas analíticas convenientes para a sua solução, então, outra abordagem tal como a simulação, deve ser considerada.

- Devemos aplicar esta técnica quando tivermos uma razoável certeza de que o conceito, sistema ou operação pode ser simulado com sucesso. Devemos ser capazes de obter informação suficiente do sistema para dar realismo ao modelo e fazer possível uma simulação com sucesso.

- Devemos aplicar a simulação sempre e quando haja um enorme sistema de equações a ser resolvido. A aplicação de computadores de alta velocidade permitiu a solução de problemas muito complexos num intervalo de tempo razoável.

- Devemos aplicar a simulação sempre e quando o mero processo de construir a simulação de um sistema possa por si mesmo ser uma experiência benéfica à aprendizagem de processos.

## 2.4 Especificação de requisitos

A especificação funcional compreende uma descrição precisa das entradas, das saídas e dos dados que devem ser mantidos pelo sistema. Obter uma definição precisa de um sistema nestes termos está longe de ser uma solução ao problema. Entender as necessidades do usuário numa forma inambigua é uma das partes mais difíceis do desenvolvimento de um sistema. A descrição dos dados é útil tanto para a compreensão da especificação como para a implementação do sistema.

### 2.4.1 Estrutura de dados

Como principal objetivo do ambiente, colocou-se a simulação de processos industriais e sistemas de controle, que geralmente, são de natureza multivariável e não-linear. Devido a estas características, o ambiente deve suportar uma estrutura de modelo que permita declarar o sistema a simular de forma integrada. Este modelo deve descrever formalmente o comportamento do sistema, baseado numa representação matemática das características dinâmicas do mesmo.

A estrutura de dados deve satisfazer alguns requisitos.

- Permitir o fácil acesso a todos os elementos da estrutura para leitura ou modificação de seus respectivos dados.
- Otimizar a memória necessária para o armazenamento dos dados de cada sistema declarado.
- Possibilitar a montagem e modificação de estruturas de forma dinâmica.
- Possibilitar o armazenamento das equações declaradas para os modelos dos sistemas.
- Possibilitar o armazenamento para cada elemento do sistema dos dados necessários à sua definição.

Uma facilidade que deve ser oferecida ao usuário é a criação de sub-modelos. Estes sub-modelos representam, na verdade, processos comumente usados na indústria. O usuário poderá construir modelos a partir de um conjunto de sub-modelos previamente definidos. Isto tornará a tarefa de declaração muito mais amigável, pois o usuário trabalharia com estruturas mais significativas. Um motor elétrico, um tanque de vazão por gravidade, uma bomba compressora são exemplos de processos que integrados formam um modelo de um processo maior. Outra

vantagem da utilização de sub-modelos é a reusabilidade dos dados declarados, isto é, os sub-modelos podem ser utilizados dentro de vários modelos.

Os dados de entrada para o modelo devem ser digitados diretamente pelo usuário ou lidos de arquivos. Estes dados, por sua vez, podem ser armazenados em disco, através de funções de acesso a diretórios e arquivos.

### 2.4.2 Integração numérica

Devem-se implementar algoritmos de integração numérica capazes de simular modelos associados a qualquer tipo de sistema multivariável e não-linear, para que o usuário possa escolher aquele que melhor se adeqüe a suas necessidades.

### 2.4.3 Gerenciamento entrada/saída

O gerenciador da entrada e saída de dados deve coordenar a execução do programa através de chamadas às operações a serem realizadas, e, conseqüentemente, o acesso às diversas unidades funcionais. Ele é formado por um alocador dinâmico de memória, o qual gerencia a parte de memória reservada para o armazenamento de dados formais, e de um seqüenciador de tarefas, o qual valida tarefas em função do estado atual de execução do programa e do status em função das tarefas realizadas.

O conjunto de unidades funcionais do programa divide-se em algumas operações.

- *Analisador sintático.* Analisa os dados introduzidos pelo usuário.
- *Manipulador de erros.* Permite tratar os erros cometidos durante a execução do programa, tanto a nível do gerenciador como das unidades funcionais, ativando mensagens de erro e fornecendo auxílio para o usuário.
- *Controlador de base de dados.* É o responsável pela coordenação de todas as operações que envolvam o armazenamento e/ou manipulação e o acesso de dados na memória.
- *Utilitários.* Contém o conjunto de rotinas de uso geral (operações com polinômios, matrizes, etc.).
- *Tratamento gráfico.* Contém as rotinas destinadas ao traçado e à manipulação dos gráficos.

### 2.4.4 Transformações auxiliares

Para a aplicação das ferramentas clássicas de análise temporal e freqüencial sobre o modelo declarado, é necessário transformar a representação do sistema. Deve-se, portanto, efetuar algumas transformações nestas representações.

Assim, deve-se implementar um algoritmo que calcule os coeficientes das matrizes A, B, C e D que representam o modelo incremental, através de sua equação dinâmica, em torno de uma condição de operação de um modelo multivariável e não-linear.

Deve-se implementar também um algoritmo que efetue a transformação de um modelo incremental, representado por sua equação dinâmica, para a matriz de transferência correspondente. Esta transformação é necessária para permitir que se utilizem as ferramentas de análise freqüencial, lugar das raízes, etc.

### 2.4.5 Interface homem-máquina

Todo programa pode ser dividido em duas partes.

- Uma parte interativa, que permite a interação do usuário com o programa.
- Uma parte operativa, que realiza todas as operações internas escolhidas pelo usuário e durante a qual este não tem ação possivel.

A parte interativa do programa está organizada na forma de uma interface homem-máquina, que permite ao usuário interagir com o programa na declaração e na simulação do modelo. Esta interface representa o mecanismo através do qual o usuário acessa as ferramentas disponíveis no programa, e deve oferecer facilidades tais como janelas de interação, menus de seleção, diálogos do tipo pergunta-resposta e diferentes possibilidades de visualização dos resultados.

## 2.5 Recursos computacionais

### 2.5.1 Hardware

Sendo o desenvolvimento deste tipo de sistema muito ligado aos recursos computacionais disponíveis, a escolha de um equipamento é de fundamental importância. Um dos equipamentos que pode ser utilizado é o PC (Personal Computer), os quais na maioria das vezes usa o sistema

operacional DOS[1] (Disk Operating System). Embora de fácil manuseio e aquisição, ele impõe algumas restrições que impedem que sistemas mais potentes sejam desenvolvidos.

As principais dificuldades se referem ao limite de memória que o sistema operacional impõe (640 kbytes) e à velocidade de processamento. Muito embora já existam computadores pessoais e processadores que estão aptos a usarem mais do que este limite, uma aplicação normal no DOS não pode ultrapassar o limite de 640 kbytes, já que o mesmo é uma imposição do sistema operacional e não do hardware da máquina. Ainda que o limite de memória possa ser resolvido, por exemplo, a partir da adoção de um outro sistema operacional (OS/2, Windows NT, Xenix, etc.), isto não tem sido uma prática comum entre os usuários.

Além disso as exigências do usuário no que se refere a interface dos sistemas têm aumentado a partir da popularização do uso de interfaces gráficas em substituição às interfaces baseadas em linhas de comando. Neste aspecto, deve-se ressaltar que não existe no momento um padrão definitivo de interface gráfica para computadores pessoais [Vecchiet 87, Perry 89, Andersen 91, Sheldon 91].

Por outro lado, o uso crescente de estações de trabalho no Brasil tem motivado o desenvolvimento de pacotes computacionais adaptados a este tipo de equipamento. Dotadas de capacidade de memória, velocidade de processamento e recursos gráficos superiores ao dos PC's, elas representam atualmente uma excelente opção para o desenvolvimento de programas. Contudo, o preço de uma estação ainda é superior ao de um PC. O sistema operacional empregado pela grande maioria das estações é o UNIX[2]. Elas também possuem uma interface gráfica que facilita o uso não só do sistema operacional como também dos aplicativos. O sistema gráfico adotado pela totalidade de fabricantes é o X Window, desenvolvido para trabalhar em redes [Nye 90, Nye 92].

O presente trabalho consiste na implantação de um sistema de simulação em estações de trabalho sob o sistema operacional UNIX. O trabalho foi desenvolvido de modo a utilizar ao máximo os recursos em termos de capacidade de memória e desempenho gráfico do equipamento.

[1] DOS é marca registrada de Microsoft, Inc.
[2] UNIX é marca registrada da AT&T.

### 2.5.2 Software

#### 2.5.2.1 Sistema operacional

A característica mais uniforme das estações de trabalho atualmente disponíveis é o sistema operacional empregado. Praticamente todas empregam alguma versão do sistema operacional UNIX. Há pouca discussão entre os fabricantes quanto ao sistema operacional a ser usado; todavia, há discordância quanto à versão a ser utilizada. Os fabricantes nos últimos anos se agruparam em torno de duas associações, a OSF (Open Software Foundation) e a UI (UNIX International). A OSF é formada pela IBM, HP, Digital, entre outros; a UI é formada pricipalmente pela Sun e AT&T. Ambas as associações de fabricantes adotaram uma versão do UNIX e um padrão de interface gráfica. Embora a aparência e funcionamento das interfaces gráficas não sejam idênticas, todas elas são elaboradas dentro do padrão X Window[3]. Este padrão, elaborado pelo MIT (Massachussetts Institute of Technology) e aprovado em 1989, é adotado atualmente pela grande maioria dos fabricantes de estações.

#### 2.5.2.2 Linguagem de programação

Por muitos anos, de fato, a linguagem de programação C foi a versão fornecida com o sistema operacional UNIX. Porém, por não existir nenhum padrão, havia discrepâncias. Para remediar essa situação, o ANSI estabeleceu, em 1983, um comitê para criar um padrão que definiria de uma vez por todas a linguagem C.

A idéia inicial na criação do C foi a simplicidade com o objetivo de assegurar a eficiência, uma vez que ele se destinava à criação de sistemas. Ele suporta como tipos básicos aqueles que são suportados por hardwares típicos. Contudo, é possível criar tipos mais complexos tais como estruturas e arranjos.

A linguagem C++ foi desenvolvida principalmente para facilitar o gerenciamento, a programação e a manutenção de sistemas de *software*. C++ tenta preservar a liberdade e o espírito de C tornando possível criar unidades funcionais tipo caixa preta, que têm acesso rigorosamente controlado. Essas unidades caixa preta são geralmente chamadas de objetos, e C++ é, às vezes, chamada de linguagem orientada por objeto [Pappas 91, Schildt 91].

O paradigma orientado para objeto oferece um novo método de desenvolvimento de software para uma variedade de aplicações. O conceito fundamental do paradigma orientado para

[3] X Window é marca registrada do Massachussetts Institute of Technology.

o objeto é que as soluções dos problemas são implementadas através de envios de mensagens a objetos. Para isto, os objetos de uma solução precisam ser definidos juntamente com as mensagens que cada objeto responderá. Este conceito contrasta com a programação estruturada onde definimos estruturas de dados e as enviamos como parâmetros para procedimentos definidos [Meyer 88, Wiener 91].

A característica mais importante de C++, não encontrado no C normal, é a classe. Uma classe é semelhante a uma estrutura, sob diversos aspectos, e é usada para permitir a criação de objetos. Essencialmente, uma classe permite a criação de um tipo conglomerado definido pelo programador. Na sua definição, podem ser incluídas informações e também podem ser definidas precisamente que funções tem acesso às variáveis.

Em C++, as classes realmente servem como um padrão para a criação de objetos. Os objetos criados são instâncias da classe. Podemos desenvolver uma hierarquia de classe onde há uma classe raiz (ou classe-pai), e várias subclasses (ou classes-filhas). Em C++, a base para a hierarquia de classes são as classes derivadas. Classes-pai representam tarefas mais generalizadas enquanto classes-filhas derivadas recebem tarefas mais específicas.

A herança refere-se à derivação de uma nova classe a partir de uma classe-pai existente. A classe-pai serve como um padrão para a classe derivada e pode ser alterada de muitas formas. Por exemplo, podemos sobrecarregar funções membro, acrescentar novos membros e mudar privilégios de acesso. A herança é um conceito importante, pois permite que reutillizemos uma definição de classe, através de simples mudanças, sem serem necessárias maiores alterações no código.

#### 2.5.2.3 Interface gráfica

Na implantação da interface gráfica utilizou-se um pacote de rotinas gráficas chamado XView[4]. O XView é um *toolkit* que serve para desenvolver aplicações gráficas interativas que rodam sobre o sistema X Window. Ele fornece um conjunto de funções escritas em linguagem C que permitem criar, manipular e destruir objetos gráficos, tais como menus, botões e painéis. A aparência e funcionalidade destes objetos seguem as especificações do padrão OPEN LOOK[5]. Ele é totalmente baseado na Xlib, biblioteca de mais baixo nível do sistema X Window [Heller 90, Nye 90, Nye 92].

[4] XView é marca registrada da Sun Microsystems, Inc.
[5] OPEN LOOK é marca registrada da AT&T

Para o uso eficiente deste *toolkit* é necessário o conhecimento da programação orientada para objeto (POO), uma vez que ele é organizado dentro destes princípios. Além disso, seu uso não exclui o uso de funções da Xlib.

O XView permite ao programador construir uma interface gráfica sem ter que se envolver com muitos dos detalhes acerca do sistema de janelas no qual ele trabalha. Este sistema também é extensível, ou seja, o programador pode incluir outras classes no mesmo, bem como construir classes a partir das já existentes [Cox 86].

## 2.6 Conclusões

Neste capítulo apresentaram-se alguns conceitos relacionados com a modelagem e simulação de sistemas. Estes conceitos são importantes porque ajudaram a definir com maior clareza os requisitos que o ambiente deve satisfazer.

Em função destes requisitos, foram tomadas as primeiras decisões quanto aos recursos computacionais a serem utilizados para a implementação do ambiente de simulação. Deve-se dizer também que, os recursos utilizados talvez não sejam os mais adequados, mas dentro do que se dispunha, mostraram ser as melhores opções.

# CAPÍTULO 3: IMPLEMENTAÇÃO

## 3.1 Introdução

A especificação do ambiente proposto descrito no capítulo anterior, define o esqueleto da implementação deste pacote computacional denominado SPISC (Simulador de Processos Industriais e Sistemas de Controle). Este pacote é um ambiente integrado porque utiliza uma biblioteca de tipos abstratos de dados relativa à área de controle clássico (tipos tais como função de transferência e variáveis de estado).

Em função de experiências anteriores no desenvolvimento de pacotes de simulação tais como SAPIC (Sistema de Análise e Projeto Integrado por Computador) [Savi 87], SDPID [Caetano 89], SIMAP (Sistema de Identificação, Modelagem, Análise e Projeto de Sistemas de Controle Associados a Sistemas Elétricos de Potência) [Da Luz 90], ISAC (Identificação de Sistemas Assistido por Computador) [Lima 91] e SADECA (Sistema de Avaliação de Desempenho de Controladores Adaptativos) [Kammer 92], pode-se afirmar que certas características como flexibilidade e modularidade de código-fonte devem ser priorizadas, além do fator tempo, durante o desenvolvimento de um programa. Estas características começam a ser delineadas na própria especificação do ambiente.

É de suma importância ressaltar que os detalhes relativos à utilização deste ambiente estão descritos no manual do usuário (ver anexo), não cabendo neste capítulo repetir estas informações, mas dar uma idéias das potencialidades implementadas.

Neste capítulo apresentam-se as principais características das estruturas adotadas para o bloco e o modelo, assim como os algoritmos de integração numérica implementados para a simulação propriamente dita.

Apresenta-se também a interface homem-máquina, e alguns comentários justificando o uso do pacote gráfico XView.

## 3.2 Estrutura do ambiente

A estrutura interna deste pacote computacional é formado por um conjunto de classes. Estas classes são responsáveis pela criação da estrutura do modelo, pela modificação dos parâmetros do modelo, pela construção de super-blocos, pela simulação no tempo do sistema

modelado, pela análise freqüencial do modelo linearizado, pela representação gráfica dos resultados, etc.

A seguir, faz-se uma breve explanação das principais características do pacote, e apresentam-se os fundamentos matemáticos utilizados na implementação dos algoritmos de integração numérica (método trapezoidal com amortecimento), transformação de diagrama de blocos para variáveis de estado (método do modelo incremental) e transformação de variáveis de estado para matriz de transferência (método de Fadeev-Leverrier [Chen 84]).

### 3.2.1 Estrutura do modelo

Como já foi dito no capítulo anterior, um sistema pode ser organizado num conjunto de blocos (figura 3.1). Este sistema, como um todo, pode ser descrito em termos das interconexões entre os blocos. Em outras palavras, o sistema pode ser modelado graficamente como um diagrama de blocos.

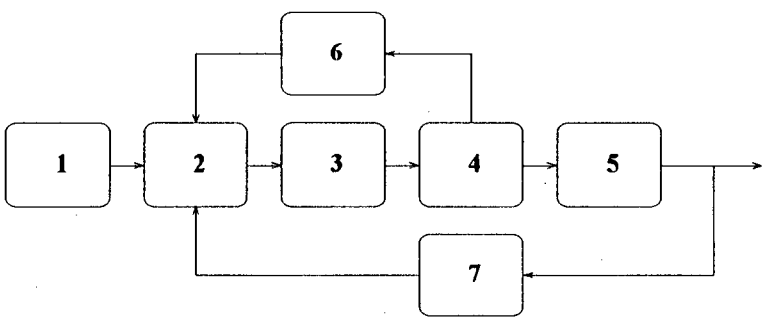



**Figura 3.1** - Diagrama de blocos

Este tipo de modelo é prático, pois simplifica a especificação das interações dentro do ambiente. Além disso, requisitos importantes tais como facilidade de acesso aos elementos da estrutura para leitura ou modificação, otimização de memória, construção ou modificação dinâmica de estruturas e armazenamento de dados são satisfeitos.

Este modelo foi adotado para o pacote, sendo que ele integra processo e controladores de forma integrada. Para a construção do modelo, está disponível uma biblioteca com um conjunto básico de blocos que permitem a representação de muitos fenômenos associados a processos industriais.

O modelo pode ser construído a partir dos blocos elementares contidos na biblioteca interna do pacote e dos super-blocos construídos por usuários (figuras 3.2 e 3.3).

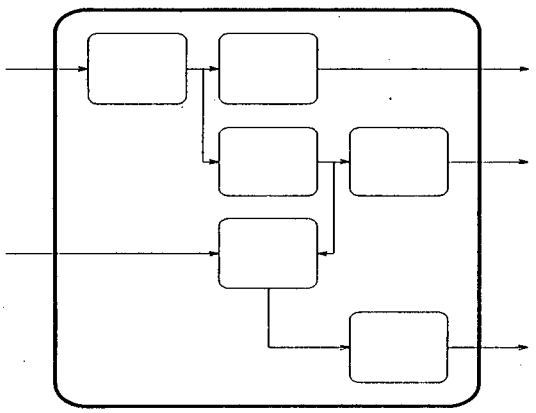


**Figura 3.2** - Diagrama de blocos de um super-bloco com 2 entradas e 3 saídas, composto por blocos elementares.

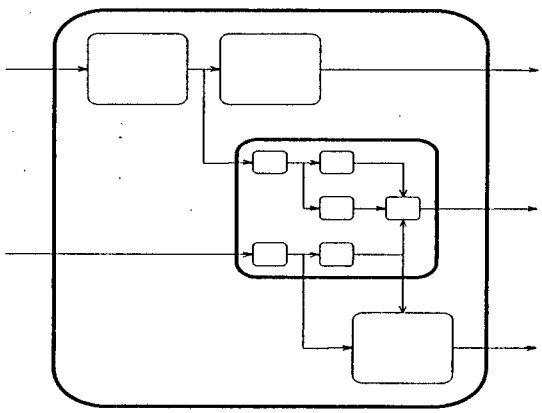


**Figura 3.3** - Diagrama de blocos de um super-bloco com 2 entradas e 3 saídas,composto por blocos elementares e um super-bloco.

