EDUARDO DE FREITAS ROCHA LOURES

VIEnCoD - PROPOSTA DE UM AMBIENTE CACSD BASEADO EM PLATAFORMA DE INSTRUMENTAÇÃO VIRTUAL E MATLAB.

Dissertação apresentada ao Curso de Mestrado em Informática Aplicada da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Ciências.

Área de Concentração: Modelagem de Sistemas

Orientador: Dr. Marco Antonio Busetti de Paula

Curitiba

1999

Ficha Catalográfica

Rocha Loures, Eduardo

VIEnCoD - Proposta de um Ambiente CACSD Baseado em Plataforma de Instrumentação Virtual e Matlab. Curitiba, 1999

197 p

Dissertação (Mestrado) -- Pontificia Universidade Católica do Paraná. Mestrado em Informática Aplicada.

1. Modelagem de Sistemas 2. Identificação de Sistemas 3. Otimização de Controladores 4. Ferramenta CACSD (*Computer Aided Control System Design*) - I. Pontificia Universidade Católica do Paraná II. Centro de Ciências Exatas e de Tecnologia III. Mestrado em Informática Aplicada.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
Pró-Reitoria de Pós-Graduação, Pesquisa e Extensão

ATA DA SESSÃO PÚBLICA DE EXAME DE DISSERTAÇÃO DO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA APLICADA DA PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ.

Exame de dissertação nº 011

Aos 08 dias do mês de outubro de 1999, realizou-se a sessão pública de defesa de dissertação "**VIEnCoD – PROPOSTA DE UM AMBIENTE CASCD BASEADO EM PLATAFORMA DE INSTRUMENTAÇÃO VIRTUAL E MATLAB**", apresentada por Eduardo de Freitas Rocha Loures, ano de ingresso 1998, para obtenção do título de Mestre em Ciências. A Banca Examinadora foi composta pelos seguintes professores:

MEMBROS DA BANCA	ASSINATURA
Presidente:. **Prof. Dr. Marco Antonio Busetti de Paula (PUCPR)**	
Prof. Dr. Marco Antonio Barbosa Cândido (PUCPR)	
Prof. Dr. Júlio Cesar Nievola (PUCPR)	
Prof. Dr. Joaquim Eloir da Rocha (CEFETPR)	
Prof. Dr. Paulo Eigi Miyagi (USP/SP)	

De acordo com as normas regimentais a Banca Examinadora deliberou sobre os conceitos a serem atribuídos e que foram os seguintes:

MEMBROS DA BANCA	CONCEITOS
Presidente:. **Prof. Dr. Marco Antonio Busetti de Paula (PUCPR)**	A
Prof. Dr. Marco Antonio Barbosa Cândido (PUCPR)	A
Prof. Dr. Júlio Cesar Nievola (PUCPR)	A
Prof. Dr. Joaquim Eloir da Rocha (CEFETPR)	A
Prof. Dr. Paulo Eigi Miyagi (USP/SP)	A
Conceito Final	A

Observações da Banca Examinadora

Profº Júlio Cesar Nievola
Coordenador do Programa de Pós-Graduação em Informática Aplicada–PUC-PR

Rua Imaculada Conceição, 1155 – Prado Velho – CEP 80215-901 – Telefone: +55 41 330-1669 / + 55 41 330-1654
Fax: + 55 41 330- 1669 – Curitiba – Paraná – Brasil - E-mail:ppgia@rla01.pucpr.br

Dedico este trabalho a Leonardo e Iracema, que me deram vida, a minha esposa Karyn e meu filho Matheus pela compreensão e amigos pelo apoio e incentivo que sempre me deram.

Agradecimentos

Ao Prof. Dr. Marco Antônio Busetti de Paula pela orientação e apoio durante todo o desenvolvimento do trabalho.

Ao Prof. Dr. Paulo Eigi Miyagi pela sua disponibilidade e importante contribuição na avaliação deste trabalho.

Ao Prof. Dr. Marco Cândido e Prof. Dr. Júlio César Nievola pelo apoio prestado no fornecimento de condições para a conclusão do trabalho.

A Prof.ª Dr.ª Lúcia Valéria de Arruda pela orientação durante as fases iniciais do projeto.

Ao amigo Marcos Silveira que compartilhou de momentos alegres e difíceis comigo, sempre pronto para ajudar.

Ao colega Júlio Jansson e em especial ao amigo Augusto Erzinger, pelas importantes contribuições ao projeto.

Aos colegas e amigos Humberto Araújo, Gustavo Oliveira, Fabrício Cabral, Sérgio Gouvêa e Edson Pinheiro pelo apoio e pelas opiniões acerca do projeto.

A todos que contribuíram para a realização deste trabalho e foram involuntariamente omitidos.

A Deus pela fé.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

ADC – *Analog Digital Converter*

API – *Application Program Interface*

CACSD – *Computer Aided Control System Design*

CAD – *Computer Aided Design*

CAE – *Computer Aided Engineering*

CAM – *Computer Aided Manufacturing*

CAP – *Computer Aided Planning*

CAQ – *Computer Aided Quality Control*

CAT – *Computer Aided Testing*

CDSC – Ciclo de Desenvolvimento de Sistemas de Controle

CIM – *Computer Integrated Manufacturing*

CIVM – *Controller Implementation VIEnCoD Module*

CLP – Controlador Lógico Programável

CVI – *C for Virtual Instrumentation*

DAC – *Digital Analog Converter*

DAQ – *Data Acquisition Board*

DDE – *Dynamic Data Exchange*

DSP – *Digital Signal Processing*

ECU – Electronic Control Unit

ECUVI – *Electronic Control Unit Virtual Instrumentation-based*

ERP – *Enterprise Resource Planning*

FFT – *Fast Fourier Transform*

FMS – *Flexible Manufacturing System*

GUI – *Graphical User Interface*

HIL – *Hardware-in-the-Loop*.

ISA – *International Society for Measurement and Control*

LAS – Laboratório de Automação e Sistemas

LTI – *Linear Time Invariant*

MA – Malha aberta

MES – *Manufacturing Execution Systems*

MF – Malha fechada

MIMO – *Multi-Input Multi-Output*

MRP – *Manufacturing Resource Planning*

OS – *Operation System*

PC – *Personal Computer*

PCP – Planejamento e Controle da Produção

PCVM – *Plant Configuration VIEnCoD Module*

PID – Controle Proporcional Integral e Derivativo

PIVM – *Plant Identification VIEnCoD Module*

PRBS – *Pseudorandom Binary Sequence*

RCP – *Rapid Control Prototyping.*

RTI – *Real Time Interface*

RTOS – *Real Time Operational System*

RTW – *Real-Time Workshop*

RTX – *Real Time Extension for Windows.*

S/H – *Sample / Holder*

SDCD – Sistema Digital de Controle Distribuído

SISO – *Single-Input Single-Output*

TIPS – *Totally Integrated Production System*

VI – *Virtual Instrumentation* ou *Virtual Instrument*

VIEnCoD – *Virtual Instrumentation-based Environment Controller Design*

VISA – *Visual Instrumentation Software Arquitecture*

VXI – *VME eXtension for Instrumentation*

Resumo

Este trabalho apresenta a proposta de um ambiente CACSD (*Computer-Aided Control System Design*) destinado ao estudo, projeto e otimização de controladores para as mais variadas plantas, desde sistemas mecatrônicos à processos industriais.

Métodos de otimização, controle e supervisão mais específicos ao ambiente da indústria são necessários já ao nível de chão de fábrica. Neste nível, surge a motivação da utilização de uma ferramenta que otimize a operação das diversas plantas existentes, refinando e estendendo os objetivos no controle da produção. Neste cenário surge o contexto de Gestão Empresarial Integrada.

Em laboratórios relacionados à Instrumentação, Controle e Automação nas Universidades e Instituições de Ensino e Pesquisa, as necessidades convergem para investigação científica, processo de ensino-aprendizagem e serviços na área de controle, havendo a necessidade de um ambiente CACSD adequado de suporte.

A ferramenta proposta, VIEnCoD (*Virtual Instrumentation-based Integrated Environment for Controllers Design*) é um ambiente baseado no conceito de Instrumentação Virtual integrado às ferramentas de análise e projeto de sistemas de controle do Matlab e Simulink. A metodologia de integração deste ambiente é descrita de forma a dar suporte integral a todo o Ciclo de Desenvolvimento de Sistemas de Controle (CDSC), desde a fase de Modelagem e Identificação da planta à Síntese do controlador.

Ao final, um protótipo de controle de nível de líquido é utilizado como estudo de caso onde todo o Ciclo é aplicado e todos os recursos do ambiente são explicitados. O estágio final é a validação de um controlador PID.

Resumo

Este trabalho apresenta a proposta de um ambiente CACSD (*Computer-Aided Control System Design*) destinado ao estudo, projeto e otimização de controladores para as mais variadas plantas, desde sistemas mecatrônicos à processos industriais.