A estrutura do bloco constitui-se do número, natureza, nome, blocos cujas saídas lhe servem de entrada, condição inicial, valores de entrada, valor de saída e parâmetros. A estrutura do modelo constitui-se de um vetor de tamanho *n*, onde *n* é o número de blocos que compõem o mesmo.

Para a simulação, implementou-se um algoritmo auxiliar com o propósito de criar uma lista ligada a partir do vetor de blocos constituintes do modelo. Esta lista representa a seqüência em que os blocos serão simulados. Como esta seqüência não existe em todos os casos, cabe ao algoritmo validar ou não o modelo declarado.

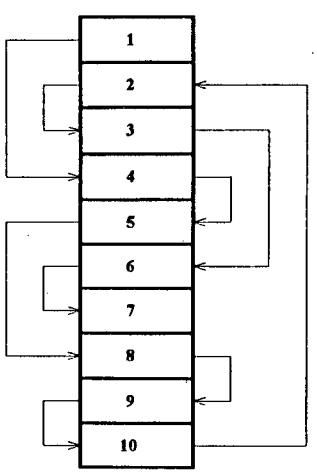



**Figura 3.4** - Lista ligada formada a partir de um vetor de 10 blocos.

Caso o modelo seja validado pelo algoritmo de ordenamento de blocos, pode-se passar à simulação, utilizando um dos métodos de integração numérica; caso contrário, será exibida uma mensagem de erro notificando a presença de um *loop* algébrico.

### 3.2.2 Algoritmos de integração

Para simular os modelos declarados, o usuário dispõe de dois métodos de integração numérica. Estes são o método trapezoidal amortecido e o método de Runge-Kutta de 4ª ordem. O usuário em ambos os casos, define o intervalo de simulação assim como o passo de integração. Não se optou pelo passo variável, que apesar de ser mas veloz, em determinadas circunstâncias o algoritmo de integração diverge, como ocorre no SIMNON[1].

#### 3.2.2.1 Trapezoidal com amortecimento

Descreve-se aquí uma variação do método de integração trapezoidal[2] para a solução de equações dinâmicas. Este método possui várias vantagens em relação à integração trapezoidal para a simulação de transitórios em processos industriais. A integração trapezoidal com amortecimento está livre de problemas numéricos, mesmo quando usado como um derivador. Ele é completamente compatível com a integração trapezoidal.

As principais características da integração trapezoidal são sua simplicidade, estabilidade numérica para sistemas com autovalores muito distintos e sua natureza auto-iniciante. Por outro lado, ela sofre de oscilações numéricas quando usada como um derivador. De modo a eliminar o problema de oscilações numéricas permanentes na diferenciação, a seguinte equação foi indiretamente introduzida [Alvarado 83]:

$$y_n = y_{n-1} + \frac{\Delta t}{2}[(1+\alpha)\dot{y}_n + (1-\alpha)\dot{y}_{n-1}] \qquad (3.1)$$

Observe que para $\alpha = 0$, o método resulta na integração trapezoidal, enquanto que para $\alpha = 1$, resulta no método de integração retangular-regressivo de Euler.

Qualquer método de integração é relevante apenas para equações dinâmicas. No caso do pacote SPISC, equações dinâmicas estão associadas somente aos blocos do tipo dinâmica linear contínua.

Abaixo mostra-se o desenvolvimento de uma expressão matemática que pode facilmente ser implementada em um simulador. O ponto de partida é a expressão da saída de um bloco do tipo dinâmica linear contínua com condições iniciais nulas.

---

[1] SIMNON é marca registrada do Department of Automatic Control, Lund, Sweden e eum produto do SSPA.
[2] Integração trapezoidal refere-se, na verdade, à integração trapezoidal ordinária.

$$\frac{y(s)}{u(s)} = \frac{a_0 + a_1 s + \ldots + a_{m-1} s^{m-1} + a_m s^m}{b_0 + b_1 s + \ldots + b_{n-1} s^{n-1} + b_n s^n} \tag{3.2}$$

Como já foi dito anteriormente, o pacote utiliza uma biblioteca de tipos abstratos de dados. Um destes tipos é a função de transferência (classe *trfunct*). O usuário pode construir uma instância da classe tanto na forma polinomial como na forma fatorada. Independente do construtor utilizado, a classe oferece ao usuário funções de acesso aos coeficientes de ambas as representações [Suguieda 93].

Outro tipo abstrato de dado utilizado pelo pacote é a matriz de números reais de dupla precisão (classe *dmatrix*). Isto permite expressar as equações a seguir na forma matricial.

A representação por variáveis de estado associada à função de transferência (3.2) é

$$\dot{\bar{x}} = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ -b_0 & -b_1 & -b_2 & \ldots & -b_{n-1} \end{bmatrix} \bar{x} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} a_0 & a_1 & a_2 & \ldots & a_{n-1} \end{bmatrix} \bar{x} + d^* u \tag{3.3}$$

Combinando as expressões (3.1) e (3.3) tem-se

$$\bar{x}_n = \bar{x}_{n-1} + \frac{\Delta t}{2}[(1+\alpha)\dot{\bar{x}}_n + (1-\alpha)\dot{\bar{x}}_{n-1}] \tag{3.4}$$

O valor de $\bar{x}$ nos instantes **n** e **n-1** são, respectivamente

$$\dot{\bar{x}}_n = \mathbf{A}\bar{x}_n + \mathbf{b}u_n \tag{3.5a}$$

$$\dot{\bar{x}}_{n-1} = \mathbf{A}\bar{x}_{n-1} + \mathbf{b}u_{n-1} \tag{3.5b}$$

Substituindo (3.5) em (3.4), tem-se

$$\bar{x}_n = \bar{x}_{n-1} + \frac{\Delta t}{2}(1+\alpha)[\mathbf{A}\bar{x}_n + u_n] + \frac{\Delta t}{2}(1-\alpha)[\mathbf{A}\bar{x}_{n-1} + \mathbf{b}u_{n-1}] \tag{3.6}$$

$$[\mathbf{I}_n - \frac{\Delta t}{2}(1+\alpha)\mathbf{A}]\bar{x}_n = [\mathbf{I}_n + \frac{\Delta t}{2}(1-\alpha)\mathbf{A}]\bar{x}_{n-1} + \frac{\Delta t}{2}(1+\alpha)u_n + \frac{\Delta t}{2}(1-\alpha)\mathbf{b}u_{n-1} \quad (3.7)$$

$$\bar{x}_n = [\mathbf{I}_n - \frac{\Delta t}{2}(1+\alpha)\mathbf{A}]^{-1}\left\{[\mathbf{I}_n + \frac{\Delta t}{2}(1-\alpha)\mathbf{A}]\bar{x}_{n-1} + \frac{\Delta t}{2}(1+\alpha)\mathbf{b}u_n + \frac{\Delta t}{2}(1-\alpha)\mathbf{b}u_{n-1}\right\} \quad (3.8)$$

$$y_n = \mathbf{c}\bar{x}_n + \mathbf{d}u_n \quad (3.9)$$

#### 3.2.2.2 Runge-Kutta de 4ª ordem

Este método de integração está implementado na biblioteca de tipos abstratos de dados. Este método é chamado quando se acessa a função *timeresp* sobre um objeto do tipo função de transferência [Suguieda 93].

### 3.2.3 Transformações auxiliares

Com o propósito de permitir a análise clássica sobre um sistema multivariável e não-linear, faz-se necessário transformar o modelo do sistema representado por um diagrama de blocos numa representação linear. Para isto, a critério do usuário, o sistema pode ser linearizado num determinado ponto de operação. Pode-se dizer então, que o sistema comporta-se linearmente em torno deste ponto de operação.

Apresentam-se a seguir, os algoritmos utilizados para a obtenção da equação dinâmica e da matriz de transferência.

#### 3.2.3.1 Obtenção da equação dinâmica

Para obter-se a representação por variáveis de estado de um sistema multivariável não-linear estacionário, modelado por diagrama de blocos, faz-se uso do modelo incremental.

O algoritmo consiste basicamente em obter os coeficientes das matrizes A, B, C e D a partir da aplicação de um pequeno distúrbio, em torno do ponto de operação, em cada variável de entrada e de estado do modelo, verificando sua repercussão em cada derivada de estado e variável de saída, respectivamente.

Considerando a equação dinâmica

$$\dot{\bar{x}} = \mathbf{A}\bar{x} + \mathbf{B}\bar{u}$$
$$y = \mathbf{C}x + \mathbf{D}\bar{u} \qquad \textbf{(3.10)}$$

tem-se $A_{n*n}$, $B_{n*ne}$, $C_{ns*n}$ e $D_{ns*ne}$, onde **n** é o número de estados, **ne** o número de entradas e **ns** o número de saídas do modelo linear.

Os coeficientes da matriz A são dados por

$$\mathbf{a_{i,j}} = \frac{\Delta \dot{x}_i}{\Delta x_j}, \quad \mathbf{i,j = 1,2,\ldots,n} \qquad \textbf{(3.11)}$$

onde:

$\Delta x_j$ é o degrau incremental sobre a variável de estado $x_j$
$\Delta x_k$=0, k≠j, k=1,2,...,n
$\Delta u_k$=0, k=1,2,...,ne

Os coeficientes da matriz B são dados por

$$\mathbf{b_{i,j}} = \frac{\Delta \dot{x}_i}{\Delta u_j}, \quad \mathbf{i = 1,2,\ldots,n\ j = 1,2,\ldots,ne} \qquad \textbf{(3.12)}$$

onde:

$\Delta u_j$ é o degrau incremental sobre a entrada $u_j$,
$\Delta u_k$=0, k≠j, k=1,2,...,ne
$\Delta x_k$=0, k=1,2,...,n

Os coeficientes da matriz C são dados pòr

$$\mathbf{c_{i,j}} = \frac{\Delta y_i}{\Delta x_j}, \quad \mathbf{i = 1,2,\ldots,ns,\ j = 1,2,\ldots,n} \qquad \textbf{(3.13)}$$

onde:

$\Delta x_j$ é o degrau incremental sobre a variável de estado $x_j$
$\Delta x_k$=0, k≠j, k=1,2,...,n

$\Delta u_k$=0, k=1,2,...,ne

Os coeficientes da matriz D são dados por

$$\mathbf{d_{i,j} = \frac{\Delta y_i}{\Delta u_j}, \quad i = 1,2,\ldots,ns,\ j = 1,2,\ldots,ne} \tag{3.14}$$

onde:

$\Delta u_j$ é o degrau incremental sobre a entrada $u_j$,
$\Delta u_k$=0, k≠j, k=1,2,...,ne
$\Delta x_k$=0, k=1,2,...,n

O algoritmo realiza o cálculo dos coeficientes acima referidos sobre o modelo declarado para simulação. Para sua execução, é necessário que os modelos estejam completamente estabilizados em torno do ponto de operação desejado, isto é, que todas as derivadas sejam nulas.

### 3.2.3.2 Obtenção da matriz de transferência

Para obter-se a matriz de transferência a partir de uma equação dinâmica, implementou-se o algoritmo de Fadeev-Leverrier [Chen 84]. Este algoritmo permite que se obtenha cada uma das funções de transferência existentes em um determinado modelo multivariável e não-linear.

A aplicação deste algoritmo a sistemas de ordem elevada (maior que 10), deve levar em conta o condicionamento das matrizes, pois este fato influencia significativamente a qualidade do resultado. Para melhorar o condicionamento da matriz A, efetuou-se um escalamento, utilizando-se o conceito de medida de matriz [Vidyasagar 78], que resultou numa melhora significativa no desempenho do algoritmo.

O escalamento da matriz A é efetuado como segue:

$$\mathbf{k1 = \min_i \left| a_{ii} + \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \right|} \tag{3.15a}$$

$$k2 = \max_i \left| a_{ii} - \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \right| \tag{3.15b}$$

$$FEA=\max(k1,k2) \tag{3.16}$$

onde FEA é o fator de escalamento da matriz A.

A matriz escalada Â, é dada por

$$\hat{a}_{ij} = \frac{a_{ij}}{FEA}, \qquad i,j = 1,2,\ldots,n \tag{3.17}$$

A matriz de transferência da equação dinâmica (3.10) é obtida como se segue:

$$M(s) = C(sI - \hat{A})^{-1}B + D \tag{3.18}$$

$$(sI - \hat{A})^{-1} = \frac{Adj(sI - \hat{A})}{det(sI - \hat{A})} \tag{3.19}$$

Substituindo (3.19) em (3.18) tem-se

$$M(s) = \frac{1}{det(sI - \hat{A})}[C\,Adj(sI - \hat{A})B + D(sI\_\hat{A})] \tag{3.20}$$

Considerando as expressões

$$F(s) = C\,Adj(sI - \hat{A})B + D(sI - \hat{A}) \tag{3.21}$$

$$Adj(sI - \hat{A}) = E_0 s^{n-1} + E_1 s^{n-2} + \ldots + E_{n-2}s + E_{n-1} \tag{3.22}$$

$$det(sI - \hat{A}) = s^n + a_1 s^{n-1} + \ldots + a_{n-1}s + a_n \tag{3.23}$$

Substituindo (3.22) e (3.23) em (3.21) tem-se

$$F(s) = Ds^n + (CE_0B + a_1D)s^{n-1} + (CE_1B + a_2D)s^{n-2} + \ldots + (CE_{n-2}B + a_{n-1}D)s + (CE_{n-1}B + a_nD) \tag{3.24}$$

O algoritmo para a obtenção de $E_i$, $a_i$ e $F_i$ é constituido dos seguintes passos:

1. $\mathbf{E_0 = I}$
2. $\mathbf{i=1}$
3. $\mathbf{a_i = -(1/i)traço(\hat{A}E_{i-1})}$
4. $\mathbf{F_i = CE_{i-1}B * FEA^{i-1} + a_iD * FEA^i}$
5. $\mathbf{E_i = \hat{A}E_{i-1} + a_iI}$
6. $\mathbf{i=i+1}$
7. $\mathbf{i \leq n}$

Repetem-se os passos 3, 4, 5, 6 e 7 enquanto a condição do passo 7 for satisfeita.

Obtidos os polinômios (3.24) e (3.23), que representam os numeradores e o denominador das funções de transferência, respectivamente, procede-se ao re-escalamento. Para tanto, multiplica-se cada coeficiente $a_i$, pelo fator de escalamento elevado na potência do termo em *s* associado.

Observa-se, a partir da equação (3.20), que o polinômio do denominador das funções de transferência é a própria equação característica da matriz A. Este fato sugere que podem existir cancelamentos entre zeros e pólos destas funções, principalmente no caso multivariável. A classe *trfunct* da biblioteca de tipos abstratos de dados possui mecanismos para estes cancelamentos.

### 3.2.4 Interface homem-máquina

O paradigma de "simplicidade, consistência e eficiência" assim como muitas outras diretrizes funcionais, guiaram a implementação da interface homem-máquina deste pacote computacional [Sun 89a].

A simplicidade é a característica mais importante da aplicação para usuários novatos ou intermitentes. A aplicação dá a impressão de ser simples e intuitiva. Ela inspira confiança para continuar explorando a capacidade do produto.

A consistência permite ao programador aplicar conhecimentos, previamente adquiridos, a novas áreas. O programador pode esboçar um projeto de um produto consistente por inspeção e aprendizagem de padrões semelhantes.

Uma aplicação eficiente minimiza o número de passos necessários para realizar uma operação, além de prover "atalhos". Mesmo usuários iniciantes podem realizar ações com um número mínimo de passos.

Toda a implementação da interface homem-máquina do SPISC, foi realizada com objetos do pacote de interface gráfica XView, descrito em [Heller 90]. Estes objetos obedecem ao padrão OPEN LOOK.

Devido ao pouco tempo disponivel para a realização deste trabalho, não foi possivel submeter o pacote a um β-teste para avaliar as facilidades implementadas.

#### 3.2.4.1 Declaração

O pacote, operando no modo "Declaração", apresenta ao usuário as opções de entrada ("Edita" e "Carrega") e saída ("Grava modelo" e "Grava super-bloco") de dados. Através destas opções, o usuário pode editar seus modelos e/ou super-blocos, armazená-los em arquivo, e recuperá-los posteriormente (figura 3.5).

Para a edição, o usuário dispõe de um editor embutido, onde ele deve declarar as interações entre os blocos e/ou super-blocos. Além disso, o usuário deve especificar para cada bloco características tais como natureza, parâmetros, entradas, etc.

Para a entrada e saída de dados via arquivo, o usuário dispõe de funções de acesso a diretórios e arquivos.

O usuário dispõe também de uma função de leitura de comentários. Estes comentários podem estar contidos dentro do arquivo onde está declarado o respectivo modelo ou em um arquivo independente. Isto é bastante útil, especialmente quando a biblioteca particular de super-blocos e modelos é extensa.


**Figura 3.5** - Janela de apresentação do ambiente, operando por *default* no modo **"Declaração"**.

### 3.2.4.2 Simulação

Ao trocar o modo de operação do sistema, vários painéis e tela de desenho são apagados, cedendo sua área aos novos elementos do modo. Desta forma a simulação faz uso de painéis e telas de desenho distintos aos da declaração, como pode ser observado na figura 3.6.

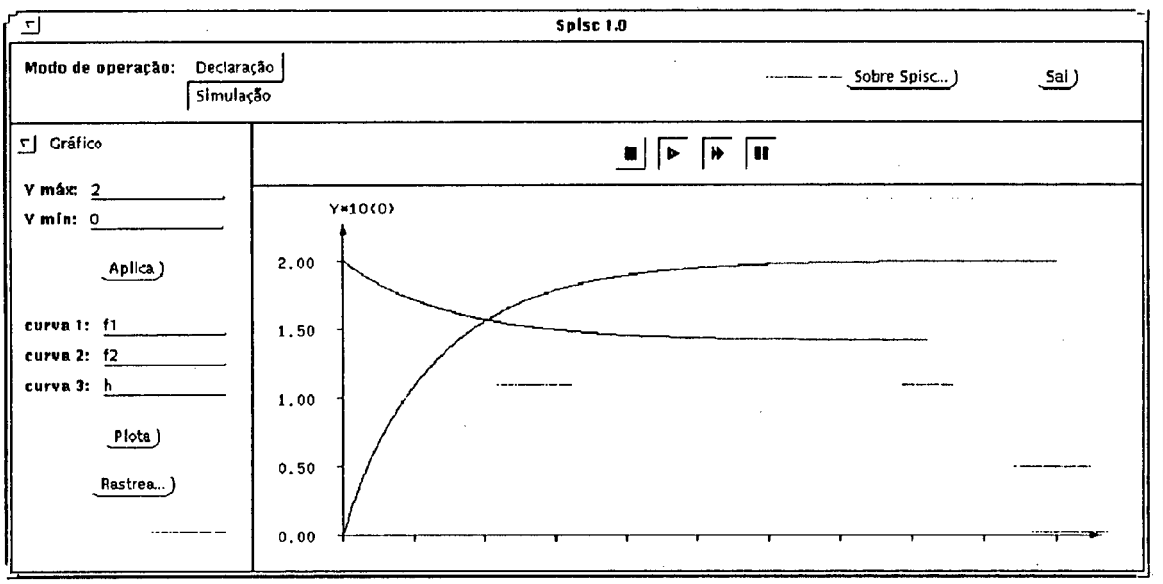



**Figura 3.6** - Representação gráfica da simulação.

O acompanhamento da evolução da simulação é realizado através do gráfico Y-t, no qual são plotadas as saídas dos blocos escolhidos pelo usuário. Estes valores podem ter suas escalas de amplitude alteradas, se necessário. Para facilitar esta operação foram criadas funções que ajustam a escala do gráfico (as escalas das curvas plotadas mudam proporcionalmente).

Outra ação que pode ser realizada sobre o gráfico é o rastreamento, ponto a ponto, das curvas plotadas.

Acima da tela de desenho encontra-se o painel de comando da simulação. Neste painel pode-se observar a presença de botões contendo símbolos semelhantes ao de um toca fitas comum, o que facilita a memorização das operações realizadas por estes [Kammer 92].

Ao lado esquerdo da tela pode-se observar o painel de alteração do gráfico. Na realidade existem mais dois painéis, mas apenas um está visível num determinado instante. Estes painéis destinam-se à alteração do gráfico (escala), bloco (parâmetros) e modelo (condições iniciais).

## 3.3 Conclusões

Declarar um modelo como um diagrama de blocos é uma forma simples de representar graficamente um modelo multivariável e não-linear. Além disso, esta estrutura é realmente simples de manipular (criar, modificar e destruir) computacionalmente, e permite que um conjunto de blocos possa ser agregado de forma tal a representar alguma outra estrutura mais significativa (e complexa).

Graças à biblioteca de tipos abstratos de dados, a implementação dos algoritmos de integração numérica tornaram-se tarefas mais simples, como no caso do método de integração trapezoidal com amortecimento, baseada em cálculos matriciais. Como este método inclui cálculo de matriz inversa, a simulação, utilizando este algoritmo, é as vezes demorada.

A implementação de algoritmos de linearização permite que se apliquem sobre o modelo linear (operando em torno de um ponto de operação) ferramentas clássicas de análise (temporal e freqüencial). Várias destas ferramentas estão implementadas na biblioteca de tipos abstratos de dados; com a linearização, pode-se explorar mais os recursos desta biblioteca.

# CAPÍTULO 4: RESULTADOS

## 4.1 Introdução

Neste capítulo ver-se-ão algumas simulações acerca de processos comumente encontrados na indústria química, assim como algumas demonstrações dos recursos implementados no ambiente. Os resultados obtidos são comparados com os de outros pacotes computacionais já consagrados, tais como SIMNON[1] e MATLAB[2].

## 4.2 Reação exotérmica

Um processo bastante comum na indústria química é a reação exotérmica que ocorre no interior de um reator revestido (figura 4.1). Duas substâncias *A* e *B* reagem de modo a produzir as substâncias *C* e *D* segundo a lei *A+B* --> *C+D*. O fluxo constante do fluido frio que passa através do revestimento entra à temperatuta $T_{ji}$ (40 °C).

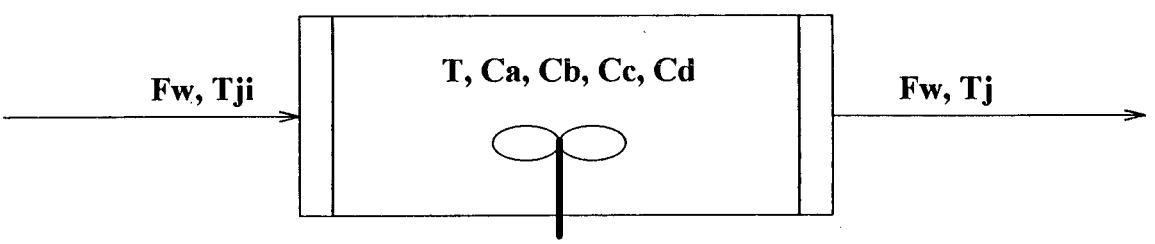



**Figura 4.1** - Reator revestido.

O que procura-se conhecer é a temperatura e a composição da massa fluida, usando os seguintes dados:

a) Volume total de líquido (V): 30 $ft^3$, sem mudança de densidade durante a reação.
b) Carga inicial (m): 30 moles de *A* e 24 moles de *B*.
c) Temperatura inicial da massa fluida: 70 °C.
d) A taxa de reação é de 2ª ordem, proporcional à concentração (moles/$ft^3$) de cada componente. O coeficiente da taxa é $k=2,58*10^5e^{-5000/T(°K)}$ $ft^3$/min/mol.

---

[1] SIMNON (*Simulation language for non-linear systems*) é marca registrada do Department of Automatic Control, Lund, Sweden e eum produto do SSPA.
[2] MATLAB (*Matrix laboratory*) é marca registrada de The MathWorks, Inc.

e) O calor de reação é $H_r$=10,8*$10^4$ PCU/mol de *A* ou *B* reagindo, e a capacidade média calorífica da massa reagente é $C_p$=300 PCU/mol/°C.

f) Assume-se que a temperatura média do revestimento é também a temperatura de saída $T_J$, e a condição inicial é 40 °C. O coeficiente de transferência de calor global entre o revestimento e o conteúdo do reator é UA=40000 PCU/min/°C. A capacidade calorífica da água no revestimento é $C_w$=2000 PCU/°C, e a taxa de fluxo do fluido de resfriamento é $F_w$=6877 lb/min.

Através do balanço de componentes e do balanço de energia, pode-se obter as equações que regem o comportamento do processo. Estas equações são:

$$\frac{dC_A}{dt} = -r, \qquad C_A(0) = 1,0\ \frac{mol}{ft^3} \tag{4.1}$$

$$\frac{dC_B}{dt} = -r, \qquad C_B(0) = 0,8\ \frac{mol}{ft^3} \tag{4.2}$$

$$\frac{dC_C}{dt} = r, \qquad C_C(0) = 0,0\ \frac{mol}{ft^3} \tag{4.3}$$

$$\frac{dC_D}{dt} = r, \qquad C_D(0) = 0,0\ \frac{mol}{ft^3} \tag{4.4}$$

$$\frac{dT}{dt} = \frac{r \cdot V \cdot H_r - Q}{m \cdot C_p}, \qquad T(0) = 70\ ^\circ C \tag{4.5}$$

$$\frac{dT_J}{dt} = \frac{Q + F_w \cdot C_{pw}(T_{Ji} - T_J)}{C_w}, \qquad T_J(0) = 40\ ^\circ C \tag{4.6}$$

onde:

$$C_{pw} = 1\ PCU/lb/^\circ C$$

$$r = k \cdot C_A \cdot C_B \tag{4.7}$$

$$Q = UA(T - T_J) \tag{4.8}$$

Estas equações, introduzidas no ambiente na forma de diagrama de blocos, foram simuladas e obtiveram-se as seguintes respostas (figura 4.2 e 4.3):

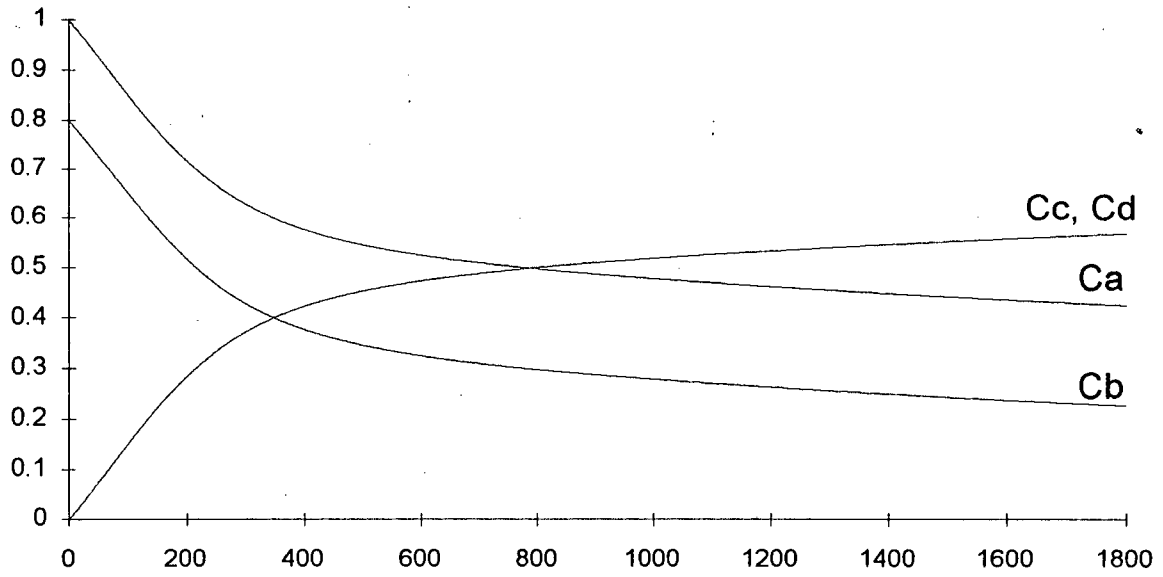



**Figura 4.2** - Comportamento das concentrações das substâncias *A*, *B*, *C* e *D* segundo SPISC.

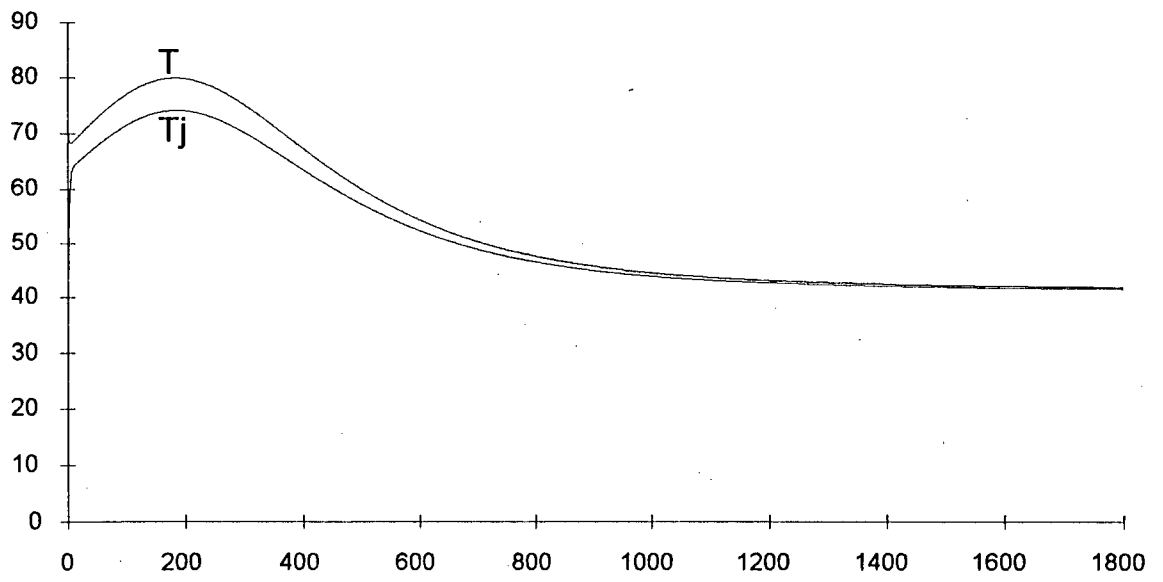



**Figura 4.3** - Comportamento das temperaturas T e $T_J$ segundo SPISC.

Como pode-se observar na figura 4.2, a concentração de *A* no volume era de 1 mol/ft³ no instante t=0. Esta concentração tende para um valor de regime estacionário igual a 0,2 mol/ft³. Isto deve-se de fato ao excesso de moles da substância *A* em relação a *B*, a qual reagiu na sua totalidade. Assim, conforme a lei da reação, 24 moles de *A* e 24 moles de *B* reagiram para formar 24 moles de *C* e 24 moles de *D*. Há um excesso de 6 moles de *A*. As concentrações finais podem ser obtidas pela divisão das respectivas massas molares e o volume total de líquido (V).

A reação em estudo é um processo exotérmico, isto é, libera calor. Por isso, a temperatura da massa fluida aumenta durante a reação (figura 4.3). Para resfriar o líquido, o reator possui um revestimento pelo qual flui água fria. Assim, como pode-se observar na figura 4.3, a temperatura

da massa fluida T no interior do reator aumenta durante a reação, para depois diminuir lentamente devido ao resfriador. A temperatura da água $T_J$ também aumenta já que ela absorve a energia liberada pela reação, para depois diminuir. Aa temperaturas T e $T_J$ tendem ao valor $T_{Ji}$=40 °C, que é a temperatura de entrada do líquido frio.

O processo da figura 4.1 pode ser modificado de modo a produzir as substâncias *C* e *D* de forma contínua. Algumas modificações são propostas na figura 4.4.

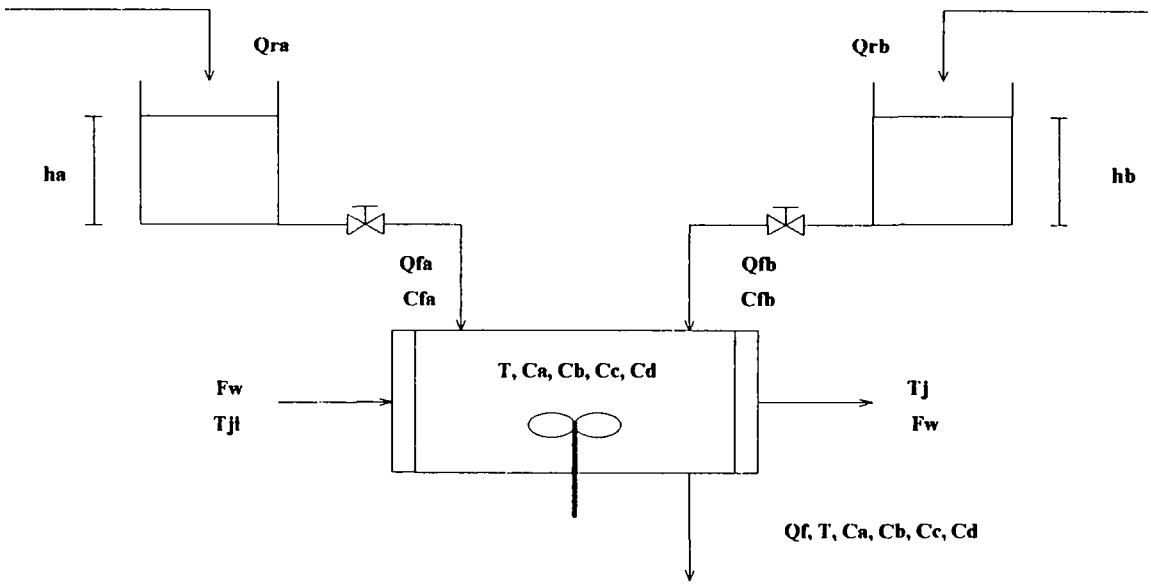



**Figura 4.4** - Reator alimentado por dois tanques de vazão.

Pode-se observar na figura 4.4 que o tanque da esquerda é alimentado com a substância *A*. Este tanque por sua vez alimenta o reator com esta mesma substância. Analogamente, o tanque da direita alimenta o reator com a substância *B*.

Como no caso do reator isolado, procura-se conhecer a temperatura e a composição da massa fluida, usando os seguintes dados:

a) Vazão de entrada do tanque A ($Q_{RA}$): 0,05 ft³/min.
b) Concentração de *A* em $Q_{RA}$: 2 mol/ft³.
c) Vazão de entrada do tanque B ($Q_{RB}$): 005 ft³/min.
d) Concentração de *B* em $Q_{RB}$: 1,6 mol/ft³.
e) Nível do tanque A ($h_A$): 0,0025 ft.
f) Nível do tanque B ($h_B$): 0,000625 ft.
g) Volume total de líquido (V): 30 ft³, sem mudança de densidade durante a reação.
h) Carga inicial (m): 30 moles de *A* e 24 moles de *B*.
i) Temperatura inicial da massa fluida: 40 °C.
j) Temperatura de entrada do líquido resfriador ($T_{Ji}$): 40 °C.

k) A taxa de reação é de 2ª ordem, proporcional à concentração (moles/ft$^3$) de cada componente. O coeficiente da taxa é $k=2,58*10^5e^{-5000/T(°K)}$ ft$^3$/min/mol.

l) O calor de reação é $H_r=10,8*10^4$ PCU/mol de *A* ou *B* reagindo, e a capacidade média calorífica da massa reagente é $C_p=300$ PCU/mol/°C.

m) Assume-se que a temperatura média do revestimento é também a temperatura de saída $T_J$, e a condição inicial é 40 °C. O coeficiente de transferência de calor global entre o revestimento e o conteúdo do reator é UA=40000 PCU/min/°C. A capacidade calorífica da água no revestimento é $C_w=2000$ PCU/°C, e a taxa de fluxo do fluido de resfriamento é $F_w=6877$ lb/min.

n) Assume-se que a pressão sobre os líquidos é igual à pressão atmosférica (psia).

o) A vazão de saída do reator ($Q_F$) é igual à soma das vazões de entrada do restor ($Q_{FA}+Q_{FB}$).

Podem-se obter as equações de estado do sistema através dos balanços de massa, componentes e energia.