Métodos de otimização, controle e supervisão mais específicos ao ambiente da indústria são necessários já ao nível de chão de fábrica. Neste nível, surge a motivação da utilização de uma ferramenta que otimize a operação das diversas plantas existentes, refinando e estendendo os objetivos no controle da produção. Neste cenário surge o contexto de Gestão Empresarial Integrada.

Em laboratórios relacionados à Instrumentação, Controle e Automação nas Universidades e Instituições de Ensino e Pesquisa, as necessidades convergem para investigação científica, processo de ensino-aprendizagem e serviços na área de controle, havendo a necessidade de um ambiente CACSD adequado de suporte.

A ferramenta proposta, VIEnCoD (*Virtual Instrumentation-based Integrated Environment for Controllers Design*) é um ambiente baseado no conceito de Instrumentação Virtual integrado às ferramentas de análise e projeto de sistemas de controle do Matlab e Simulink. A metodologia de integração deste ambiente é descrita de forma a dar suporte integral a todo o Ciclo de Desenvolvimento de Sistemas de Controle (CDSC), desde a fase de Modelagem e Identificação da planta à Síntese do controlador.

Ao final, um protótipo de controle de nível de líquido é utilizado como estudo de caso onde todo o Ciclo é aplicado e todos os recursos do ambiente são explicitados. O estágio final é a validação de um controlador PID.

Abstract

This work presents the proposal of a CACSD (Computer-Aided Control System Design) environment aimed to the study, design and optimization of controllers for the most varied plants, from mechatronic systems to industrial processes.

More specific supervision, control and optimization methods to the company, are necessary at the shop floor level. At this level, there's the motivation on using a tool that optimize the operation of several existent plants, refining and extending the goal of the production control. At this environment the Integrated Management Systems context appears.

At the laboratories related to Control, Instrumentation and Automation of Universities and Research Centers, the needs converge to the scientific investigation, teaching-learning process and services in the control area, needing an appropriate CACSD environment to support.

The proposed tool, VIEnCoD (Virtual Instrumentation - based Integrated Environment for Controllers Design) is an environment based on the Virtual Instrumentation concept integrated to the control system analysis and design tools of Matlab and Simulink. The methodology of integration of this environment is described to give integral support to the whole Control System Development Cycle (CDSC), from the phase of Identification and Modelling of the plant to the controller's Synthesis.

At the end, a plant of liquid level control is used as case study where the whole Cycle is applied and all the environment resources are validated. The final stage is the validation of a PID controller.

Abstract

This work presents the proposal of a CACSD (Computer-Aided Control System Design) environment aimed to the study, design and optimization of controllers for the most varied plants, from mechatronic systems to industrial processes.

More specific supervision, control and optimization methods to the company, are necessary at the shop floor level. At this level, there's the motivation on using a tool that optimize the operation of several existent plants, refining and extending the goal of the production control. At this environment the Integrated Management Systems context appears.

At the laboratories related to Control, Instrumentation and Automation of Universities and Research Centers, the needs converge to the scientific investigation, teaching-learning process and services in the control area, needing an appropriate CACSD environment to support.

The proposed tool, VIEnCoD (Virtual Instrumentation - based Integrated Environment for Controllers Design) is an environment based on the Virtual Instrumentation concept integrated to the control system analysis and design tools of Matlab and Simulink. The methodology of integration of this environment is described to give integral support to the whole Control System Development Cycle (CDSC), from the phase of Identification and Modelling of the plant to the controller's Synthesis.

At the end, a plant of liquid level control is used as case study where the whole Cycle is applied and all the environment resources are validated. The final stage is the validation of a PID controller.

Capítulo 1

1 Introdução

O grande avanço da microeletrônica e dos sistemas computacionais permitiram o estudo e aplicação de novos algoritmos e técnicas de controle. O aumento da complexidade dos processos industriais, com requisitos específicos de desempenho, elevado número de variáveis de entrada e saída, demandaram diferentes abordagens e produção científica na área de controle.

Como conseqüência da necessidade de suporte a elevada quantidade de cálculos e iterações que os novos métodos de controle exigiam, aliado à redução do custo dos computadores, os ambientes de simulação computacional desenvolveram-se. A simulação pode ser utilizada para validar uma determinada estratégia de controle antes de sua aplicação em um processo real.

Os pacotes com proposta CACSD aglutinam hoje todo este conhecimento em ambientes estruturados de análise, projeto e simulação. Entretanto, seus algoritmos e ferramentas são dispostos de forma setorizada, em áreas específicas de controle, onde apenas especialistas, com algum tempo de uso, adquirem maior proveito das mesmas.

O ambiente proporcionado por um pacote integrado deve conter recursos que possibilitem a fácil interação homem-máquina, interfaces gráfica intuitivas e alto grau de interatividade. Desta forma, prevê-se que o tempo de ambientação do usuário reduz com maior aproveitamento das ferramentas disponíveis.

A proposta geral deste ambiente não deve se prender apenas em áreas específicas, mas sim dar suporte a todo um ciclo de desenvolvimento. Tal ciclo abrange as etapas em um projeto de sistemas de controle que podem ser resumidas a :

- Modelagem do sistema físico: geração de um modelo matemático e representação no computador;

- Identificação : quando o conhecimento de alguns parâmetros físicos da planta não são possíveis de serem obtidos, requerendo-se, portanto, técnicas de identificação para a modelagem do sistema;
- Síntese de leis de controle : análise dos resultados incluindo técnicas de simulação;
- Implementação do controlador : realização da lei de controle.

Esta abrangência de enfoque sugere o direcionamento dos ambientes CACSD, não apenas à pesquisa e serviços (como otimização de malhas industriais), mas também ao ensino dos conceitos e teoria de controle. Este enfoque exige, além do alto grau de interatividade gráfica com o usuário, neste caso aluno, uma integração estruturada dos algoritmos e ferramentas correlatas a cada etapa do ciclo de desenvolvimento. Os conceitos de controle devem se apresentar de forma clara durante a caracterização dos algoritmos como ferramenta de suporte.

O pequeno conteúdo auto-instrutivo e intuitivo dos ambientes existentes dificultam a assimilação de conceitos de controle clássico, moderno e conteúdos mais avançados. Este ponto tem sido tema para uma linha de pesquisa que conta com a participação de renomados autores na área de controle [WITTENMARK et al., 1998] [ÅSTRÖM et al., 1998].

Um recurso desejável ao processo ensino-aprendizagem e investigação científica através de um ambiente CACSD é a possibilidade de realização de experimentos com sistemas físicos como tratado, por exemplo, em [ARGONDIZZA et al., 1997]. Os resultados e conhecimentos trabalhados na simulação podem ser contextualizados e validados, desde a identificação da planta à síntese do controlador. Esta preocupação tem sido tema de outros trabalhos tais como [ARMSTRONG, 1997] [COELHO, 1995] [SOARES et al., 1995] [HORÁCEK, 1994].

A inserção deste recurso a um ambiente CACSD deve estar perfeitamente integrado à proposta do mesmo e sob uma adequada metodologia e conceitos. **O *Ciclo de Desenvolvimento de Sistemas de Controle*** (**CDSC**) é a metodologia base para o atendimento e suporte dos pontos acima colocados sendo a ***Instrumentação Virtual*** e

simulação ***Hardware-in-the-Loop*** conceitos presente no suporte à simulação com elementos físicos presentes. Ambos (metodologia e conceitos) serão tratados em detalhes no presente trabalho.

De acordo com o cenário colocado, observa-se a caracterização de duas plataformas ou ambientes distintos: *ambiente de simulação, análise e projeto de sistemas de controle* e *plataforma de interface com elementos físicos*. Ambas devem apresentar elevado grau de integração e características que possibilitem a minimização ou eliminação dos problemas colocados, comuns aos ambientes CACSD atuais.

A interface com o sistema físico se dá através de uma *plataforma de instrumentação* (*hardware e software*) sendo desejável a presença de algumas características:

- Geração, tratamento e aquisição de todos os tipos de sinais envolvidos no escopo de sistema de controle proposto;
- Fácil configuração e alta confiabilidade, eliminando preocupações e esforços do usuário neste sentido e não desviando do objetivo principal que é controle;
- Ambiente de programação e configuração da plataforma: interfaces gráficas intuitivas com grande interatividade com o usuário;
- Troca de dados e códigos dinamicamente com o ambiente de simulação;
- Programação e características de tempo real que permita a implementação do controlador obtido na simulação.

Na *plataforma para análise e projeto de sistemas de controle* os algoritmos clássicos e de aplicações específicas, desenvolvidos pela comunidade científica, devem estar disponíveis. Da mesma forma, um ambiente de simulação com recursos gráficos e programação eficientes devem ser disponibilizados. O Matlab-Simulink é plataforma considerado no estado-da-arte que apresenta estas características, tendo uma elevada disseminação na comunidade científica e acadêmica e, atualmente, com alguma penetração na indústria.