Para o tanque A:

$$\frac{dh_A}{dt} = \frac{Q_{RA} - Q_{FA}}{A_A}, \tag{4.9a}$$

$$\frac{dh_A}{dt} = \frac{Q_{RA} - C_{VA}\sqrt{\frac{\Delta P}{G_A}}}{A_A}, \tag{4.9b}$$

$$\frac{dh_A}{dt} = \frac{Q_{RA} - C_{VA}\sqrt{\frac{\rho g h_A}{144 g_c G_A}}}{A_A}, \tag{4.9c}$$

$$\frac{dh_A}{dt} = \frac{Q_{RA} - C'_{VA}\sqrt{h_A}}{A_A}, \qquad h_A(0) = 0,0025 \text{ ft} \tag{4.9d}$$

onde:

$A_A$ (Área da seção transversal do tanque A)=0,5 ft$^2$;
$C_{VA}$ (coeficiente da válvula), em ft$^3$/min/psi$^{1/2}$;
$\Delta P$ (queda de pressão através da válvula), em psi;
$G_A$ (gravidade específica do líquido fluido através da válvula), adimensional;
$\rho$ (densidade do líquido), em lbm/ft$^3$

g (aceleração devido à gravidade)=32,2 ft/sec$^2$;
gc (fator de conversão)=32,2 lbm-ft/lbf-sec$^2$;
$C'_{VA}$ (simplificação das constantes)=1

Para o tanque B (por analogia com o tanque A):

$$\frac{dh_B}{dt} = \frac{Q_{RB} - C'_{VB}\sqrt{h_B}}{A_B}, \qquad h_B(0) = 0,000625 \text{ ft} \qquad (4.10)$$

onde:

$A_B$ (Área da seção transversal do tanque B)=1 ft$^2$
$C'_{VB}$=2

Para o reator:

$$\frac{dC_A}{dt} = -r + (C_{FA} - C_A)\frac{Q_F}{V}, \qquad C_A(0) = 1,0 \ \frac{mol}{ft^3} \qquad (4.11)$$

$$\frac{dC_B}{dt} = -r + (C_{FB} - C_B)\frac{Q_F}{V}, \qquad C_B(0) = 0,8 \ \frac{mol}{ft^3} \qquad (4.12)$$

$$\frac{dC_C}{dt} = r - C_C\frac{Q_F}{V}, \qquad C_C(0) = 0,0 \ \frac{mol}{ft^3} \qquad (4.13)$$

$$\frac{dC_D}{dt} = r - C_D\frac{Q_F}{V}, \qquad C_D(0) = 0,0 \ \frac{mol}{ft^3} \qquad (4.14)$$

$$\frac{dT}{dt} = \frac{r \cdot V \cdot H_r - Q + (C_{AF} + C_{BF})Q_F \cdot C_p(T_F - T)}{m \cdot C_p}, \qquad T(0) = 40\ ^\circ C \qquad (4.15)$$

$$\frac{dT_J}{dt} = \frac{Q + F_w \cdot C_{pw}(T_{Ji} - T_J)}{C_w}, \qquad T_J(0) = 40\ ^\circ C \qquad (4.16)$$

onde:

$C_{pw} = 1 \text{ PCU}/\text{lb}/^\circ C$
$r = k \cdot C_A \cdot C_B$ (4.17)

$$Q = UA(T - T_J) \qquad (4.18)$$

Estas equações, para serem simuladas no ambiente, devem ser introduzidas na forma de blocos. Perceba-se que neste caso podem ser criados super-blocos para depois integrá-los e formar o modelo da figura 4.4. Os diagramas de blocos associados aos tanques e ao reator estão representados nos diagramas 4.1 e 4.2.

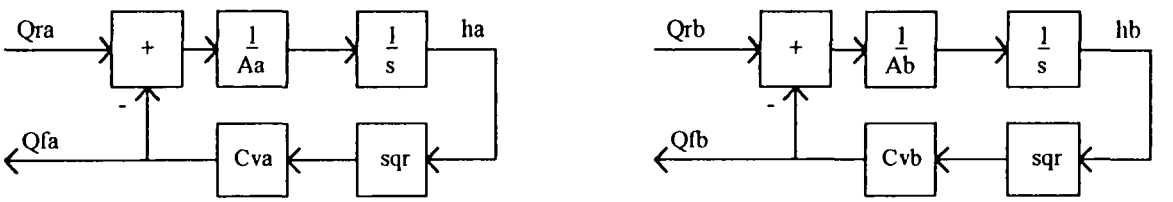



**Diagrama 4.1** - Diagrama de blocos associados aos tanques A e B.

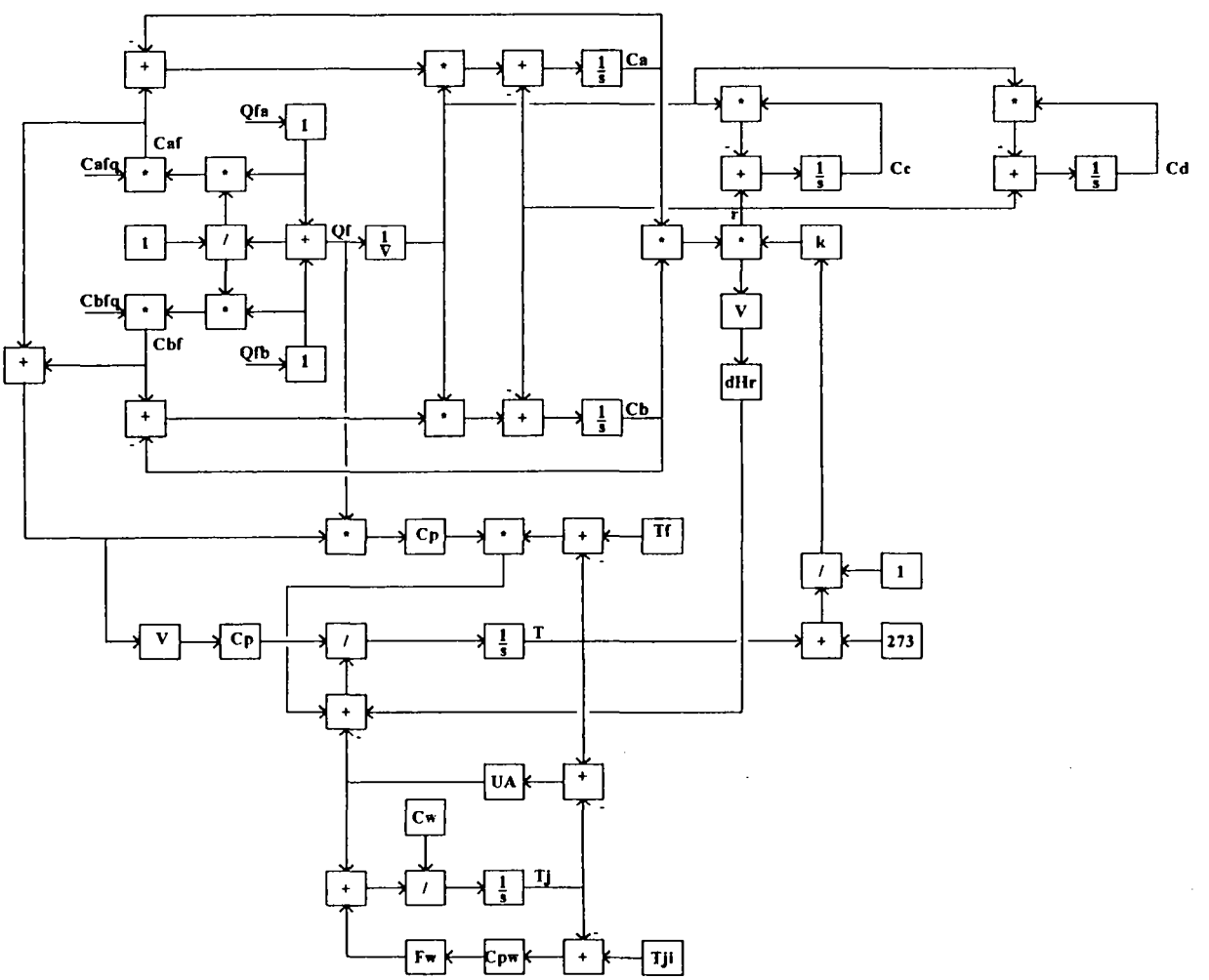



**Diagrama 4.2** - Diagrama de blocos associado ao reator.

No diagrama 4.3 pode-se observar como construiu-se o modelo original a partir das definições dos super-blocos.

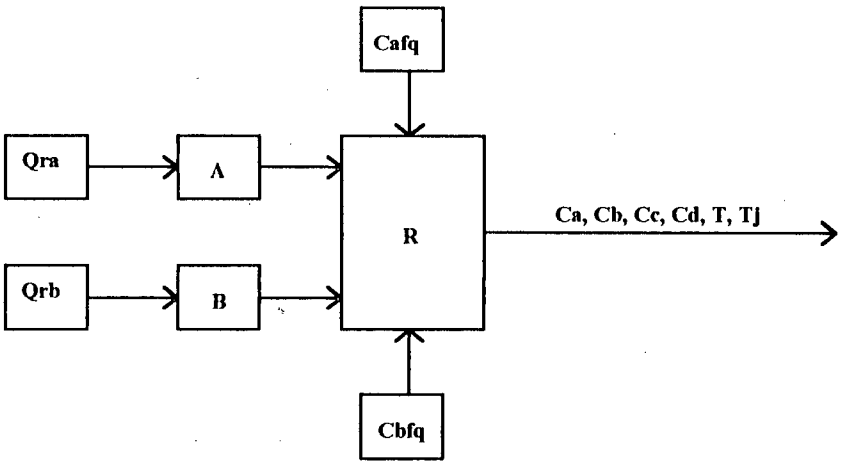



**Diagrama 4.3** - Modelo composto por tanque A (A), tanque B (B) e reator (R).

Simulando estas equações, obtiveram-se os gráficos das figuras 4.4 e 4.5.

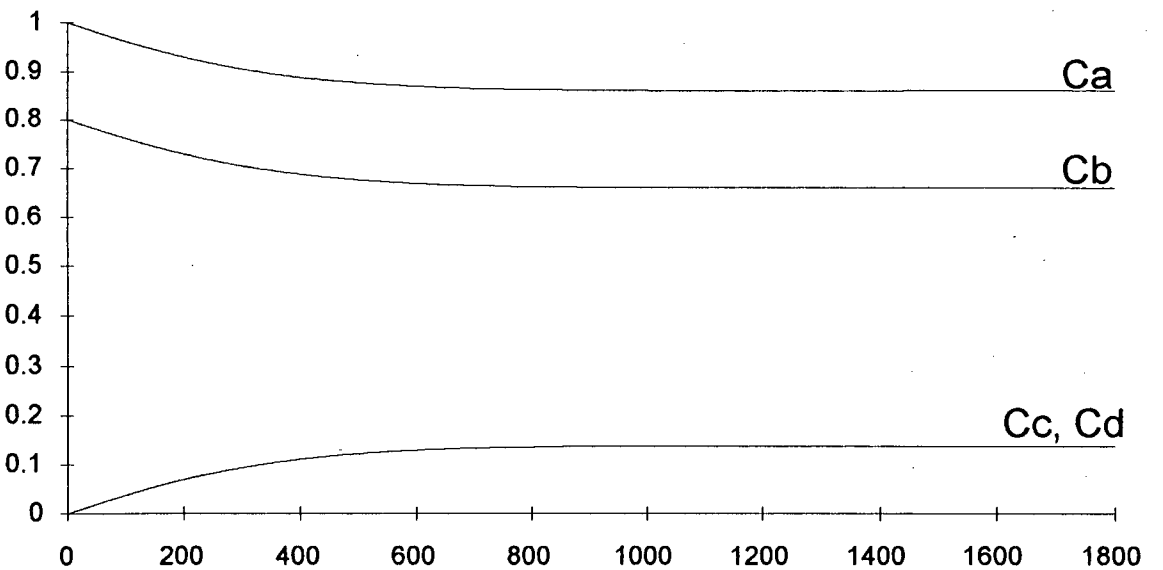



**Figura 4.5** - Comportamento das concentrações das substâncias *A*, *B*, *C* e *D* segundo SPISC. Neste caso definiu-se o tempo de simulação em 1800 s e o passo de integração em 100 ms, utilizando o método de Runge-Kutta de 4ª ordem.

Como definiu-se, as vazões de entrada nos tanques são iguais, e a concentração de *A* é maior que a concentração de *B* nos respectivos fluxos de entrada. Assim, pode-se dizer que a massa molar de *A* no interior do reator é maior que a de *B*.

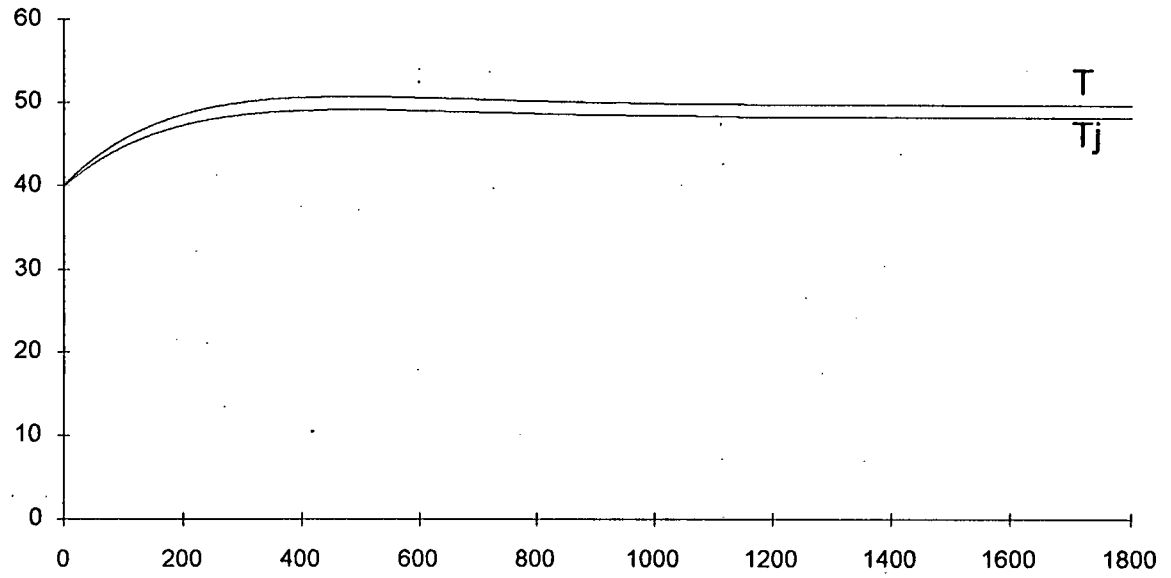



**Figura 4.6** - Comportamento das temperaturas T e $T_J$ segundo SPISC. Nesta simulação definiu-se o tempo de simulação em 1800 s e o passo de integração em 100 ms, utilizando o método de Runge-Kutta de 4ª ordem.

Viu-se no caso do reator isolado, que a concentração de *B* (que é menor que a de *A*) cai para zero. Neste caso, mesmo com uma massa molar menor, a concentração de *B* não cai para zero já que a reação não é imediata e o reator está continuamente sendo alimentado por estas duas substâncias. Este comportamente pode ser observado na figura 4.4.

Analogamente, a concentração de A não chega a 0,8 mol/ft$^3$ como no caso do reator isolado devido às razões explanadas no parágrafo anterior.

Então, como as duas substâncias estão sempre presentes no interior do reator, a reação é contínua, isto é, a produção de C e D é constante. Assim, energia está sempre sendo liberada pela reação o que aumenta a temperatura da massa fluida (T) e do líquido refrigerante (TJ), fazendo com que estas temperaturas não diminuam até a temperatura de entrada do líquido frio, mas estabilizem em 49,8 °C e 48,4 °C respectivamente.

### 4.2.1 Comparação de resultados

Um pacote computacional bastante utilizado (e comercializado) é o SIMNON (*Simulation language for non-linear systems*). Este pacote consegue, entre outros recursos, simular sistemas multivariáveis e não-lineares. O que se pretende nesta seção é comparar os resultados obtidos no SPISC com os obtidos no SIMNON.

Nas simulações realizadas no SIMNON, selecionou-se o método de integração numérica de Euler, utilizando os mesmos intervalos de simulação e os mesmos passos de integração

adotados nas simulações realizadas no SPISC. Assim, obtiveram-se os gráficos representados nas figuras 4.7 e 4.8.

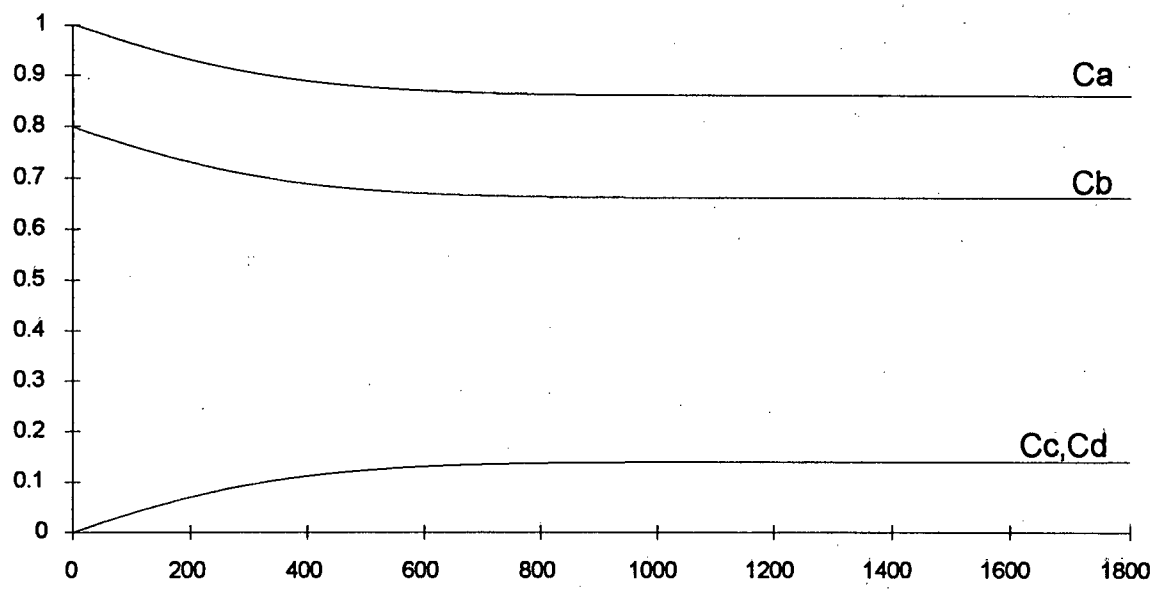



**Figura 4.7** - Comportamento das concentrações das substâncias *A*, *B*, *C* e *D* segundo SIMNON.

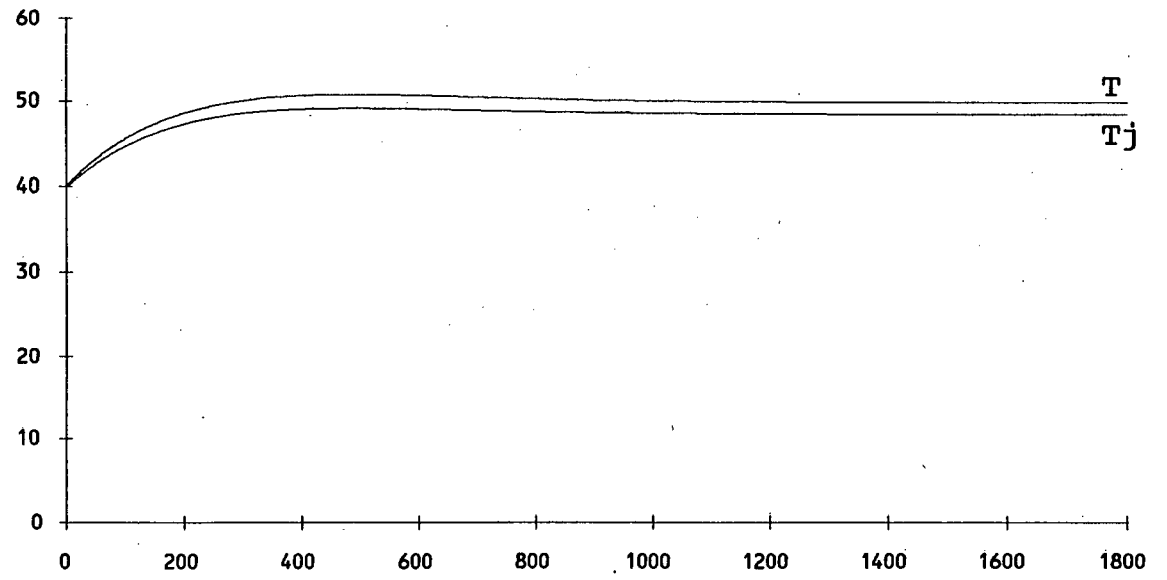



**Figura 4.8** - Comportamento das temperaturas T e $T_J$ segundo SIMNON.

Pode-se ver que as figuras 4.5 e 4.7 possuem formas semelhantes, assim como as figuras 4.6 e 4.8. Para comparar melhor estas curvas, faz-se a sobreposição delas. Assim, na figuras 4.9 pode-se observar as curvas das concentrações obtidas no SPISC e no SIMNON. Analogamente, as curvas das temperaturas obtidas no SPISC e no SIMNON são sobrepostas na figura 4.10.