Dentro do escopo então apresentado, apresenta-se o ***VIEnCoD*** (*Virtual Instrumentation-based Integrated Environment for Controllers Design*), um *toolbox*

Matlab com proposta CACSD destinado à análise, projeto, simulação e implementação de controladores. Sua ***proposta*** (***objetivo***) **principal** é eliminar ou minimizar as deficiências dos pacotes comerciais existentes bem como a introdução de novas funcionalidades, no direcionamento ao ensino, pesquisa e serviços na área de controle através do emprego de *metodologia* e *conceitos* específicos. Desta forma são estabelecidos alguns ***objetivos específicos***:

- Promover ambiente amigável provido de adequado conteúdo informativo e intuitivo com característica multifuncional: suporte ao ensino, pesquisa e serviços na área de controle;
- Integração com elementos físicos dando suporte integral ao CDSC;
- Possibilidade de expansão e adaptação às necessidades do pesquisador como simulação e testes de novas estratégias de controle - fácil integração/incorporação de novos módulos (*toolboxes*);
- Flexibilidade e abertura em termos de *hardware* (instrumentação) na interface com elementos físicos;

A reunião destas características, permite a ampliação do contexto de aplicações para o ambiente CACSD. O direcionamento apenas acadêmico e científico na área de controle pode se estender a patamares de integração com sistemas operando em níveis hierárquicos superiores dentro de um processo produtivo.

Segundo [SIPPER et al., 1997], o Controle é um ingrediente essencial em todo sistema de produção automatizada. Linhas de transferência, comando numérico, robôs industriais, sistemas de armazenagem de materiais de suporte e sistemas flexíveis de manufatura, indústria de processos - todos requerem alguma forma de controle para assegurar a operação adequada.

Este contexto sugere a caracterização do ambiente CACSD proposto como ferramenta de otimização da operação de máquinas e processos ao nível de chão de fábrica, estendendo os objetivos do sistema produtivo e funcionando como elemento de integração a um sistema de gestão integrada ou ERP (*Enterprise Resource Planning*). Esta caracterização é colocada sob uma abordagem introdutória não tendo sido objeto de

implementações no atual trabalho, mas sugerindo e motivando a sua continuidade futura nesta linha.

Este trabalho apresenta a descrição de todas as concepções e implementações que se fundamentaram nos *objetivos principais* acima propostos e está dividido em sete capítulos descritos como se segue.

O capítulo 2 descreve de forma mais detalhada todos os elementos de motivação do projeto VIEnCoD. As necessidades vinculadas ao ensino, pesquisa e prestação de serviços relacionados à estruturas de laboratórios voltados à instrumentação, controle e automação. A contextualização com sistema de gestão empresarial integrada é apresentada de forma introdutória, conforme já comentado. Os pacotes comerciais com proposta CACSD e as atuais pesquisas científicas são analisados.

O capítulo 3 descreve a metodologia e conceitos envolvidos no desenvolvimento de sistemas de controle. O conteúdo de controle abordado em cada etapa do ciclo é colocado.

O capítulo 4 apresenta a estrutura e organização do VIEnCoD. Uma metodologia de integração das ferramentas de análise e projeto de controle e sistema de Instrumentação Virtual é descrita. Considerações sobre a concepção e objetivos do ambiente, programação, estrutura de arquivos, troca de dados e códigos, ambiente em tempo real e outros são apresentados.

O capítulo 5 descreve o aspecto operacional do ambiente, descrevendo as características e objetivos de cada um dos módulos que compõem o *toolbox*. São módulos que compõem o sistema de supervisão e controle da plataforma de *hardware* e módulos integrados que caracterizam cada etapa do CDSC.

O capítulo 6 apresenta a implementação do VIEnCoD através de estudo de caso. O CDSC foi aplicado a um protótipo de uma planta de controle de nível de fluido. Desde a modelagem e identificação da planta à síntese do controlador todos os recursos do ambiente são avaliados. Considerações sobre tempo real, implementação de códigos de

algoritmos de controle e troca de dados são apresentados. Por fim, o controlador resultante PID é implementado pela plataforma de *hardware* com propósito de validação.

O sétimo capítulo apresenta conclusões sobre o trabalho e a proposta de continuidade do mesmo através da sugestão de novas implementações e suprimento das deficiências existentes.

Capítulo 2

2 Ambientes CACSD e Motivação do Projeto VIEnCoD

2.1 Direcionamento ao Ensino e Pesquisa

O Controle Automático cobre um largo campo de tópicos da matemática aos processos físicos e sistemas computacionais e de engenharia. Um engenheiro de controle deve dominar um grande número de conceitos, técnicas e idéias além de ser capaz de aplicá-los em problemas reais. Tarefas típicas incluem modelagem matemática, análise, simulação, projeto e implementação[MATKO et al., 1994].

O ensino de controle encontra um papel importante para que todos estes tópicos e requisitos sejam alcançados. Para tal, sugerem-se novas contribuições aos conteúdos e metodologias empregadas nos cursos de graduação. Neste cenário, os ambientes CACSD encontram-se como principais ferramentas no auxílio deste processo, sendo alvo de investigações da comunidade científica.

Neste item serão abordadas as deficiências encontradas nos ambientes CACSD existentes no tocante ao ensino e pesquisa; analisando, direcionando e sugerindo a inserção de novas características, ferramentas e principalmente metodologia apropriadas. O objetivo é demonstrar conceitos, ferramentas e aplicabilidade da engenharia de controle através de um ambiente CACSD.

As disciplinas de Controle nos cursos de engenharia normalmente se apresentavam com conteúdo extensivo de métodos matemáticos e cálculo numérico, muitas vezes retirando o principal foco - os conceitos e a teoria de controle. Com o desenvolvimento dos recursos computacionais esta tarefa pode ser suportada pelas ferramentas CACSD,

destinando-se mais tempo ao trabalho de concepção, projeto e análise dos sistemas de controle [SILVEIRA et al., 1998].

Em suas primeiras versões, tais ambientes agrupavam um grande número de algoritmos e soluções numéricas sob sintaxes e recursos gráficos pouco amigáveis. O usuário tinha que adquirir primeiramente um bom conhecimento dos comandos e limitações para se começar a obter algum resultado. Muitas vezes, para a elaboração de uma simulação de um problema técnico era necessário um montante de implementações a nível de programação : base e leitura de dados, algoritmos de simulação, interface de apresentação de dados,... etc. Desta forma, gastava-se muito tempo em depuração de programa e pouco tempo na compreensão do problema [SILVEIRA, 1998].

Um outro problema presente era a ausência de conteúdo conceitual ou didático do ambiente; existia apenas o conjunto de algoritmos e funções específicas a problemas específicos dispostos isoladamente. Em pesquisa a literatura de ambientes com proposta CACSD, nota-se que até recentemente era dada maior ênfase a arquitetura e desempenho [BOOM et al., 1994]. Com as bibliotecas de funções e algoritmos de controle avançado desenvolvidos e difundidos no meio acadêmico, inciaram-se esforços no sentido de disponibilizá-los no ambiente de forma mais amigável. Entretanto, ainda hoje existe deficiência ou ausência na integração destas ferramentas, necessárias por exemplo, no suporte ao ciclo desde a identificação à síntese do controlador.

As últimas versões dos pacotes comerciais vêm incorporando novas características didáticas. A maior parte dos exemplos clássicos do ensino de engenharia já são possíveis de serem solucionados pelos aplicativos de forma amigável. Porém, não é este o enfoque principal deles, e sua utilização em simulações de situações reais requer um estudo mais aprofundado do aplicativo.

A grande vantagem em se usar uma ferramenta CACSD no ensino de controle é a de fornecer ao usuário uma realimentação de suas decisões. Ela ajuda a integrar os conhecimentos e observar as relações de causa-e-efeito através das simulações. Um exemplo disto é a entrada de critérios de especificação de desempenho no tempo durante o projeto de um determinado controlador e obter-se sua resposta ao degrau. Entretanto, o

processo de aprendizado não é, em geral, suportado pelo ambiente, isto é, o mesmo deveria prover [SEIDEL, 1994]:

- Informação na teoria de controle;
- Informações no uso do ambiente;
- Métodos e estratégias para solução de problemas;
- Exemplos para elucidar a teoria e solução do problema;
- Demonstrativos como alternativas para treinamentos.

Um outro aspecto que envolve o perfil educacional nos ambientes comerciais é com relação ao suporte ao ciclo de desenvolvimento de um sistema de controle. Normalmente tal processo envolve etapas de análise e obtenção de resultados parciais que formam subsídios no projeto de um controlador. O usuário deve possuir conhecimento suficiente para o gerenciamento deste ciclo, filtrando e integrando as ferramentas necessárias e resultados relevantes. Sugere-se neste caso um ambiente onde os problemas são decompostos em pequenas etapas aonde o usuário ou estudante é orientado e guiado a observações relevantes [MUNTEANU et al., 1997]. Com este perfil os objetivos do ambiente podem estender-se ao treinamento e ser utilizado como ferramenta industrial na sintonia de malhas de controle, por exemplo.