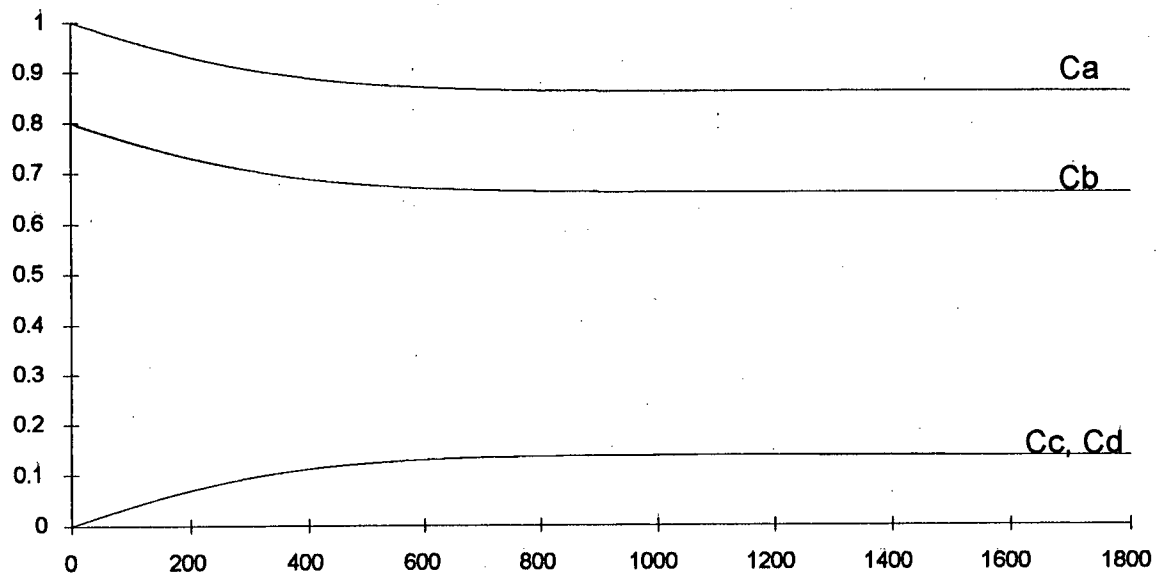



**Figura 4.9** - Sobreposição das curvas das concentrações obtidas no SPISC e no SIMNON.

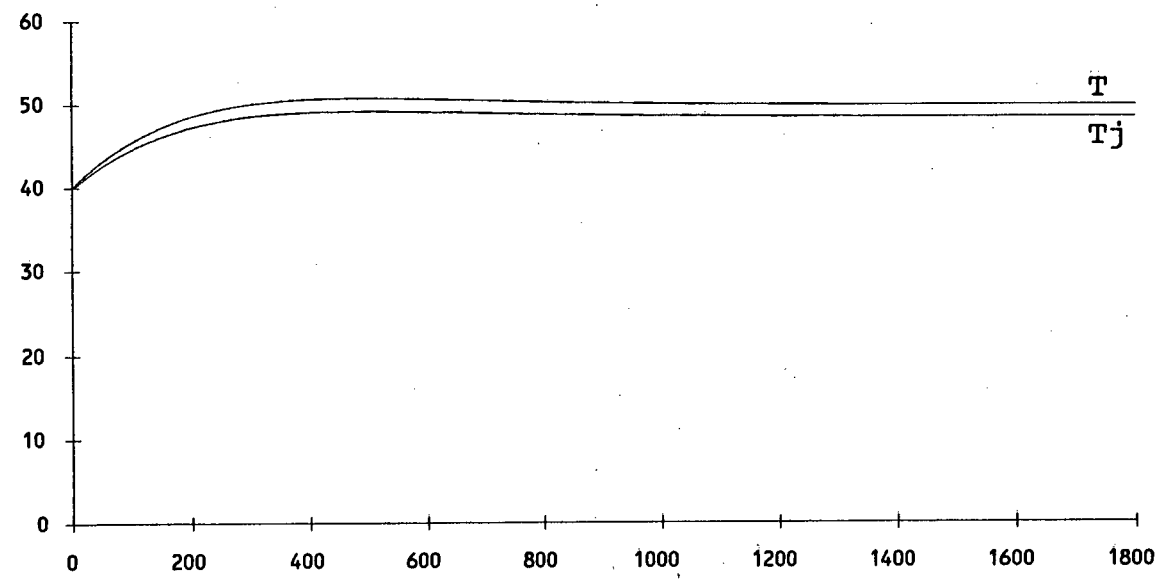



**Figura 4.10** - Sobreposição das curvas das temperaturas obtidas no SPISC e no SIMNON.

### 4.2.2 Linearização

Pelas equações que regem o comportamento do sistema pode-se perceber, claramente, que se trata de um sistema não-linear. Como explanou-se nos capítulos anteriores, para poder aplicar-se as ferramentas clássicas de análise, o sistema deve ser linearizado em torno de um ponto de operação.

Para poder comparar-se a resposta no tempo da representação por variáveis de estado do modelo linearizado obtido no SPISC, com a resposta do sistema não-linear a uma perturbação na

entrada, fez-se uso de um pacote computacional famoso, denominado MATLAB (*matrix laboratory*).

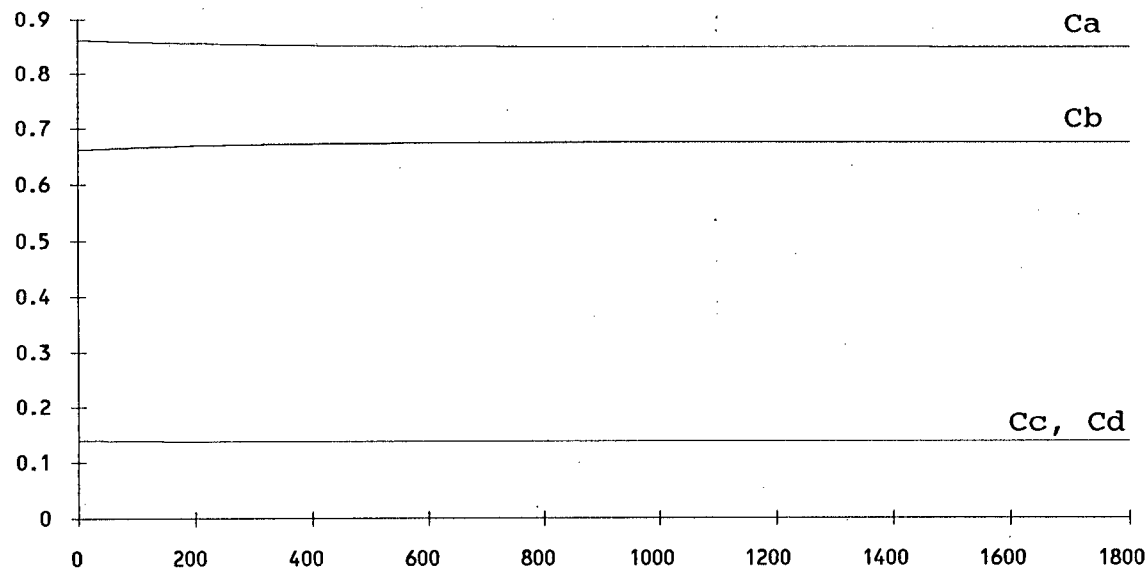



**Figura 4.11** - Comportamento das concentrações das substâncias *A*, *B*, *C* e *D* segundo SPISC, para uma perturbação do tipo degrau incremental na entrada. O intervalo de simulação definiu-se em 1800 s e o passo de integração em 100 ms, utilizando o método de integração trapezoidal amortecido.

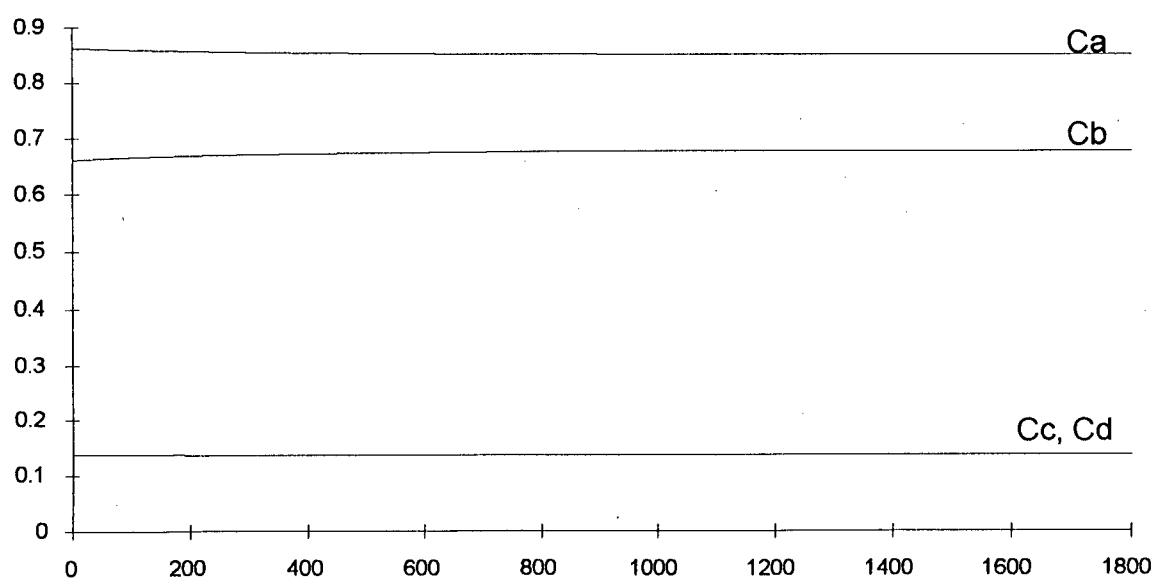



**Figura 4.12** - Comportamento das concentrações das substâncias *A*, *B*, *C* e *D* segundo MATLAB, para uma entrada do tipo degrau incremental. O intervalo de simulação definiu-se em 1800 s e o passo de integração em 100 ms.

Considere-se como única entrada do sistema a alimentação do tanque A, e como saídas as concentrações das substâncias *A*, *B* e *C*. Aplicando uma perturbação na entrada igual a 3% do valor de regime, as concentrações comportam-se como na figura 4.11.

Para verificar se o modelo linear obtido é válido, podem-se sobrepôr ar curvas das simulações realizadas no SPISC e no MATLAB. Ainda, apresenta-se uma tabela com o erro percentual da simulação do modelo linear em relação ao modelo não-linear no instante final de simulação t=1800 s.

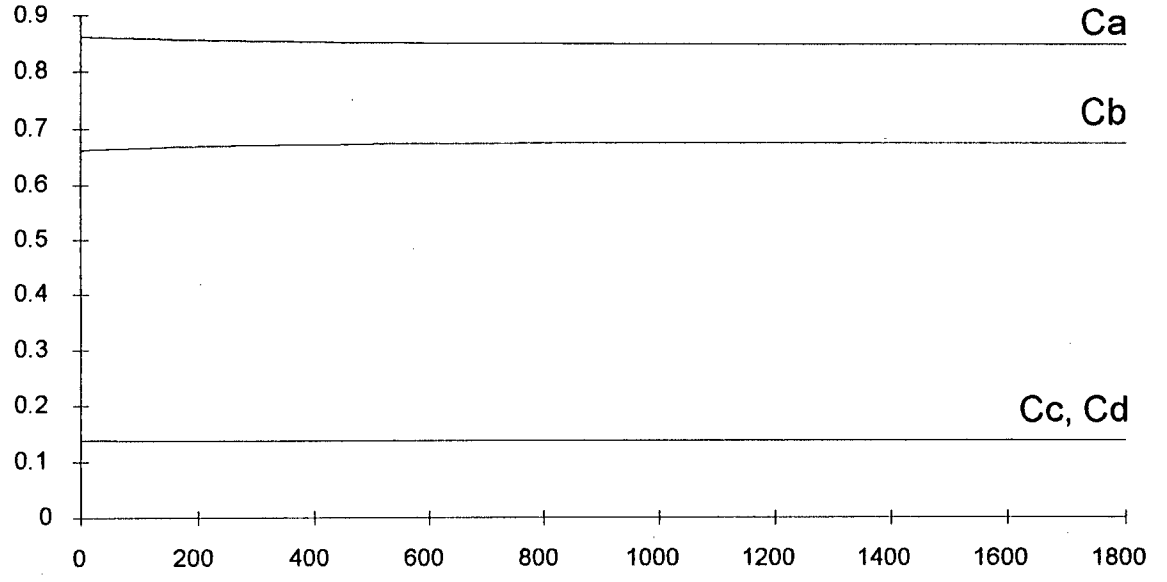



**Figura 4.13** - Comportamento das concentrações das substâncias A, B, C e D. As curvas obtidas do modelo linear e do modelo não-linear estão sobrepostos.

| (mol/ft³) | Modelo linear | Modelo não-linear | Erro (%) |
|---|---|---|---|
| $C_A$ | 0,8482 | 0,8485 | 0,0354 |
| $C_B$ | 0,6751 | 0,6749 | -0,0296 |
| $C_C$, $C_D$ | 0,1369 | 0,1368 | -0,0731 |

**Tabela 4.1** - Comparação dos resultados obtidos do modelo linear e do modelo não-linear.

## 4.3 Predador-caça

Este exemplo conhecido como "predador-caça" é definido pelas seguintes equações:

$$\dot{x} = x - x*y, \qquad x(0) = 10 \tag{4.19}$$

$$y = x*y - y, \qquad y(0) = 1 \tag{4.20}$$

Introduzindo estas equações no ambiente na forma de diagrama de blocos e simulando, obtem-se as curvas apresentadas na figura 4.14.

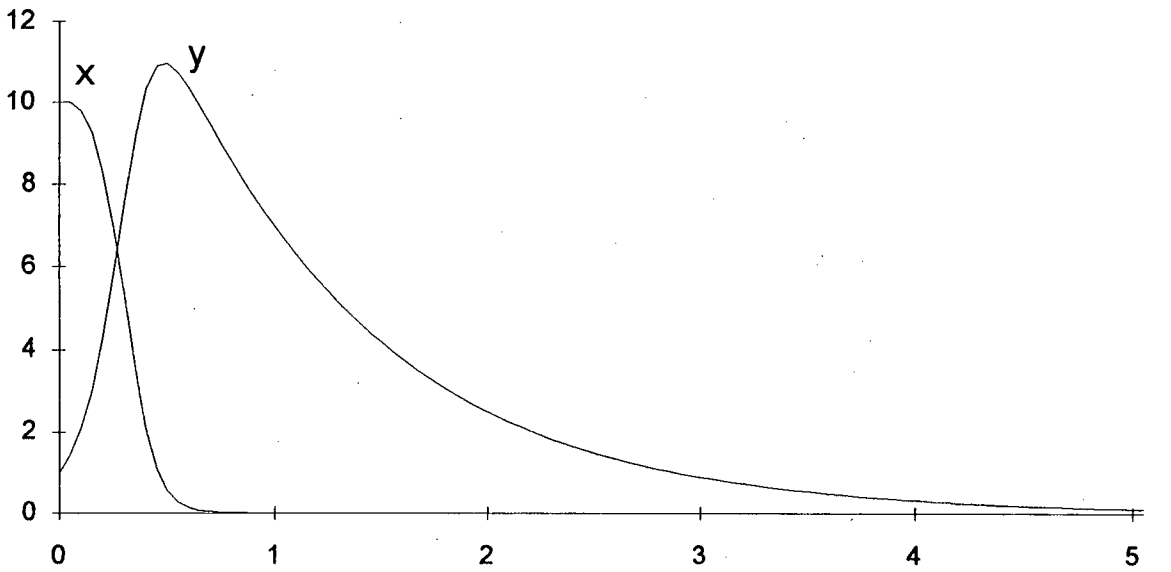



**Figura 4.14** - "Predador-caça" segundo SPISC, simulado num intervalo de 5 s com passo de integração de 50 ms.

### 4.3.1 Comparação de resultados

Para comparar os resultados obtidos faremos novamente uso do SIMNON. O intervalo de simulação e o passo de integração adotados são os mesmos que o anterior.

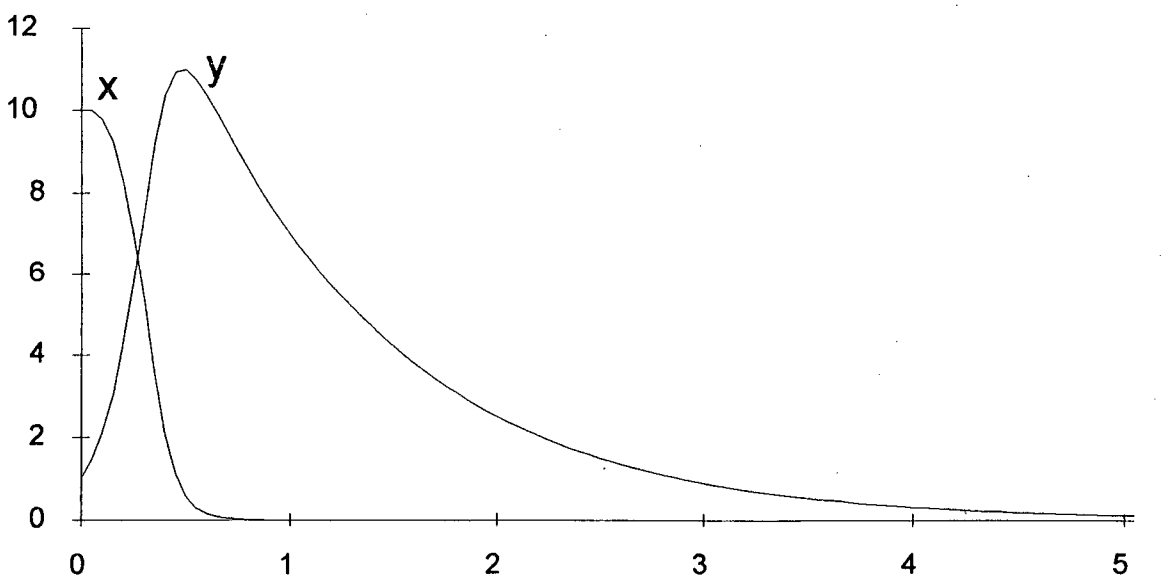



**Figura 4.15** - "Predador-caça" segundo SIMNON.

Para observar melhor a precisão das curvas, uma em relação às outras, faz-se a sobreposição das figuras 4.14 e 4.15. Assim, obtem-se a figura 4.16. Pode-se observar que de fato as curvas praticamente idênticas.

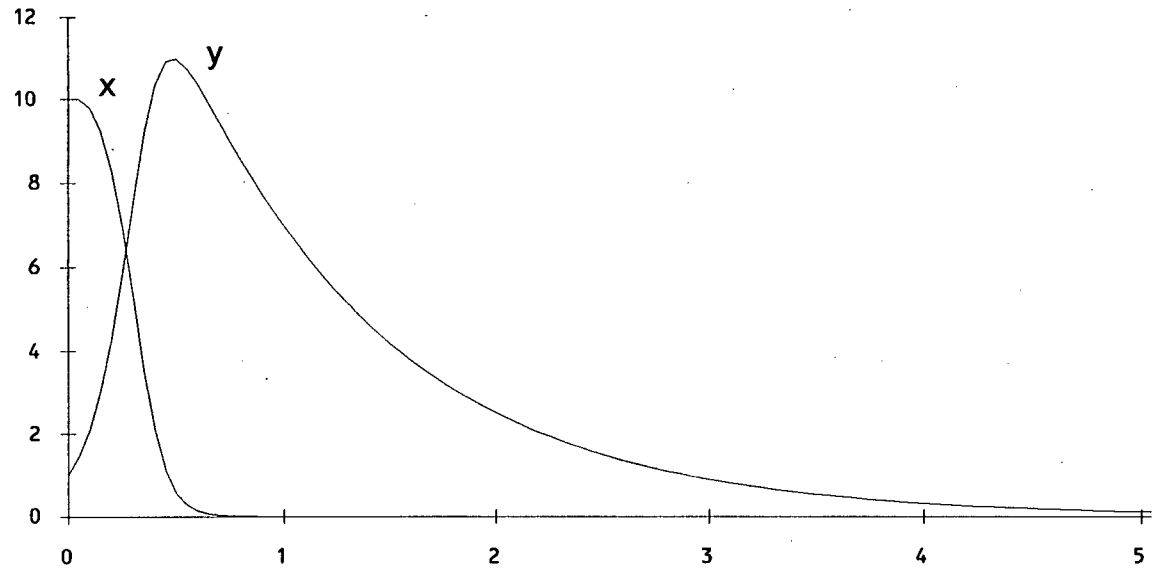



**Figura 4.16** - Sobreposição das curvas representadas nas figuras 4.14 e 4.15.

## 4.4 Conclusões

A construção de super-blocos permite ao usuário, de fato, trabalhar com estruturas mais significativas. Esta característica do pacote possui outras vantagens, como a redução de erro na entrada de dados e a reusabilidade dos super-blocos criados.

A partir das simulações realizadas, pode-se concluir que os métodos de integração numérica empregados são bastante precisos. Estes métodos já forma exaustivamente testados e o presente trabalho vem apenas confirmar mais uma vez este fato. Por outro lado, os algoritmos de linearização mostraram ser bastante exatos dentro das análises realizadas.

# CAPÍTULO 5: CONSIDERAÇÕES FINAIS

Graças à adoção de uma metodologia adequada de trabalho e aos recursos de hardware e software disponíveis no LCMI, o desenvolvimento deste pacote computacional tornou-se uma tarefa agradável e bastante criativa, que resultou num produto com características de modularidade, robustez e eficiência.

Uma etapa muito importante da metodologia é a especificação dos requisitos. Estes, apresentados informalmente, forneceram dados essencias para a escolha adequada da estrutura de dados e da interface gráfica, assim como ajudaram a tomar as primeiras decisões no que se refere à adoção dos recursos computacionais mais adequados para a implementação deste pacote.

Implementar computacionalmente um diagrama de blocos como modelo de um sistema é uma tarefa fácil já que este tipo de modelo realmente simplifica a especificação das interações dentro do sistema, além de ser uma estrutura simples de criar e manipular.

Adotar o pacote gráfico XView como interface homem-máquina forneceu os meios para uma relação mais amigável entre programa e usuários, sendo que estes se sentem atraídos a descobrir e explorar os recursos implementados.

O XView é um toolkit organizado dentro dos princípios da programação orientada ao objeto. A linguagem de programação C++, além de fornecer recursos poderosos e toda a eficiência da linguagem C, permitiu que este pacote tenha sido integralmente desenvolvido dentro de uma única filosofia de programação.

Apesar do pacote atender satisfatoriamente a várias características de qualidade de software, uma delas é bastante restrita. Devido à inexistência de uma versão única do sistema operacional UNIX, a portabilidade do pacote vê-se comprometida. Certamente a portabilidade não será prejudicada se o pacote for instalado em uma estação SUN ou em outra qualquer de fabricante filiado à UI, já que estas estações usam a mesma versão do sistema operacional.

A organização das classes que compõem o programa satisfaz a característica de modularidade. Isto torna as classes que não fazem parte da interface homem-máquina mais reutilizáveis, já que elas consistem apenas de código-fonte C++.

Os algoritmos de integração numérica utilizados apresentam uma boa precisão. Estes algoritmos foram exaustivamente testados no passado em inúmeros trabalhos, e por isto, considerado irrelevante realizar algum estudo neste sentido. Os resultados obtidos em algumas

simulações que comparados com os de outros pacotes comercializados vêm apenas confirmar mais uma vez este fato.

Os algoritmos de linearização mostraram ser bastante exatos. A importância da implementação destas funções está no fato de permitir explorar a biblioteca de tipos abstratos de dados, que possui funções de análise temporal e freqüencial.

Uma perspectiva interessante é incorporar ao ambiente ferramentas de análise para sistemas multivariáveis.

Outra característica interessante do pacote é a flexibilidade oferecida no suporte ao desenvolvimento de processos (blocos) para a biblioteca interna. Esta flexibilidade é observada tanto nos mecanismos de declaração e simulação quanto na implementação (programação) de novos blocos.

Apesar da programação dos novos blocos fornecer a flexibilidade mencionada, esta tarefa pode tornar-se bastante árdua ao usuário, especialmente na criação e tratamento de dados dos painéis de declaração. Uma perspectiva para este ambiente é a criação de funções básicas de manipulação dos objetos do pacote XView necessários a estes painéis, o que tornaria menos complexa a atividade de implementação de novos blocos.

Hoje em dia é necessário todo engenheiro possuir certo dominio sobre uma linguagem de programação. A partir disto, acredito que seja mais prático ampliar a biblioteca interna do ambiente através da criação de classes derivadas da classe Bloco do que criar super-blocos, com a vantagem de facilitar e aliviar o gerenciamento.

Com estas melhorias e dada a estrutura modular do programa, podemos dizer que a integração de algumas ferramentas existentes no LCMI com características semelhantes, será uma atividade interessante de realizar. O desenvolvimento de um pacote computacional a partir dos já desenvolvidos tem uma grande probabilidade de sucesso, cujo resultado espera-se, seja um produto muito mais poderoso.

# ANEXO I: MANUAL DO USUÁRIO

# SIMULADOR DE PROCESSOS INDUSTRIAIS E SISTEMAS DE CONTROLE (SPISC)

**Manual do Usuário**

**Juan Antonio Salvatierra Gimenes**

**Laboratório de Controle e Microinformática (LCMI)**

**Departamento de Engenharia Elétrica**

**Universidade Federal de Santa Catarina**

# SUMÁRIO

# I.1: INTRODUÇÃO

O SPISC (Simulador de Processo Industriais e Sistemas de Controle) é resultado da demanda da área de sistemas de controle por pacotes assistidos por computador voltados à modelagem, simulação, análise e projeto de controladores associados a processos industriais que, em geral, são de natureza multivariável e não-linear.

O SPISC é um pacote desenvolvido utilizando-se a linguagem orientada a objetos C++ sobre o ambiente UNIX[1] em estações de trabalho. As interfaces de entrada e saída do SPISC foram implementadas com o pacote gráfico XView[2] (padrão OPEN-LOOK[1] de interfaces gráficas). Por este motivo a execução do SPISC deve ser realizada dentro do ambiente OpenWindows[2].

O SPISC é também um ambiente integrado porque utiliza uma biblioteca de tipos abstratos de dados relativa à área de controle clássico (tipos tais como função de transferência e variáveis de estado).

O SPISC divide-se em dois modos de operação: declaração e simulação. No modo declaração, o usuário define o sistema como um todo (processo + controladores); no modo simulação, opera-se a simulação do sistema.

Para executar o SPISC, o usuário precisa criar um diretório "/lib" a partir do diretório onde se encontra instalado o sistema. Este será o diretório de trabalho, onde o usuário poderá armazenar seus modelos e super-blocos. Para chamar o SPISC, basta digitar "Spisc" na linha de comando, dentro do ambiente OpenWindows.

---

1 UNIX e OPEN LOOK são marcas registradas da AT&T.
2 XView e OpenWindows são marcas registradas da SUN Microsystems, Inc.

# I.2: ESTRUTURA DA INFORMAÇÃO

A estrutura de bloco adotada para o SPISC é um bloco com nenhuma, uma, duas ou três entradas e apenas uma saída. Quando o bloco possui uma ou mais entradas, sua saída é definida em função das saídas dos blocos que lhe servem de entrada. Quando o bloco não possui entradas, significa que sua saída é conhecida em todo o intervalo de simulação e que independe dos blocos restantes que compõem o modelo.

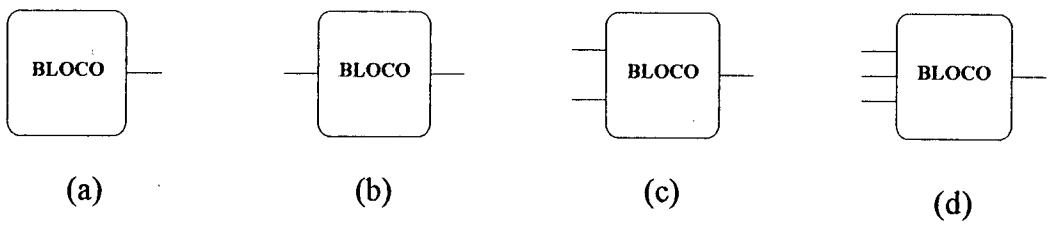



**Figura 2.1** - a) Bloco com nenhuma entrada. b) Bloco com 1 entrada. c) Bloco com 2 entradas. d) Bloco com 3 entradas.

Em função disto, o sistema pode ser representado graficamente como um diagrama de blocos.

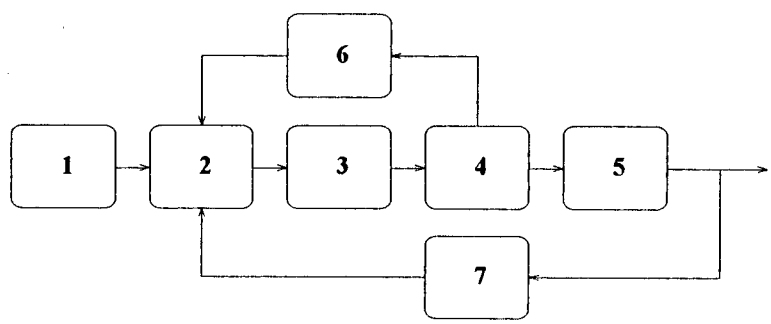



**Figura 2.2** - Diagrama de blocos

O SPISC oferece ao usuário a facilidade dele mesmo poder construir seus próprios blocos a partir dos blocos elementares definidos internamente no ambiente (figura 2.3) e/ou blocos definidos por usuários (figura 2.4). Estas agregações de blocos serão chamadas de super-blocos. A vantagem da utilização de super-blocos está na reusabilidade das informações editadas. Isto poupa tempo além de permitir ao usuário trabalhar com estruturas mais significativas. Um motor elétrico, um tanque de vazão por gravidade, uma bomba compressora, são exemplos de super-blocos que integrados formam um modelo apto para simulação. Tanto os modelos como os super-

blocos podem ser armazenados e posteriormente recuperados mediante funções de acesso a diretórios e arquivos.

O super-bloco pode ter tantas entradas e tantas saídas quantas o usuário considerar necessário, mas pelo menos deverá possuir uma entrada.

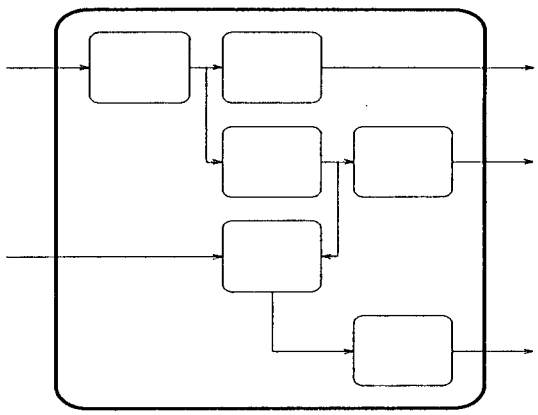


**Figura 2.3** - Diagrama de blocos de um super-bloco com 2 entradas e 3 saídas, composto por blocos elementares.

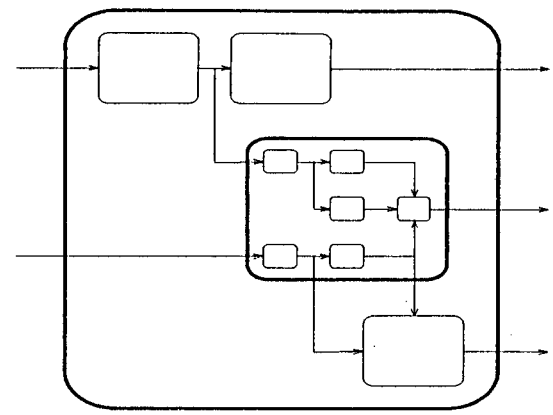


**Figura 2.4** - Diagrama de blocos de um super-bloco com 2 entradas e 3 saídas,composto por blocos elementares e um super-bloco.

# I.3: TIPOS DE BLOCO

A biblioteca contém um conjunto básico de blocos que permite a representação dos modelos utilizados nos estudos referentes a processos industriais e sistemas de controle. Cada bloco é descrito matematicamente, representando uma determinada função. A biblioteca está dividida em 5 grupos, sendo que cada grupo contém um conjunto de blocos com características comuns (ver tabela abaixo).

| Grupo 1<br>Blocos sem entrada | Grupo 2<br>Blocos de função | Grupo 3<br>Blocos dinâmicos | Grupo 4<br>Blocos estáticos | Grupo 5<br>Blocos gerais |
|---|---|---|---|---|
| Constante<br>Rampa<br>Ruído<br>Gerador de seno | Somador<br>ponderado<br>Multiplicador<br>Divisor<br>Raíz<br>Exponencial<br>Seno<br>Polinômio | Dinâmica linear<br>contínua | Limitador<br>Relé<br>Zona morta<br>Histerese<br>Atrito estático | Ganho<br>Ganho controlado<br>Chaveamento<br>Mínimo<br>Máximo<br>Módulo |

**Tabela 3.1** - Divisão dos blocos em grupos.

Para descrever matematicamente cada função associada a estes blocos usar-se-ão algumas convenções.

- A saída do bloco será representada por **y**.
- Se o bloco possuir uma entrada, esta será representada por $\mathbf{u}_1$, se possuir duas, estas serão representadas por $\mathbf{u}_1$ e $\mathbf{u}_2$, e se possuir três, estas serão representadas por $\mathbf{u}_1$, $\mathbf{u}_2$ e $\mathbf{u}_3$.
- O instante de simulação será representado por **t**.

### 3.1 Bloco constante

Seja $A$ o valor da amplitude do degrau. A saída do bloco será dada por

$$\mathbf{y = A}, \quad \forall \mathbf{t} \geq \mathbf{0}$$

### 3.2 Bloco rampa

Seja θ o ângulo formado pela rampa em relação ao eixo horizontal. A saída do bloco será dada por

$$\mathbf{y = \tan(\theta)^* t, \quad \forall t \geq 0}$$

### 3.3 Bloco ruído

Seja **A** o máximo valor que o nível de ruído pode alcançar, e **random** um número randômico entre -1 e 1. A saída do bloco será dada por

$$\mathbf{y = A * random, \quad \forall t \geq 0}$$

### 3.4 Bloco gerador de seno

Seja **A** o valor da amplitude do seno, ω o valor da velocidade angular (em rad/s) e θ o valor do deslocamento angular (em rad). A saída do bloco será dada por

$$\mathbf{y = A * sen(\omega * t + \theta), \quad \forall t \geq 0}$$

### 3.5 Bloco somador ponderado

Seja **a** o valor da ponderação aplicada sobre $\mathbf{u_1}$, **b** o valor da ponderação aplicada sobre $\mathbf{u_2}$, **c** o valor da ponderação aplicada sobre $\mathbf{u_3}$, e **d** o valor de *setup*. A saída do bloco será dada por

$$\mathbf{y = a*u_1 + b*u_2 + c*u_3 + d, \quad \forall t \geq 0}$$

### 3.6 Bloco multiplicador

Seja **k** um fator qualquer. A saída do bloco será dada por

$$\mathbf{y = k*u_1*u_2, \quad \forall t \geq 0}$$

### 3.7 Bloco divisor

A saída do bloco será dada por

$$y = u_1/u_2, \quad \forall t \geq 0 \text{ e } u_2 > 10^{-30}$$

### 3.8 Bloco raíz

A saída do bloco será dada por

$$y = \sqrt{u_1}, \quad \forall t \geq 0 \text{ e } u_1 \geq 0$$

### 3.9 Bloco exponencial

Sejam **a** e **b** dois coeficientes. A saída do bloco será dada por

$$y = a * \exp^{(b*u_1)}, \quad \forall t \geq 0$$

### 3.10 Bloco seno

Seja **A** o valor da amplitude do seno, $\omega$ o valor da velocidade angular (em rad/s) e $\theta$ o valor do deslocamento angular (em rad). A saída do bloco será dada por

$$y = A*\text{sen}(\omega*u_1+\theta), \quad \forall t \geq 0$$

### 3.11 Bloco polinômio

Seja $a_i$ o coeficiente associado ao termo $u_1^i$, $i=0,1,2,...,n$, onde **n** é o grau do polinômio. A saída do bloco será dada por

$$y = \sum_{i=0}^{n} a_i * u_1^i, \quad \forall t \geq 0$$

### 3.12 Bloco dinâmica linear contínua

#### 3.12.1 Forma polinomial

A saída do bloco será dada por

$$y(s) = \frac{a_0 + a_1 * s + \ldots + a_{m-1} * s^{m-1} + a_m * s^m}{b_0 + b_1 * s + \ldots + b_{n-1} * s^{n-1} + b_n * s^n} * u_1(s), \quad \forall t \geq 0$$

onde $a_m \neq 0$, $b_n \neq 0$ e $n \geq m$.

#### 3.12.2 Forma fatorada.

A saída do bloco será dada por

$$y(s) = \frac{k(s-z_1)(s-z_2)\ldots(s-z_m)}{(s-p_1)(s-p_2)..(s-p_n)} * u_1(s), \quad \forall t \geq 0$$

onde $n \geq m$; $k$ é o ganho, $z_i$ i=1,2,...,$m$ são os zeros e $p_j$ j=1,2,...,$n$ são os pólos. Tanto $z_i$ como $p_j$ podem ser complexos.