Segundo o resultado de pesquisas [SILVEIRA, 1998], foram atribuídas às dificuldades em estudar teoria de controle em disciplinas afins, duas razões principais :

1. A falta de experiência do estudante nas indústrias. Eles têm pouca percepção dos sistemas de controle automático reais e dificuldades para entender conceitos básicos, como o da realimentação;
2. Há tantas derivações matemáticas e fórmulas, que o aluno, em geral fica desmotivado e cansado após ter cursado metade da disciplina; mesmo porque ele ainda não tem idéia de onde aplicar a teoria de Controle, o que controlar e como compensar um sistema.

Baseado em todo o contexto exposto, das deficiências e características educacionais negativas encontradas, surge a motivação na concepção de um novo ambiente CACSD. Este ambiente dever possuir características que, dentre outras coisas, possua um conteúdo instrucional e não apenas uma biblioteca de algoritmos e ferramentas isoladas.

2.2 Pesquisa e Prestação de Serviços

A teoria de controle sofreu grande avanço nos últimos 30 anos. Sistemas adaptativos, sistemas ótimos e sistemas de controle inteligentes e de aprendizado. Entretanto, o seu uso na prática se limita ainda a aplicações básicas, como controles proporcionais e PID. A principal razão para esta deficiência é a distância entre a teoria e a prática. É necessário que em paralelo ao contato com a teoria de sistemas de controle, o aluno ou pesquisador tenha contato com problemas de projeto de controladores no mundo real [MATKO et al., 1994].

Na indústria de processos as técnicas de controle se limitam apenas em algoritmos PID. No Brasil, segundo a ISA (*International Society for Measurement and Control*), 95% dos controladores industriais são PID [COGHI, 1996]. Destes, mais de 80% são mal ajustados e não operam adequadamente. Segundo Flaus em [FLAUS et al., 1994], existem dois principais fatores na dificuldade de transferência de tecnologia à indústria :

1. As pessoas da indústria são despreparadas no uso de algoritmos de controle avançado;
2. Vantagens de uma nova técnica de controle é difícil de se estabelecer para uma determinada aplicação industrial devido ao tempo necessário para implementar o algoritmo no Sistema de Controle Digital conectado ao processo. Em acréscimo, este trabalho de implementação normalmente não é compatível com a operação normal da planta.

A intensificação da pesquisa voltado aos problemas industriais fortaleceria os seguintes pontos:

- A variedade e qualidade de produtos são mais acentuados, consumo de energia, melhor desempenho da produção – são todos critérios que são melhor atendidos com estratégias de controle complexas. Muitos processos industriais apresentam características particulares de operação, requerimentos de sistema e desempenho que não permitem a implementação de metodologias clássicas de controle [ARRUDA et al., 1994];

- Pesquisadores, no contato com os problemas reais, teriam realimentação às suas propostas de linhas de pesquisa. Desta forma, as soluções de controle poderiam ser mais específicas ao processo ou planta.

Entretanto, a preocupação de aproximação da prática à teoria de controle em ambiente acadêmico e um maior direcionamento da pesquisa aplicada à indústria, requerem uma infra-estrutura adequada. Isto determina um perfil de laboratório voltado ao estudo de sistema de controle de sistemas físicos reais, a saber :

- Conter protótipos de plantas industriais;
- Conter diversos elementos de controle industriais;
- Possibilidade de separação ou identificação nas plantas de subprocessos caracterizados pela aplicação de problemas de controle específicos (clássicos ou complexos).

Esta estrutura, fonte de grande quantidade e heterogeneidade de equipamentos, processos e sistemas carece de uma plataforma CACSD integrada a mesma. Esta plataforma deve apresentar características de *hardware* e operacionais adequadas para permitir uma diversidade de experimentos seguindo as etapas de desenvolvimento de sistemas de controle : identificação, projeto do controlador, validação do controlador com o modelo identificado, programação de unidades de controle industriais, teste de controladores programáveis com o *hardware* na malha de controle simulada com o modelo identificado e finalmente, o controle direto da planta real pelas unidades de controle industriais [VIZJAK et al., 1994].

Para este suporte o ambiente deve ser capaz ou apresentar as seguintes características [FLAUS et al., 1994]:

- Ambiente de fácil utilização, interfaces amigáveis, recursos de simulação eficientes e metodologia que atenda as diversas fases no desenvolvimento do controlador;
- Teste e simulação de qualquer estratégia de controle. O ambiente deve permitir a fácil inserção de módulos correspondentes;
- Permitir o teste do novo algoritmo de controle numa planta industrial, isto é, abertura do ambiente em termos de *hardware*.

Existem, portanto, diferentes contextos ao qual uma ferramenta CACSD está inserida e integrada. Os pontos abordados neste item podem ser identificados na figura 1, onde propõe-se um modelo de sistema integrado ensino-pesquisa-extensão suportado por uma estrutura de laboratório com perfil adequado [LOURES et al., 1995].

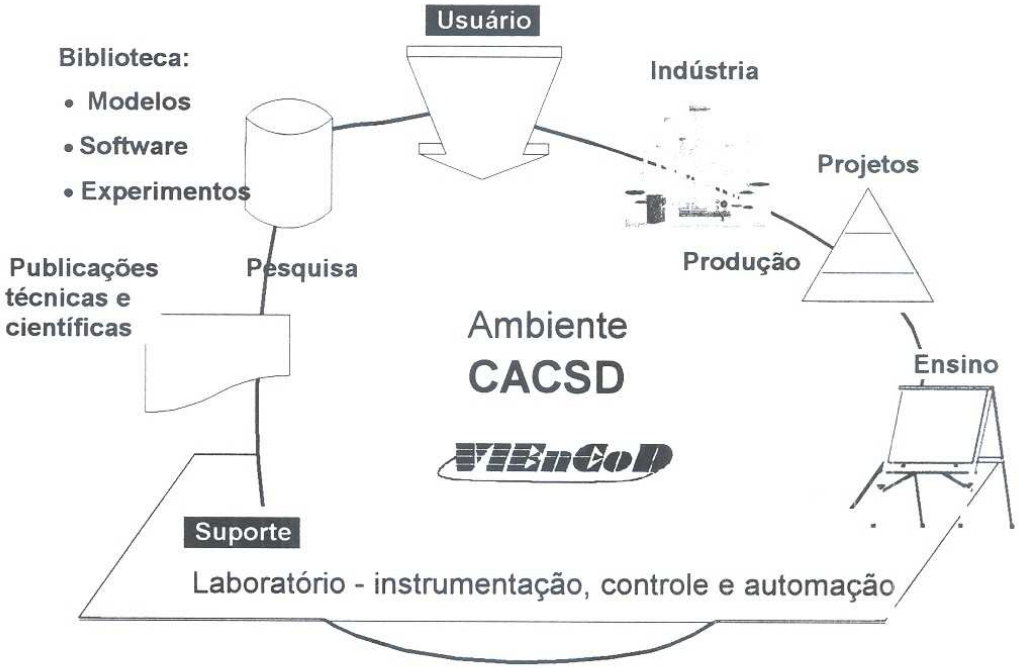


Figura 1 – Sistema Integrado Ensino-Pesquisa-Extensão

Neste modelo, os *usuários* são alunos da graduação, cursos de especialização, mestrado e também cursos de extensão; o pesquisador figura trazendo os problemas de controle de acordo com as necessidades da indústria. Desta forma são criadas condições para a geração de *publicações técnicas e científicas*. Esta atividade incrementa a biblioteca de modelos, dados experimentais e aplicativos que deve encontrar no ambiente uma estrutura de banco de dados adequado.

2.3 Cenário das Ferramentas CA (*Computer Aided*) no Setor Produtivo

Já na década de 80 o uso de computador como assistente no gerenciamento e execução de funções de manufatura foi bem estabelecido. O computador tem tido um papel significante no projeto de produto, engenharia de manufatura, gerenciamento de

materiais, controle e planejamento da produção e como elemento de controle e *otimização ao nível de chão de fábrica* [ALSÈNE, 1999].

O desenvolvimento da microeletrônica trouxe dispositivos programáveis, com alto grau de confiabilidade e possibilidade de comunicação com outros computadores ou mesmo dispositivos, permitindo a integração dos recursos no ambiente fabril.

Várias ferramentas CA surgiram no campo da indústria de manufatura, como CAD (*Computer Aided Design*). Estas ferramentas tem uso tanto no desenvolvimento de produtos ou organização de recursos de produção como no decorrer do trabalho de produção.

Os Sistemas Flexíveis de Manufatura - FMS (*Flexible Manufacturing System*) têm por objetivo permitir alta produtividade juntamente com flexibilidade na produção [MACIEL, 1997]. Isto significa diversificar a produção com o aproveitamento máximo dos recursos, bem como dotar o sistema de manufatura de flexibilização para contornar problemas ou falhas com máquinas.

A integração de todas estas ferramentas computadorizadas de planejamento, gerenciamento e controle com o ambiente industrial define o termo Manufatura Integrada por Computador - CIM (*Computer Integrated Manufacturing*). O CIM cobre muitas das tradicionais áreas da automação de manufatura incluindo CAD, CAM, MRP, PCP,... etc. Segundo [ALSÈNE, 1999] não é soma ou totalidade destes componentes mas é a ligação dos mesmos a um sistema interoperacional que irá satisfazer os objetivos e a estratégia de negócio do empreendimento.

A integração dos recursos computacionais, incluindo computadores de planejamento, computadores de controle, máquinas de produção e dispositivos programáveis só é possível devido ao estágio de desenvolvimento das redes de comunicação.

A produção de bens de consumo ou indústria de manufatura pode ser dividida funcionalmente em duas partes principais. Tem-se as atividades envolvidas diretamente

no processo de manufatura, como usinagem de peças, e as atividades de planejamento como projeto do produto, gerenciamento do fornecimento de matéria prima e controle de qualidade. Não é suficiente, sob um ponto de vista operacional, que as partes ou setores dentro do processo de produção trabalhem com o auxílio de computador de forma isolada para desempenhar suas tarefas.

O termo CIM significa integração total dos recursos computacionais, não apenas do ambiente de manufatura mas também do gerenciamento e desenvolvimento. O conceito escondido atrás do termo CIM é comunicação de dados entre sistemas computacionais, iguais ou não. Ou seja, a integração do que se considera 'ilhas' dentro do processo de produção. Por 'ilhas' entenda-se as unidades auto-suficientes no processo de produção. Estas unidades realizam funções de planejamento, engenharia e controle, atividades conhecidas através de siglas como CAE (*Computer Aided Engineering*) e CAM (*Computer Aided Manufacturing*).

Para possibilitar a visualização do fluxo de dados para a implementação de CIM deve-se trabalhar o conceito de hierarquia do sistema de controle de manufatura. A figura 2 mostra a hierarquia do sistema CIM com cinco níveis principais [MACIEL, 1997].

O nível 1 comporta os elementos das máquinas que interagem diretamente com o material de produção, como os sensores, atuadores e chaves. O nível 2 complementa a máquina, sendo composto pelos elementos inteligentes que controlam o comportamento desta, como CLP´s , comando numérico e dispositivos de controle e otimização. Os dois níveis são os responsáveis diretos pela realização de determinadas funções no processo de produção. São dispositivos que operam ao chão de fábrica. Neste nível, conforme será visto no item seguinte, é sugerido um ambiente CACSD - o VIEnCoD.

O nível 3 é o controle de célula. A célula representa um grupo de máquinas, sendo exercidas no nível 3 as funções de controle e monitoramento dessas. Os níveis 4 e 5 definem os níveis organizacionais, como planejamento, controle de qualidade, controle de matéria prima. Também encontram-se no nível 4 atividades como CAD e CAP (*Computer Aided Planning*).

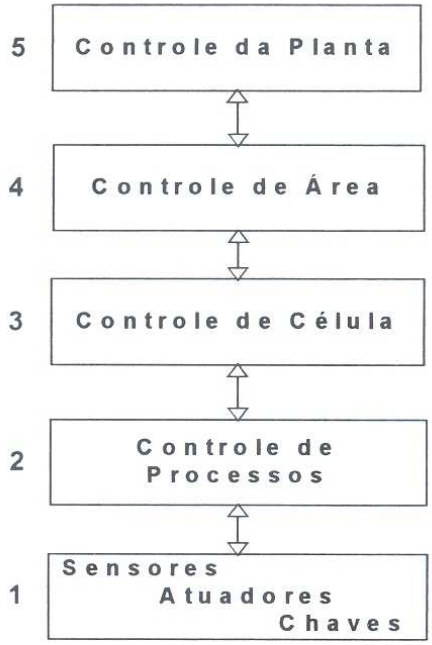


Figura 2 - Hierarquia de um Sistema CIM

A tecnologia CIM envolve várias ferramentas já automatizadas de engenharia de planejamento e controle, como os já citados CAE e CAM. A manufatura auxiliada por computador (CAM) e engenharia auxiliada por computador (CAE) concentram diversas atividades. Entre as atividades englobadas pelo CAE pode se citar:

- CAD (*Computer Aided Design*): desenho de projeto de peças.
- CAQ (*Computer Aided Quality Control*) e CAT (*Computer Aided Testing*): análise para controle de qualidade dentro de limites de tolerância.
- **CACSD (*Computer Aided Control System Design*)**: esta área refere-se ao projeto de sistemas de controle auxiliado por computador. Ambientes que disponibilizam uma série de ferramentas para análise, simulação de sistemas e otimização, com o objetivo de dar suporte ao ciclo de desenvolvimento de sistemas de controle, desde a modelagem e identificação da planta à síntese do controlador. Para atuação eficaz ao chão de fábrica esta ferramenta deve dispor de características operacionais e de *hardware* adequados.

2.4 Contextualização em um Sistema de Gestão Integrada

Primeiramente vale ressaltar que esta abordagem pretende, no escopo do presente trabalho, fornecer apenas um contexto motivacional não tendo sido objeto de implementações, conforme mencionado anteriormente. A relevância, interesse e tendência atuais sobre o tema descrito neste item confere ao trabalho um contexto que permite extensão em sua aplicabilidade, sugerindo sua continuidade futura nesta linha.

Conforme abordado no item anterior, os sistemas computacionais foram introduzidos nas indústrias para auxiliar e facilitar o uso e gerenciamento de recursos produtivos bem como nos vários outros setores da empresa (finanças, recursos humanos, planejamento da produção, gerência de materiais, vendas, entre outros).

Os Sistemas de Gestão Integrada surgiram pela necessidade de integrar a empresa como um todo, do ponto de vista gerencial, mas não considera a integração com as máquinas que realmente produzem.

Atualmente as empresas estão com seus interesses voltados à implantação dos Sistemas de Gestão Integrada ou ERP's (*Enterprise Resource Planning*). Sistemas computacionais de ERP, tais como o SAP R/3, tratam do gerenciamento integrado da cadeia logística : compra – fabricação – distribuição. Tais sistemas facilitam e organizam o acesso a informações, controlando o fluxo de informações e acelerando os processos de tomada de decisão nos níveis estratégico, tático e operacional [VOLMANN et al., 1997].

Entretanto, após a implantação torna-se necessário a integração com o chão de fábrica, onde são necessários métodos de otimização, controle e supervisão mais específicos ao ambiente da empresa. Comando numérico, robôs industriais, máquinas e processos, sistemas de manufatura flexível requerem alguma forma (estratégia) de controle para assegurar a operação adequada.

Neste contexto, uma ferramenta CACSD pode figurar como elemento de integração dos elementos finais de produção (plantas) aos níveis superiores em um sistema ERP. Para melhor entendimento da motivação do mesmo inserido em um Sistema de Gestão Integrada é necessário a contextualização do ambiente. A figura 3 ilustra uma Pirâmide Organizacional onde se observam diferentes níveis com específicos graus e formas de integração entre os mesmos.

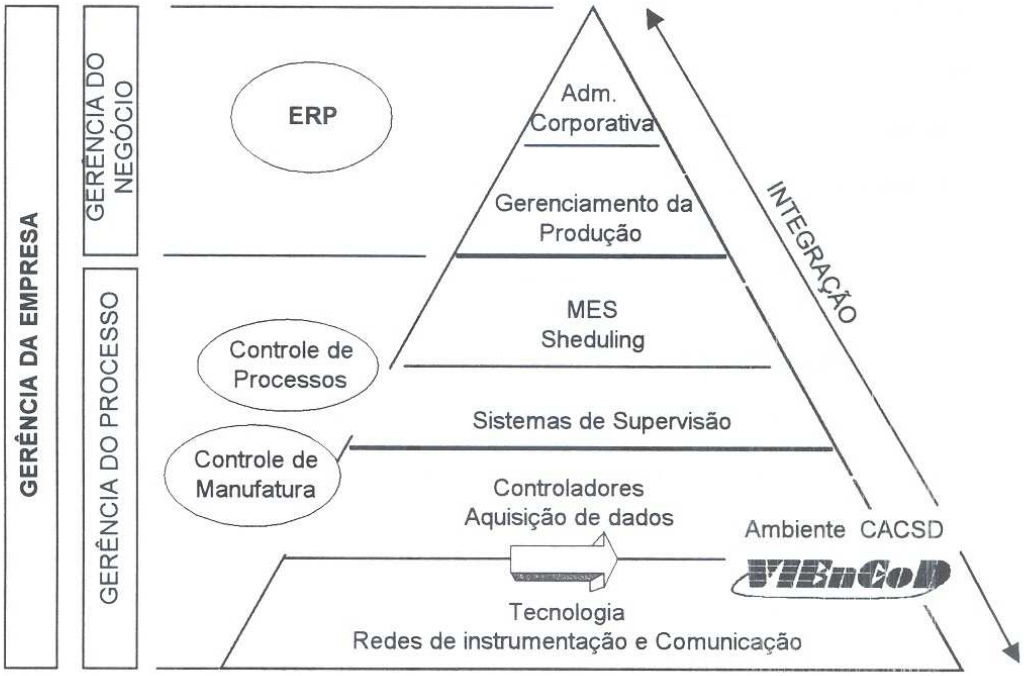


Figura 3 – CACSD no contexto de Gestão Empresarial Integrada

Os dois níveis no alto da pirâmide estão relacionados com a gerência do negócio englobando decisões (planejamento) gerenciais a longo prazo tais como : planejamento estratégico corporativo, planejamento de mercado, planejamento financeiro, estratégias de atendimento ao consumidor, planejamento de capacidades,... etc.