### 3.13 Bloco limitador

Sejam $l_i$ o valor do limite inferior e $l_s$ o valor do limite superior. A saída do bloco será dada por

$$y = \begin{cases} l_s & u_1 \geq l_s \\ u_1 & l_s > u_1 > l_i \\ l_i & u \leq l_i \end{cases} \quad \forall t \geq 0$$

### 3.14 Bloco relé

Sejam $l_i$ o valor do limite inferior e $l_s$ o valor do limite superior. A saída do bloco será dada por

$$y = \begin{cases} l_i, & \text{se } u_1 < 0 \\ l_s, & \text{se } u_1 \geq 0 \end{cases} \quad \forall t \geq 0$$

### 3.15 Bloco zona morta

Sejam $l_i$ o valor do limite inferior e $l_s$ o valor do limite superior. A saída do bloco será dada por

$$y = \begin{cases} u_1 - l_i, & \text{se } u_1 \leq l_i \\ u_1 - l_s, & \text{se } u_1 \geq l_s \\ 0, & \text{se } l_i < u_1 < l_s \end{cases} \quad \forall t \geq 0$$

### 3.16 Bloco histerese

Sejam $l_i$ o valor do limite inferior e $l_s$ o valor do limite superior. A saída do bloco será dada pela curva mostrada no gráfico abaixo

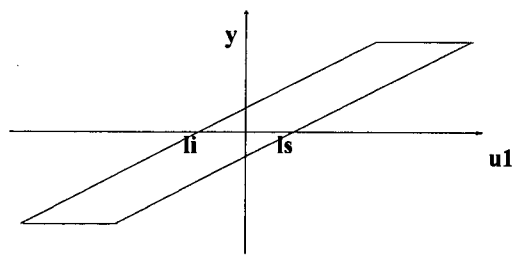



**Figura 3.1** - Gráfico representativo de histerese.

### 3.17 Bloco atrito estático

Sejam $l_i$ o valor do limite inferior e $l_s$ o valor do limite superior. A saída do bloco será dada por

$$y = \begin{cases} u_1, & \text{se } u_1 \le l_i \text{ ou se } u_1 \ge l_s \\ 0, & \text{se } l_i < u_1 < l_s \end{cases} \quad \forall t \ge 0$$

### 3.18 Bloco ganho

Seja **k** o valor do ganho. A saída do bloco será dada por

$$y = k * u_1, \quad \forall t \ge 0$$

### 3.19 Bloco ganho controlado

Sejam $g_i$, $g_S$, $l_i$ e $l_S$ os valores do ganho inferior, ganho superior, limite inferior e limite superior, respectivamente. A saída do bloco será dada por

$$y = \begin{cases} g_s * u_1, & \text{se } l_i < u_2 < l_s \\ g_i * u_1, & \text{se } u_2 \le l_i \text{ ou } u_2 \ge l_s \end{cases} \quad \forall t \ge 0$$

### 3.20 Bloco chaveamento

A saída do bloco será dada por

$$y = \begin{cases} u_2, & \text{se } u_1 \ge 0 \\ u_3, & \text{se } u_1 < 0 \end{cases} \quad \forall t \ge 0$$

### 3.21 Bloco mínimo

Seja **a** o valor da ponderação aplicada sobre $u_1$, **b** o valor da ponderação aplicada sobre $u_2$ e **c** o valor da ponderação aplicada sobre $u_3$. A saída do bloco será dada por

$$y = \min(a * u_1, b * u_2, c * u_3), \quad \forall t \ge 0$$

### 3.22 Bloco máximo

Seja **a** o valor da ponderação aplicada sobre $\mathbf{u_1}$, **b** o valor da ponderação aplicada sobre $\mathbf{u_2}$ e **c** o valor da ponderação aplicada sobre $\mathbf{u_3}$. A saída do bloco será dada por

$$\mathbf{y = max(a * u_1, b * u_2, c * u_3),} \qquad \forall t \geq 0$$

### 3.23 Bloco módulo

Seja **a** o valor da ponderação aplicada sobre $\mathbf{u_1}$, **b** o valor da ponderação aplicada sobre $\mathbf{u_2}$ e **c** o valor da ponderação aplicada sobre $\mathbf{u_3}$. A saída do bloco será dada por

$$\mathbf{y} = \begin{cases} \sqrt{\mathbf{a * u_1 + b * u_2 + c * u_3}}, & \textbf{se radicando} > 0 \\ \mathbf{0}, & \textbf{se radicando} \leq \mathbf{0} \end{cases} \qquad \forall t \geq 0$$

# I.4: DECLARAÇÃO

Quando o SPISC é chamado, por condição *default* o pacote apresenta-se neste modo de operação, como mostrado na figura abaixo. O pacote voltará para este modo sempre que o usuário selecione a opção **"Modo de operação: Declaração"** no painel de controle superior.

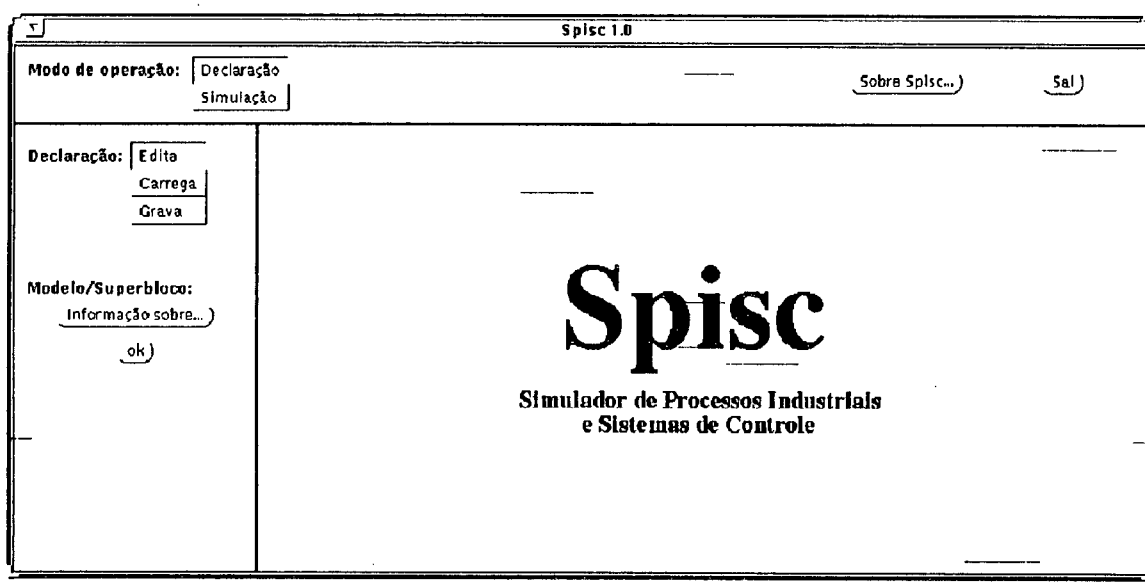



**Figura 4.1** - Janela de apresentação.

## 4.1 Editar

Uma das formas de introduzir um modelo no pacote é via edição. O usuário, fazendo uso do editor embutido no pacote, pode definir as interconexões entre os blocos e super-blocos que compõem o modelo. Quando o usuário seleciona esta opção, abre-se a janela auxiliar **"Declaração:Edição"**, como mostrado na figura abaixo, solicitando o número de linhas que deverão aparecer no editor.

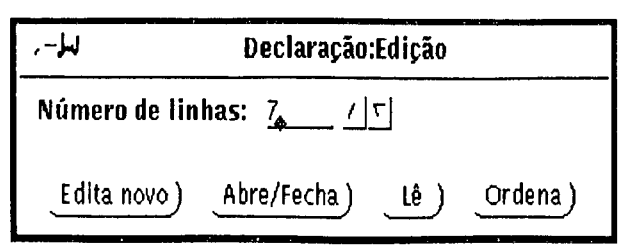



**Figura 4.2** - Janela auxiliar associada à opção **"Declaração-Edita"**.

O número de linhas a editar é dado pela soma do número de blocos elementares e do somatório do número de entradas dos super-blocos. Por exemplo, considere o modelo **M** representado na figura 4.3, composto por 4 blocos elementares e 2 super-blocos, sendo que:

- O super-bloco 1 possui 2 entradas e 1 saída.
- O super-bloco 2 possui 1 entrada e 1 saída.

Então, o número de linhas a aparecerem no editor é 4+2+1=7. Se este número for menor de 1 ou maior de 100, o pacote mostrará um aviso de erro. Procure construir super-blocos, de forma a simplificar a edição e, sobretudo, aproveitar estruturas que possam aparecer em outros modelos.

Para editar o modelo ou super-bloco, o usuário deve identificar os blocos e os super-blocos da seguinte maneira (considere o modelo **M**):

- Os 4 blocos elementares devem ser identificados como **b1**, **b2**, **b3** e **b4**.
- Os 2 super-blocos devem ser identificados como **s1** e **s2**. O super-bloco **s1** deve ter suas entradas identificadas como **s1.1** e **s1.2**, e sua saída como **s1.1**. O super-bloco **s2** deve ter sua entrada identificada como **s2.1** e sua saída como **s2.1**.

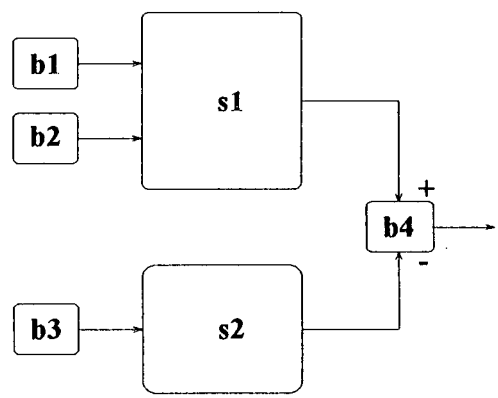



**Figura 4.3** - Diagrama de blocos representativo do modelo **M**.

Uma vez que o usuário entrou com o número de linhas a editar, ele deve selecionar a opção **"Edita novo"**. Se neste momento houver algum modelo presente na memória, o pacote mostrará um aviso de confirmação ou cancelamento da operação. Se a operação for confirmada, o modelo será destruído e o usuário poderá visualizar o editor. Abaixo mostra-se o editor, com os campos preenchidos para o modelo **M**.

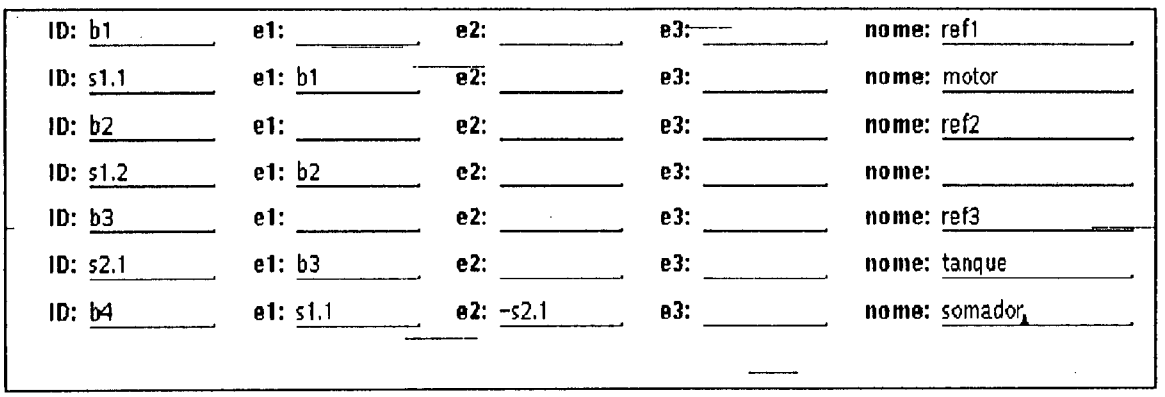



**Figura 4.4** - Editor para a entrada das interconexões entre os blocos.

Observe que em cada linha há cinco campos que o usuário pode preencher. O campo **"ID"** deve ser preenchido obrigatoriamente para todos os blocos e super-blocos. Neste campo, o usuário deve identificar os blocos elementares por **b***i*, *i*=1,2,...,r, onde **r** é o número de blocos elementares componentes do modelo. No caso dos super-blocos, estes devem ser identificados por ***si.j***, *i*=1,2,...,s, *j*=1,2,...,t, onde **s** é o número de super-blocos componentes do modelo e **t** o número de entradas do super-bloco ***i***.

Já o preenchimento dos campos **"e1"**, **"e2"** e **"e3"** depende do tipo de bloco que se está declarando. Se o bloco suporta uma entrada, esta deve ser declarada no campo **"e1"** (não nos campos **"e2"** ou **"e3"**!). Os blocos com apenas uma entrada devem ter este campo preenchido obrigatoriamente. Se o bloco suporta duas entradas, estas podem ser declaradas nos campos **"e1"** e **"e2"**. Se o usuário não acha necessário utilizar as duas entradas do bloco mas apenas uma, esta entrada deve ser declarada no campo **"e1"**. Finalmente, se o bloco suporta até três entradas, os campos **"e1"**, **"e2"** e **"e3"** podem ser preenchidos. Analogamente ao caso anterior, se o usuário não acha necessário utilizar todas as entradas do bloco mas apenas uma ou duas, estas devem ser declaradas nos campos **"e1"** ou **"e1"** e **"e2"** respectivamente.

Para preencher os campos **"e1"**, **"e2"** e **"e3"**, o usuário pode adotar a mesma simbologia adotada para o preenchimento do campo **"ID"**. Por exemplo, imagine que a saída do bloco **b7** seja a única entrada do bloco **b9**. Então, o usuário deve preencher no campo **e1**, **b7**. Se a entrada sofrer inversão de sinal, o usuário deve preencher neste campo, **-b7**. Se por outro lado for um super-bloco que serve de entrada a um bloco ou a um outro super-bloco, a saída do super-bloco deverá ser declarada como ***si.k***, *k*=1,2,...,u, onde **u** é o número de saídas do super-bloco ***i***. Analogamente ao caso do bloco, se a entrada sofrer inversão de sinal, esta deve ser declarada como ***-si.k***.

O campo **"nome"** é opcional. No caso dos blocos, o usuário pode atribuir um nome ao bloco que está declarando. Este campo apesar de opcional, é importante pois, é através deste que o usuário consegue acessar o bloco para outras operações, tais com alteração dos parâmetros,

plotagem da saída, rastreamento da saída, etc. No caso dos super-blocos, este campo serve para associar um arquivo existente no diretório de trabalho a uma **"ID"** do tipo ***si.j***. Observe que o usuário só precisa preencher este campo apenas quando o **"ID"** ***si.j*** aparecer pela primeira vez, independente do número de entrada do super-bloco que esteja sendo declarada.

Uma vez editado o modelo ou super-bloco, o usuário deve selecionar a opção **"Edita"** novamente, para abrir a janela auxiliar **"Declaração:Edição"**, e então selecionar o botão **"Lê"**. Esta opção faz uma verificação na sintaxe da edição. Se for encontrado algum erro na edição, o pacote mostrará um aviso indicando onde está o mesmo. Caso contrário, a janela do editor desaparecerá e o pacote apresentará outra janela, onde o usuário poderá definir para cada bloco **b*i***, o tipo, através do botão-menu **"Bloco"**. Se o bloco possuir parâmetros a serem definidos, aparecerão uma ou mais janelas auxiliares para este fim. Estas janelas são auto-explicativas; o usuário precisa apenas preencher os campos com os valores respectivos.

Caso se esteja editando um modelo, ao terminar a declaração dos tipos, o usuário pode ordenar o modelo para poder simulá-lo. Para isto, ele deve selecionar a opção **"Edita"**, para abrir a janela auxiliar **"Declaração:Edição"**, e poder selecionar o botão **"Ordena"**. Caso haja alguma inconsistência na estrutura do modelo, o pacote mostrará um aviso de erro. A consistência das interconexões entre os blocos e super-blocos é responsabilidade do usuário. O super-bloco, por sua estrutura corresponder a um "modelo incompleto", ele jamais seria validado pelo algoritmo de ordenamento.

### 4.2 Carregar

O usuário, através de funções de acesso a diretórios e arquivos, pode recuperar qualquer modelo editado que tenha sido previamente armazenado. Este modelo, antes de ter sido armazenado, deve ter passado pela fase de verificação de sintaxe. Por isso, quando o modelo é carregado para a memória, ele não passa novamente por esta verificação. O ordenamento dos blocos é feito automaticamente; o usuário não precisa chamar a função que executa este processo. Para carregar um modelo, o pacote deve estar operando no modo **"Declaração"**. Dentro deste modo, o usuário seleciona a opção **"Carrega"**. Assim, abre-se a janela auxiliar **"Declaração:Carregar"**, como mostrado na figura abaixo. O usuário deve preencher este campo com um nome válido de modelo, caso contrário, o pacote mostrará um aviso de erro.

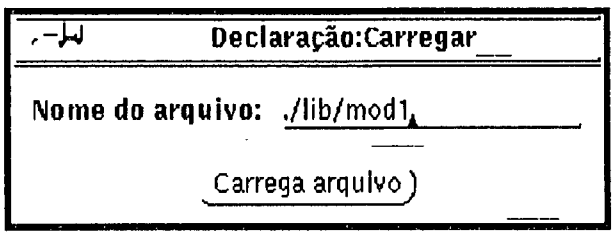



**Figura 4.5** - Janela auxiliar associada à opção **"Declaração-Carrega"**.

## 4.3 Gravar

### 4.3.1 Grava modelo

Para armazenar um modelo, este deve ter sido validado nos processos de verificação de sintaxe e de consistência nas interconexões entre blocos e super-blocos. Assim, a função que executa o armazenamento do modelo num arquivo estará no estado ativo. Para gravar um modelo, o pacote deve estar no modo de operação **"Declaração"**. O usuário deve selecionar a opção **"Grava"** para abrir a janela auxiliar **"Declaração:Grava"** (ver figura 4.6), preencher o campo disponível com o nome do modelo e depois selecionar o botão **"Grava modelo"**. Caso o modelo tenha erro de sintaxe ou inconsistência na sua estrutura, o usuário pode observar que este botão se encontra inativo.

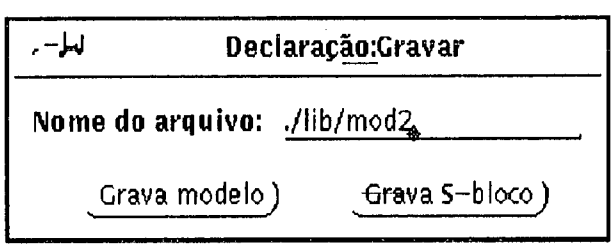



**Figura 4.6** - Janela auxiliar associada à opção **"Declaração-Grava"**.

### 4.3.2 Grava super-bloco

Para armazenar um super-bloco, este deve ter sido validado no processo de verificação de sintaxe. Caso contrário, a função que executa o armazenamento do super-bloco num arquivo estará no estado inativo. Para gravar um super-bloco, o pacote deve estar operando no modo **"Declaração"**. O usuário deve selecionar a opção **"Grava"** para abrir a janela auxiliar **"Declaração:Grava"** (ver figura 4.6), preencher o campo disponível com o nome do super-bloco e depois selecionar o botão **"Grava super-bloco"**.