Os dois níveis intermediários englobam decisões ou planejamento de curto/médio onde encontra-se os sistemas MES (*Manufacturing Execution Systems*) com suas principais ferramentas : *Scheduling* (Programação Finita da Produção), Gerenciamento de Recursos de Produção e Gerenciamento de Qualidade. *Scheduling* é o problema de determinar uma alocação de recursos de manufatura no tempo para executar operações em uma fábrica, a fim de minimizar uma certa medida de custo. Ou seja, dado um

conjunto de ordens de produção, um sistema de *scheduling* aloca cronologicamente máquinas, materiais, operadores, ferramentas e outros recursos para executar operações fabris de maneira a melhor cumprir os prazos de entrega, minimizar estoques, e maximizar a produtividade e a utilização dos recursos.

Nos níveis inferiores da pirâmide encontram-se os equipamentos e sistemas de chão de fábrica responsáveis pela aquisição de dados e otimização de operação dinâmica de plantas industriais através de controladores. A supervisão e controle consiste em integrar todos os sinais de entradas e saídas dos processos industriais, possibilitando monitorar suas tendências, parametrizar e reconfigurar remotamente operações de produção. Neste nível propõe-se um ambiente CACSD com características operacionais e de integração específicas.

A característica de *hardware* deste ambiente deve possuir abertura de especificação, permitindo facilidades na configuração e operacionalização de um sistema de aquisição de dados adequado a cada planta ou célula de produção. O conceito de Instrumentação Virtual, a ser abordado posteriormente neste trabalho, permite o suporte desta característica.

Desta forma, esta plataforma agrega vantagens adicionais relacionadas ao sistema produtivo: menor tempo para readequação do sistema de aquisição acompanhando as mudanças de *layout* das plantas ou das linhas de produção; redução das paradas e custos de manutenção; informação *online* do *status* da linha de produção. A grande capacidade gráfica do ambiente permite o projeto de diversos tipos de gráficos e interfaces com o usuário disponibilizando maior fluxo de informação *online* da produção.

A necessidade da melhora de eficiência na operação de plantas visando produtos com melhor qualidade, maiores rendimentos e redução no consumo de energia tem levado as indústrias a adotar novas estratégias de controle de processos.

O projeto e implementação de sistemas de controle baseado em tais estratégias requer um conhecimento profundo do processo a ser controlado, o que geralmente requer o desenvolvimento de um modelo. Tal modelo muitas vezes não é trivial devido a

complexidade da planta. Desta forma, é necessário a identificação da planta por métodos que não alterem as condições naturais de operação das mesmas e, portanto, do processo produtivo.

De posse do conhecimento da planta, deseja-se, em geral, colocá-la em um ponto ótimo de operação para atendimento de índices de desempenho ditados pelas necessidades da produção vindas das camadas intermediárias e superiores da pirâmide.

Neste sentido, a sintonia das malhas de controle a nível de chão de fábrica é fundamental pois afeta de maneira direta ou indireta (através de todas as interfaces dos diversos níveis hierárquicos) fatores produtivos tais como : custos de manutenção dos equipamentos e paradas de emergência pela má operação dos mesmos, redução no consumo de energia, redução no uso de matéria prima,... etc.

Em suma, sob o contexto de Gestão Integrada exposto, a proposta de integração de um ambiente CACSD específico é possibilitar a identificação das características dinâmicas das diversas plantas ou processos colocando-os em pontos de operação ótimos tornando a sintonia das malhas de controle flexíveis ao atendimento de índices de desempenho ditados pelas necessidades da produção vindas das camadas intermediárias e superiores da pirâmide. Tal flexibilidade dever também atender às perturbações advindas da produção como a variação de demanda.

O VIEnCoD, conforme será visto no decorrer deste trabalho, com seu alto grau de flexibilidade e integração agrega de forma mais intensa resultados a nível de gerência de produção, pela atuação a nível de chão de fábrica. No capítulo 7 será explanado sobre os atuais trabalhos relacionados de forma *indireta* a este contexto e as perspectivas futuras em torno desta linha.

2.5 Pesquisas e Pacotes Comerciais – Revisão sobre o Estado da Arte

Os ambientes com proposta CACSD sofreram um grande desenvolvimento nos últimos anos onde ferramentas e algoritmos clássicos e avançados podem ser encontrados e utilizados sob características operacionais amigáveis. Com a popularização destas ferramentas no meio acadêmico e industrial, o repasse deste conhecimento pela comunidade científica a estes pacotes está cada vez mais efetivo[1].

A tabela 1 relaciona alguns exemplos de trabalhos e pesquisas feitas em torno de ambientes CACSD nos últimos três anos. São colocadas resumidamente as propostas de cada trabalho de maneira a verificar a tendência nos aprimoramentos e contribuições aos ambientes. Vale ressaltar também que muitos trabalhos realizados em Universidades e centros de pesquisa acabam se tornando produtos comerciais oferecidos por empresas prestadoras de serviço na área de controle. É o caso das soluções integradas baseadas nos principais pacotes como o Matlab por exemplo, como será visto logo adiante.

Instituição	Responsável(eis)	Tema
Departamento de Controle Automático, *Lund Institute of Technology*, Suécia	**K.J.Åström**, M.Johanssson, M.Gafvert	Ferramentas interativas para ensino de controle automático. Ambiente Matlab e Simulink [ÅSTRÖM et al., 1998].
Departamento de Controle Automático, *Lund Institute of Technology*, Suécia	**Björn Wittenmark**, Helena Haglung, Mikael Johansson	Figuras dinâmicas e aprendizado interativo em controle. Ambiente Matlab e Simulink [WITTENMARK et al., 1998].

Tabela 1 – Projetos de pesquisa realizados em torno de ambientes CACSD

[1] Conforme pode ser observado no *site* da *The Mathworks*, "*Matlab Connections*", http://www.mathworks.com/products/connections, Out. 1999.

Instituição	Responsável(eis)	Tema
Departamento de Engenharia Química, Universidade de Delaware, Newark Escola de Engenharia Química, Univerdidade de Purdue, Indiana.	**Francis J. Doyle III**, Edward P. Gatzke, Robert S. Parker V. Venkatasubramanian, T. A. Kendi	Estudos de caso práticos para abordagem de controle e dinâmica de processos à graduação utilizando módulos de controle de processos. Ambiente Matlab e Simulink [DOYLE et al., 1996] [DOYLE et al., 1998].
Laboratoire d'Automatique de Grenoble, France	D. Munteanu, F. Michau, S. Gentil	Ambiente de aprendizado em Controle Automático baseado em simulação. Proposta de "Diagrama de Conhecimento": tópicos de controle disponíveis em ambiente orientado a objeto [MUNTEANU et al., 1997].
Instituto de Automação e Engenharia de *Software*, Universidade de *Stuttgart*, Alemanha.	M. Seidel	Sistema tutor baseado em hipertexto para ferramenta CACSD. Apoio ao pouco conteúdo informativo e teórico dos ambientes CACSD [SEIDEL, 1994].
Departamento de Engenharia Elétrica e Ciência da Computação, Universidade de *Wisconsin, Milwaukee*, EUA.	Brian Armstrong	Programa de laboratório de controle com enfoque à Identificação. Propostas de experiências com elementos físicos à simulação [ARMSTRONG, 1997].
Departamento de Automação e Informática, *Politecnico di Torino*, Itália	**S. Carabelli** , A . Argondiza, B. Bona	Sistema de desenvolvimento de controladores e filtros digitais baseados em Matlab e DSP. Simulação com HIL e conceito de Instrumento Virtual[2].

***Tabela 1** (continuação)* – Projetos de pesquisa realizados em torno de ambientes CACSD

[2] Trabalho apresentado no site da The Mathworks, "Matlab Connections", http://www.mathworks.com/products/connections, Out. 1999.

Instituição	Responsável(eis)	Tema
Departamento de Engenharia Elétrica, *University of North Carolina*, EUA.	B. Lanier, L. Ross, Y.Kakad	Lógica Fuzzy para controle de processo. Síntese de controlador Fuzzy através de Matlab/Simulink e implementação em controlador industrial [LANIER et al., 1997].
Universidade de Paderborn, dSPACE GmbH, Alemanha	H. Hanselmann	Simulação com *Hardware-in-the-Loop* (HIL) integrada a um ambiente CACSD baseado em Matlab [HANSELMANN, 1996].
Mechatronics Laboratory Paderborn - MlaP, Universidade de Paderborn, Alemanha	**M. Wolf** J. Lückel	Estruturas para ambiente CACSD voltado à sistemas mecatrônicos. Abordagens das linhas de pesquisa : simulações com HIL, linguagem orientada a objeto para modelagem de sistemas, sistemas em tempo real [WOLF, 1998].
Departamento de Automação, Universidade Técnica da Dinamarca	**Ole Ravn**, Anders Pjeturson	Projeto COPERNICUS: Modelagem e controle de manipuladores robóticos. CACSD voltado a sistemas mecatrônicos; desenvolvimento de MSL (*Mechatronic Simulink Library*). Baseado em Matlab/Simulink, Maple e Mathematica [RAVN et al., 1995].
Laboratório de Controle Automático, Instituto Federal da Suíça de Tecnologia	W. Schaufelberger	Projeto Leporello : ambiente CACSD para suporte do ciclo de projeto de sistemas de controle. Programação orientada a objeto e técnica de "árvore ativa" para acesso às etapas do ciclo[2].

***Tabela 1** (continuação)* – Projetos de pesquisa realizados em torno de ambientes CACSD

[2] Trabalho apresentado no site da The Mathworks, "Matlab Connections", http://www.mathworks.com/products/connections, Out. 1999.

Instituição	Responsável(eis)	Tema
Departamento de Engenharia Elétrica, Universidade de Hull, Inglaterra	R. J. Patton	Proposta de *toolbox* para Matlab voltado a controle robusto no Domínio do tempo e frequência via otimização *eigenstructure*[2].
Institute of Automation, Univ. of Magdeburg, Alemanha	Equipe Lab.	*Toolbox* (Matlab) para otimização multi-objetivo[2].
University of Bologna, Bologna, Itália	Prof. Claudio Bonivento	*Toolbox* para análise de máquinas e mecanismos complexos, sintonia de algoritmos de controle. Baseado em Matlab-Simulink[2].

***Tabela 1** (continuação)* – Projetos de pesquisa realizados em torno de ambientes CACSD

O contexto atual de produção científica sob análise conduz a algumas colocações e conclusões importantes:

- A preocupação no ensino de controle impulsiona as linhas de pesquisa ao desenvolvimento de ambientes de análise e simulação mais amigáveis;
- Estes tornam-se ferramentas indispensáveis ao ensino-pesquisa na Universidade. A disseminação neste meio é bastante acentuada, como pode-se observar com o Matlab, Vissim,... etc;
- Em paralelo, desenvolvem-se também os sistemas *hardware* (aquisição e processamento) para interfaceamento com os ambientes, criando-se as chamadas *soluções integradas*. A *The Mathworks*, por exemplo, apresenta um grande número de parcerias com empresas para provimento de serviços e produtos baseados no Matlab e Simulink. Um exemplo é o convênio firmado entre esta e a *Hewlett-Packard* no sentido de prover interface da instrumentação HP-VXI diretamente com o ambiente MATLAB. Mais adiante serão analisadas mais algumas destas soluções;

[2] Trabalho apresentado no site da The Mathworks, "Matlab Connections", http://www.mathworks.com/products/connections, Out. 1999.

- Existe a preocupação no conceito de simulações com HIL (*Hardware-in-the-Loop*) onde participam elementos físicos no ciclo de desenvolvimento de controladores. Ambientes de simulação em tempo real é também uma preocupação;
- Desta forma, estratégias de controle mesmo que complexas são facilmente tratadas e utilizadas;
- Existe a preocupação em se dotar laboratórios com protótipos reais de problemas encontrados na indústria;
- A reunião dos pontos acima possibilita um cenário propício a aproximação da Universidade com a Indústria. Os problemas reais de controle encontrados fornecem subsídio à pesquisa aplicada;
- Esta aproximação resulta na popularização dos ambientes CACSD também na indústria, principalmente a indústria automobilística e de processos;
- Neste contexto, as soluções integradas despontam como ferramentas de apoio ao desenvolvimento de sistemas de controle tanto a nível de pesquisa no meio acadêmico quanto a nível de aplicação industrial (muitas vezes como resultado desta pesquisa). Sintonia de malhas de controle podem ser conseguidas através de soluções baseadas em Matlab, por exemplo.

As soluções integradas de *hardware* e *software* fazem surgir novos conceitos e metodologias relacionados ao desenvolvimento de sistemas de controle. A proposta da DSPACE, outro exemplo de parceria com a *The Mathworks*, baseia-se nestes conceitos. A DSPACE é uma empresa com origem na Universidade de Paderborn, na qual são desenvolvidos soluções em *hardware* (microprocessadores – DSP's) e periféricos (sistema de aquisição) integrados ao ambiente Matlab-Simulink. Este foi o resultado da integração da Universidade à indústria automobilística onde linhas de pesquisas direcionaram-se aos problemas reais (p. ex. suspensão ativa).

Em suma, os ambientes com proposta CACSD mais difundidos apresentam estas soluções integradas de *hardware* através de parcerias, proporcionando sistemas de simulação em tempo real de sistemas de controle com elementos físicos.

A seguir é feita uma abordagem dos pacotes comercias com proposta CACSD mais difundidos, observados neste trabalho, no sentido de verificar suas características, deficiências e propostas. Algumas das soluções integradas existentes à cada ambiente são comentadas. O produto deste estudo forneceu subsídios a concepção e desenvolvimento da estrutura operacional e de *hardware* do VIEnCoD. Vale aqui ressaltar que em determinadas situações algumas idéias e concepções do VIEnCoD foram implementadas previamente a lançamentos comerciais. É o caso da parceria entre a HP e a Mathworks, citada acima, oficializada no início deste ano. Portanto, o conceito é recente mas com perspectiva de muita atenção pela comunidade científica e empresas provedoras de soluções.

Será dedicada uma abordagem mais profunda sobre os recursos do Matlab e Simulink, plataforma utilizada no VIEnCoD. Tal abordagem se estende nos capítulos posteriores na descrição da concepção, projeto e implementação.

2.5.1 Matlab - Simulink

2.5.1.1 Descrição

O Matlab é um ambiente de computação técnica integrada que combina computação numérica, recursos gráficos avançados e linguagem de programação de alto nível. Existem elevado número de ferramentas e funções disponíveis para : Análise de dados e visualização; computação simbólica e numérica; gráficos científicos e de engenharia; modelagem, simulação e criação de protótipos; programação, desenvolvimento de aplicações e criação de interfaces gráficas do usuário (GUI's).

Os *Toolboxes* são um conjunto de funções Matlab (*m-files*) específicas de maneira a estender o ambiente à classe de problemas particulares. Algumas das áreas que apresentam-se sob a denominação de *toolbox* são: sistemas de controle, processamento de sinais, lógica fuzzy, identificação e otimização de sistemas e outras.

O sistema Matlab consiste de cinco partes pricipais :

1. *Linguagem* : É uma linguagem matricial de alto nível com controle de fluxo de sentenças, funções, dados estruturados, entrada/saída e ferramentas de programação orientada a objeto. As chamadas *m-files,* não precisam ser compiladas permitindo depuração e edição automáticos, nem declaração de variáveis permitindo vários tipos de dados – desde escalares e matrizes de tipos de dados híbridos à matrizes multidimensionais.
2. *Ambiente operacional* : Facilidade no gerenciamento de variáveis através do chamado *workspace*, importação e exportação de dados. Ferramentas para desenvolvimento, gerenciamento, depuração (*debug*) e *profile* de *m-files* (para análise de tempos de processamento).
3. *Suporte gráfico* : Inclui comandos de alto nível para visualização de gráficos 2D e 3D, processamento de imagem, animação e gráficos de apresentação. Ferramentas para suporte completo à construção de interfaces gráficas do usuário (GUIs).
4. *Bibliotecas de funções matemáticas* : Vasta coleção de algoritmos computacionais desde funções mais simples a mais sofisticadas como FFT.
5. *Interface de programa de aplicação* : A API (*Application Program Interface*) permite que se escreva programas em C e FORTRAN que interagem com Matlab.