Assim, aparecerá uma janela de edição auxiliar onde o usuário pode entrar com os números de entradas e saídas do super-bloco. O super-bloco deverá ter, no mínimo, uma entrada. Em função destes números, aparecerão outras duas janelas, onde o usuário deve entrar com os nomes dos blocos que servem de entrada e/ou saída. Se algum bloco servir como entrada ou saída e não tiver um nome associado a ele, o usuário deverá voltar à edição e atribuir um nome ao bloco. Isto pode ser feito seguindo a seqüência **"Declaração-Edita-Abre/Fecha"**. Para encerrar a operação, deve-se selecionar o botão **"Aplica"**.

### 4.4 Informação sobre bloco

É interessante o usuário associar a cada modelo e super-bloco um arquivo de comentários sobre o mesmo. Para isto, o usuário pode utilizar uma janela **"TextEditor"** do OpenWindows. Esses comentarios podem ser chamados a partir do pacote mediante a seqüência **"Declaração-Informação sobre..."** Estes comentarios podem também estar contidos dentro do arquivo onde está declarado o modelo ou super-bloco, acima do carater de controle '#'. As linhas de comentário devem começar sempre com o carater '/'. Para fechar a janela de informação, basta o usuário selecionar a opção **"ok"**.

# I.5: SIMULAÇÃO

Uma vez ordenado o modelo, ele está apto para ser simulado. Para isto, o usuário deve primeiro mudar o modo de operação do sistema, selecionando a opção **"Simulação"** no painel de controle superior. Assim, aparecerá uma janela como na figura abaixo, mostrando os métodos de integração disponíveis ao usuário.


**Figura 5.1** - Métodos de integração.

## 5.1 Métodos de integração.

Para a simulação do modelo estão disponíveis dois algoritmos de integração numérica: trapezoidal com amortecimento e Runge-Kutta de 4ª ordem.

O trapezoidal com amortecimento é um método numericamente estável que apresenta boa precisão. Usando este método, o usuário pode associar a cada bloco de dinâmica linear contínua, um coeficiente $\alpha$, com o qual pode forçar o algoritmo a convergir, mesmo utilizando passos grandes de integração. Por *default*, este coeficiente é 0 para os blocos dinâmicos. Com este valor, o trapezoidal com amortecimento resulta no trapezoidal puro.

O Runge-Kutta de 4ª ordem é um algoritmo bastante preciso. Este algoritmo não foi implementado no SPISC, mas integrado a partir de uma biblioteca de tipos abstratos de dados implementada em outro trabalho de mestrado contemporâneo ao pacote.

O usuário, depois de selecionar o método de integração mais adequado para o modelo, deve preencher os campos **"tempo"** e **"delta"** com o instante final da simulação e o passo de integração respectivamente (ver figura 5.2).

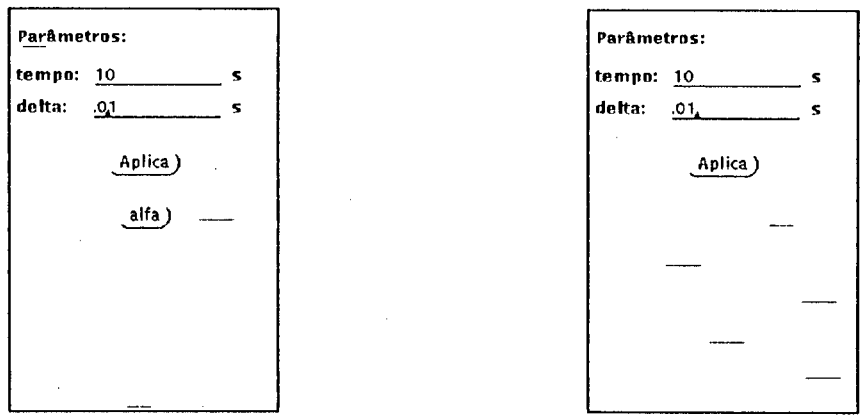



**Figura 5.2** - a) Trapezoidal com amortecimento. b) Runge-Kutta de 4ª ordem.

Observe na figura 5.2a que para o método trapezoidal com amortecimento há um botão **"alfa"**. O usuário, ao selecionar esta opção, abrirá uma janela auxiliar como na figura abaixo, onde poderá entrar com o nome do bloco (repare na importância dos blocos terem um nome associado a ele) e o valor do coeficiente de amortecimento $\alpha$.

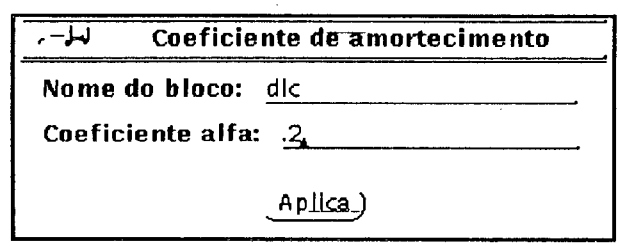


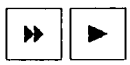

**Figura 5.3** - Coeficiente de amortecimento.

## 5.2 Painel de comando

O painel de comando da simulação é composto por quatro botões contendo símbolos semelhantes ao de um toca-fitas comum, o que facilita a memorização das operações realizadas por estes. Esta idéia foi adotada originalmente no **Sadeca** (Sistema de Avaliação e Desempenho de Controladores Adaptativos). Estes botões são:

Finaliza a simulação. O acionamento deste botão finaliza a simulação e prepara o sistema para o início de uma nova simulação.

Uma nova simulação é iniciada. Limpa-se a janela gráfica. Se o usuário deseja que o sistema parta do repouso, ele deve entrar na janela de alteração de modelo e selecionar o botão **"Condições iniciais nulas"**.



Acelera a simulação ao máximo da capacidade da CPU. Pode-se retornar à simulação em velocidade normal pressionando-se qualquer um destes dois botões.

Pausa/prossegue a simulação. Este botão interrompe temporariamente uma simulação de forma a que se realize alterações e/ou análises.

## 5.3 Alterações

À esquerda do gráfico estão disponíveis três painéis de controle, através dos quais o usuário pode interagir com a simulação, com o objetivo de facilitar a análise do modelo.

### 5.3.1 Alteração no gráfico

Neste painel, pode-se observar que existem os campos **"Y max"** e **"Y min"** (ver figura 5.4). Os valores desejados para os limites do gráfico são definidos nestes campos, e são validados a partir do momento em que seleciona o botão **"Aplica"** situado logo baixo dos mesmos.

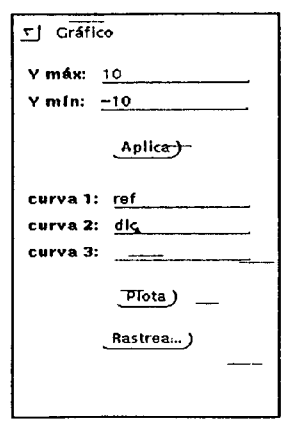



**Figura 5.4** - Painel de alteração do gráfico.

Existem também os campos **"curva 1"**, **"curva 2"** e **"curva 3"** (ver figura 5.4). Nestes campos, o usuário pode entrar com os nomes dos blocos cujas respectivas saídas deseja ver plotadas no gráfico. Se o usuário dispõe de um monitor colorido, poderá observar que a cada uma destas curvas está associada uma cor. Assim, **"curva 1"** está associada à cor vermelha, **"curva 2"**, à verde, e **"curva 3"**, à azul. Estes blocos serão validados só depois do usuário selecionar o botão **"Aplica"** situado logo abaixo destes campos. Se o monitor for monocromático, as curvas plotadas aparecerão (obviamente) com a mesma cor.

Outra facilidade que o usuário dispõe é o rastreamento de uma das curvas plotadas, ponto a ponto. Para isto, o usuário deve selecionar a opção **"Rastrea"**, para abrir a janela auxiliar **"Rastreamento de curvas"** (ver figura 5.5), preencher o campo disponível com o nome do bloco que deseja rastrear e selecionar o botão **"Aplica"**. Assim, quando o usuário colocar o cursor perto da curva que deseja rastrear, o pacote atrairá o cursor sobre um ponto da curva e mostrará os valores da saída e instante de tempo (ver figura 5.6).

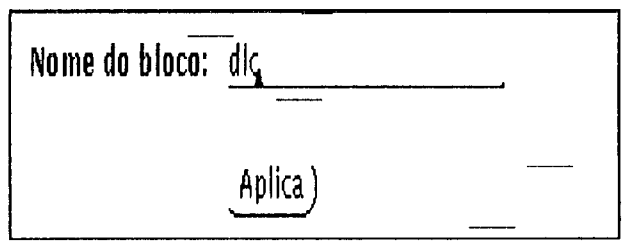



**Figura 5.5** - Janela auxiliar "Rastreamento de curvas".

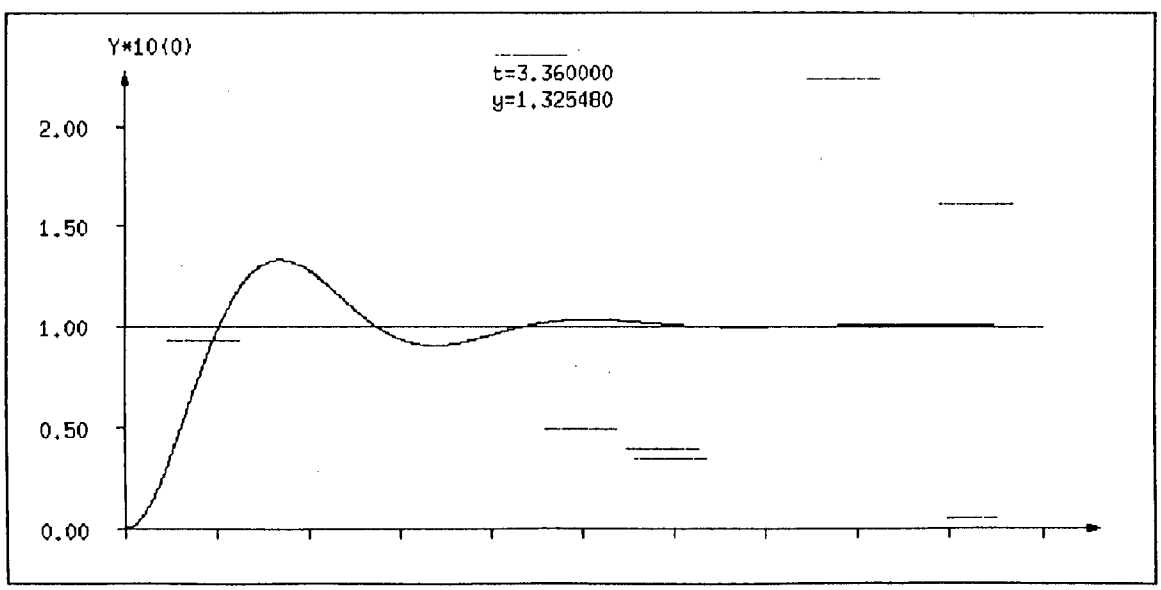



**Figura 5.6** - Exemplo de um gráfico com uma das curvas rastreadas.

### 5.3.2 Alteração no bloco

Através da análise das curvas plotadas, o usuário pode achar necessário ajustar os parâmetros de algum ou alguns blocos que compõem o modelo. Neste sentido, implementou-se uma facilidade que permite alterar estes parâmetros sem a necessidade de re-editar novamente o modelo. O usuário pode mudar o painel de alteração, através do menu, para o painel de alteração de bloco(ver figura 5.7). Neste painel, pode-se observar um campo disponível, onde o usuário pode entrar com o nome do bloco cujos parâmetros deseja mudar. Assim, ao selecionar o botão **"Aplica"**, se abrirá a janela auxiliar **"Alteração de parâmetros"**, mostrando os parâmetros do respectivo bloco. Estes parâmetros podem ser alterados, e só serão validados depois de selecionar-se o botão **"Aplica"** na janela auxiliar.

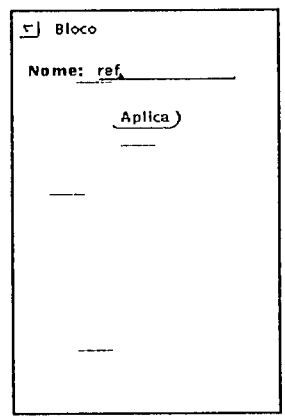



**Figura 5.7** - Painel de alteração do bloco.

### 5.3.3 Alteração no modelo

O usuário é responsável pelo controle das condições iniciais de cada bloco. Isto deve-se a que pode ser desejável analisar o modelo em torno de algum ponto de operação, principalmente quando o modelo é de natureza não-linear. Assim, se as condições dos blocos não forem alterados, ao iniciar-se uma nova simulação, poderá observar-se que as saídas do modelo, no instante t=0, correspondem exatamente aos valores do instante final na simulação anterior. Caso deseje zerar as saídas dos blocos e os estados dos blocos dinâmicos, o usuário pode selecionar a opção **"Condições iniciais nulas"**, no painel de alteração de modelo (ver figura 5.8).

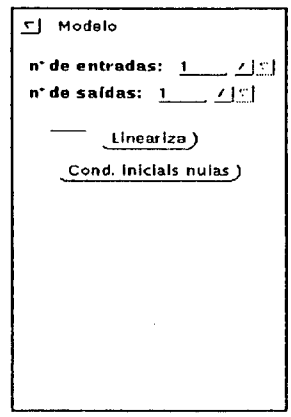



**Figura 5.8** - Painel de alteração do modelo.

# REFERÊNCIAS BIBLIOGRÁFICAS


[Alvarado 83] ALVARADO, Fernando. L., LASSETER, Robert H., SANCHEZ, Juan J. *Testing of Trapezoidal Integration with Damping for the Solution of Power Transient Problems.* IEEE Transactions on Power System Apparatus and Systems, v. PAS-102, n.12. The Institute of Electrical and Electronics Engineers, Inc. USA, December 1983.

[Andersen 91] ANDERSEN, David M., SHERWOOD, Bruce A. *Portability and GUI.* BYTE, v. 16, n. 12. McGraw-Hill. USA, November 1991.

[Caetano 89] CAETANO, Saul S., PAGANO, Daniel J. *Documentação do SDPID (Versão 1.0): Manual do Usuário.* Nota técnica LCMI 89/10. Florianópolis, Novembro 1989.

[Chen 84] CHEN, Chi-Tsong. *Linear System Theory and Design.* Holt, Rinehart and Winston. USA, 1984.

[Cox 86] COX, Brad J. *Object Oriented Programming. An Evolutionary Approach.* Addison-Wesley Publishing Company, Inc. USA, 1986.

[Fairley 85] FAIRLEY, Richard E. *Software Engineering Concepts.* McGraw-Hill Book Company. Singapore, 1985.

[Farines 86] FARINES, Jean-Marie. *Metodologia de Desenvolvimento de Sistemas.* Nota interna: LCMI 86/5. Florianópolis, 1986.

[Franks 72] FRANKS, Roger G. E. *Modeling and Simulation in Chemical Engineering.* Wiley-Interscience. New York, 1972.

[Heller 90] HELLER, Dan. *XView Programming Manual.* O'Reilly & Associates, Inc. USA. 1990.

[Kammer 92] KAMMER, Leonardo C. *Desenvolvimento de um Ambiente de Simulação para Análise de Desempenho de Controladores Adaptativos.* Dissertação de Mestrado, LCMI EEL UFSC. Florianópolis, Dezembro 1992.

[Lima 91] LIMA, Sandro C. *Identificação de Sistemas Assistido por Computador.* Dissertação de Mestrado, LCMI EEL UFSC. Florianópolis, Dezembro 1991.

[Luyben 73] LUYBEN, W. L. *Process Modeling, Simulation, and Control for Chemical Engineers.* McGraw-Hill International. Singapore, 1973.

[Meyer 88] MEYER, Bertrand. *Object-oriented Software Construction.* Prentice Hall. Cambridge, 1988.

[Nye 90] NYE, Adrian. *Xlib Programming Manual.* O'Reilly & Associates, Inc. USA, 1990.

[Nye 92] NYE, Adrian. *Xlib Reference Manual.* O'Reilly & Associates, Inc. USA, 1992.

[Da Luz 90] DA LUZ, Lucas T. Orihuela *Um Ambiente Integrado para Identificação, Modelagem, Análise e Projeto de Sistemas de Controle Associados a Sistemas Elétricos de Potência.* Dissertação de mestrado, LCMI EEL UFSC. Florianópolis, 1990.

[Pappas 91] PAPPAS, Chris H., MURRAY III, William H. *Turbo C++ Completo e Total.* Tradução de Mário Moro Fecchio, revisão técnica de José Eduardo Maluf de Carvalho. Makron Books do Brasil Editora Ltda e Editora McGraw-Hill Ltda. São Paulo, 1991.

[Perry 89] PERRY, Tekla S., VOELCHER, John. *The User-Friendly Interface.* IEEE Spectrum, v.26, n. 19. The Institute of Electrical and Electronics Engineers, Inc. September 1989.

[Pressman 87] PRESSMAN, Roger S. *Software Engineering: A Practitioner's Approach.* McGraw-Hill Book Company. Singapore, 1987.

[Savi 87] SAVI, Vanio M. *Projeto Assistido por Computador para Sistemas de Controle: Especificação e Projeto de um Pacote.* Dissertação de Mestrado, LCMI EEL UFSC. Maio, 1987.

[Schildt 91] SCHILDT, Herbert. *C Completo e Total.* Tradução de Marcos Ricardo Alcântara de Moraes. Makron Books do Brasil Editora Ltda. São Paulo, 1991.

[Sheldon 91] SHELDON, Kennet M., BARRON, Janet J., SMITH, Ben. *Windows Wars.* BYTE, v.16, n. 6. McGraw-Hill. USA, June 1991.

[Suguieda 93] SUGUIEDA, Márcio H., KAMMER, Leonardo C. *Norma para Documentação de Classes C++.* Publicação Interna PI 93-1 LCMI EEL UFSC. Florianópolis, 1993.

[Sun 89a] SUN MICROSYSTEMS. *OPEN LOOK Graphical User Interface Application Style Guidelines.* Addison-Wesley Publishing Company, Inc. USA, 1989.

[Sun 89b] SUN MICROSYSTEMS. *Sun C++ Programmer's Guide.* Sun Microsystems, Inc. USA, 1989.

[Sun 89c] SUN MICROSYSTEMS, AT&T. *AT&T C++ Language System - Reference Manual.* AT&T. USA, 1989.

[Sun 89d] SUN MICROSYSTEMS, AT&T. *AT&T C++ Language System - Selected Readings.* AT&T. USA, 1989.

[Sun 89e] SUN MICROSYSTEMS, AT&T. *AT&T C++ Translator - Library Manual.* AT&T. USA, 1989.

[Vecchiet 87] VECCHIET, K. S. *Computer Graphics Interface: A Developer's Perspective.* Computer Aided Design, v. 19 n. 8, p. 451-55. Butterworth-Heinemann. USA, October 1987.

[Vidyasagar 78] VIDYASAGAR, M. *Nonlinear Systems Analysis.* Prentice-Hall, Inc. Englewood Cliffs, N. J., USA, 1978.

[Wiener 91] WIENER, Richard S. PINSON, Lewis J. *C++: Programação Orientada para Objeto: Manual Prático e Profissional.* Makron, McGraw-Hill. São Paulo, 1991.