2.5.1.2 Simulink

Suportado pela funcionalidade numérica, gráfica e numérica do Matlab, o Simulink prove um ambiente de simulação gráfica intuitiva para projeto de sistemas dinâmicos e desenvolvimento de estratégias de controle. É um ambiente de construção de protótipos para modelagem e simulação de sistemas dinâmicos através de diagramas de blocos. Suporta sistemas lineares e não lineares, modelados em tempo contínuo e discreto ou ambos (sistema híbrido). Neste caso a saída dos blocos discretos são mantidos a nível constante entre os intervalos de amostragem. Existe também a possibilidade de sistemas onde coexistem taxas de amostragem diferentes (*multirates*) e multivariáveis.

O Simulink contém uma vasta biblioteca de blocos que suportam a construção e simulação de modelos. Alguns blocos e funções são: geração de sinais; visualização e

armazenagem de dados; descrição de componentes discretos; descrição de funções lineares e não lineares; gerenciamento e interface de modelos e sinais; *blocksets* que são blocos que realizam funções especiais, similarmente aos *Toolbox* no Matlab.

Os modelos podem ser construídos de forma hierárquica, encapsulando e integrando subsistemas de forma a dar maior grau de especificação ao mesmo. Para tal são utilizadas recursos como as *Masks* e *S-Functions*. As *Masks* são subsistemas construídos através do encapsulamento de blocos ou outros subsistemas com entradas e saídas específicas. A descrição do sistema criado é feita por especificação dos parâmetros através de interface customizada. A figura 4 ilustra uma *Mask*. Uma *S-Function* é um recurso de criação de subsistemas obtido por uma linguagem de descrição de sistemas dinâmicos. *S-Functions* podem ser escritas através de linguagem Matlab (*m-file*) ou C. Desta forma, equações matemáticas e funções que descrevem o comportamento de um sistema dinâmico podem ser encapsulados e integrados ao ambiente Simulink sob a forma de blocos.

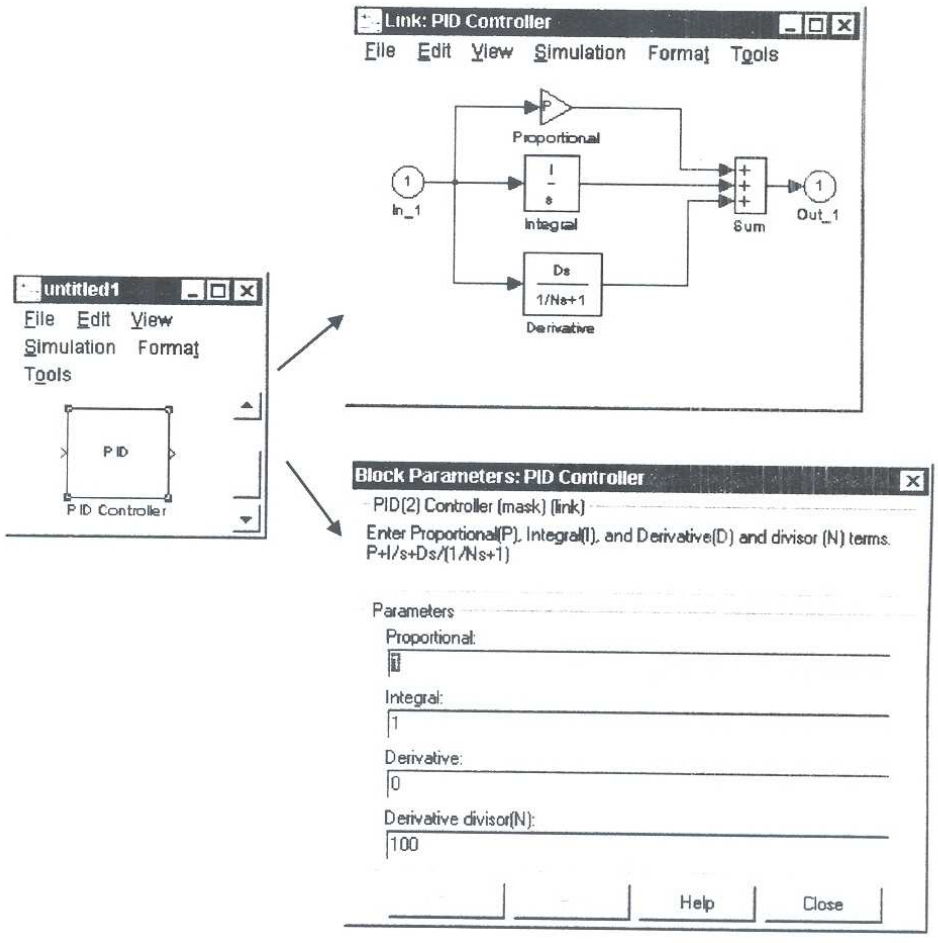


Figura 4 – Descrição de subsistemas através do *Mask*

Como o Simulink é integrado ao Matlab, os modelos e sistemas sob a forma de diagrama de blocos podem ser gerados, analisados, otimizados e gerenciados por ferramentas baseadas em Matlab. Isto propicia um ambiente bastante flexível em termos de recursos para o Ciclo de Desenvolvimento de Sistemas de Controle, como será visto adiante neste trabalho.

Através de diagramas Simulink é possível a geração automática de código C através do *Real-Time Workshop* (RTW), permitindo a execução de sistemas contínuos, discretos e híbridos em diversos tipos de plataformas de *software* e *hardware* (incluindo sistemas em tempo real). Algoritmos de controle, processamento de sinais e sistemas dinâmicos podem ser implementados sob a forma de diagrama de blocos e após obtido o respectivo código C e exportado a outros ambientes. Estas características permitem ambientes de simulação com *hardware-in-the-loop* onde participam da simulação elementos físicos conectados a um sistema implementado em Simulink. Maiores detalhes sobre estes recursos são abordados na descrição do dSPACE, solução integrada baseada em Matlab e Simulink.

2.5.1.3 Análise

O Matlab é o ambiente de computação numérica muito difundido entre cientistas e engenheiros. A maior parte da produção científica e aplicativos atuais na área de sistemas de controle são baseados neste ambiente ou utilizam-se de suas ferramentas. Desta forma, as contribuições ao seu desenvolvimento tanto a nível de novos *toolbox* e aplicativos, quanto à suas característica operacionais, intensificam-se cada vez mais.

Este contexto o torna o ambiente mais atrativo no desenvolvimento de um projeto. A própria disseminação no meio acadêmico, já a nível de graduação, permite um contínuo intercâmbio e continuidade de pesquisas. Estudantes com bons conhecimentos técnicos de controle podem direcionar esforços no aprimoramento deste conhecimento e não no desenvolvimento e adaptação de ferramentas e ambientes computacionais proprietários.

Grande parte dos recursos da plataforma Matlab e Simulink sumariamente descritos acima foram utilizados no referido trabalho. No capítulo 4 serão exploradas de forma

mais intensa as características técnicas e operacionais utilizadas no projeto VIEnCoD explicitando as necessidades de recursos. A metodologia de integração de alguns *toolboxes* e componentes externos compõem a estrutura de suporte do ambiente, como será visto nos capítulos posteriores. Desta forma, é criado um ambiente CACSD específico gerenciado sob ambiente Matlab e Simulink através de *m-files* e GUIs. Sob esta ótica temos o VIEnCoD como um *toolbox* adicional.

O grau de utilização das ferramentas e ambiente Matlab e Simulink pode ser observado pela quantidade de serviços, produtos e soluções baseadas nestes. No *site da The Mathworks*[3], encontramos estas soluções então chamadas como *Matlab Connections.* São inúmeras as áreas de aplicação onde algumas mereceram destaque por terem ligações ao referido trabalho. Por exemplo: Análise e projeto de sistemas de controle; Identificação de sistemas; Integração de sistemas; Monitoramento e Controle de Processos; Desenvolvimento de aplicações; Programação Matlab; Instrumentos e testes de equipamentos; Sistemas em tempo real; Robótica; Sistemas Automotivos; Sistemas Petroquímicos; Instrução educacional; Ferramentas instrucionais e de treinamento; outras.

2.5.1.4 Soluções Integradas

- ***dSPACE***

O dSPACE provê um ambiente de desenvolvimento de sistemas de controle baseado em duas metodologias : RCP (*Rapid Control Prototyping*) e simulação HIL (*Hardware-in-the-Loop*).

Basicamente o RCP permite a aplicação e teste de novas estratégias de controle em *hardware* sob tempo real de maneira rápida e amigável, para validação do controle antes de sua aplicação em dispositivo industrial. Desta forma, pode-se observar o desempenho do sistema (planta física inserida na malha de controle) a qualquer fase no ciclo de projeto adequando-se critérios e parâmetros. A chave do RCP é a geração e implementação automática de código do algoritmo de controle de um diagrama de blocos em um *hardware* de tempo real. Ao *hardware* carregado com a estratégia de controle e/ou

[3] The Mathworks, "Matlab Connections", http://www.mathworks.com/products/connections, Out. 1999.

modelo (representado computacionalmente) de um subsistema da planta denomina-se de ECU (*Electronic Control Unit*).

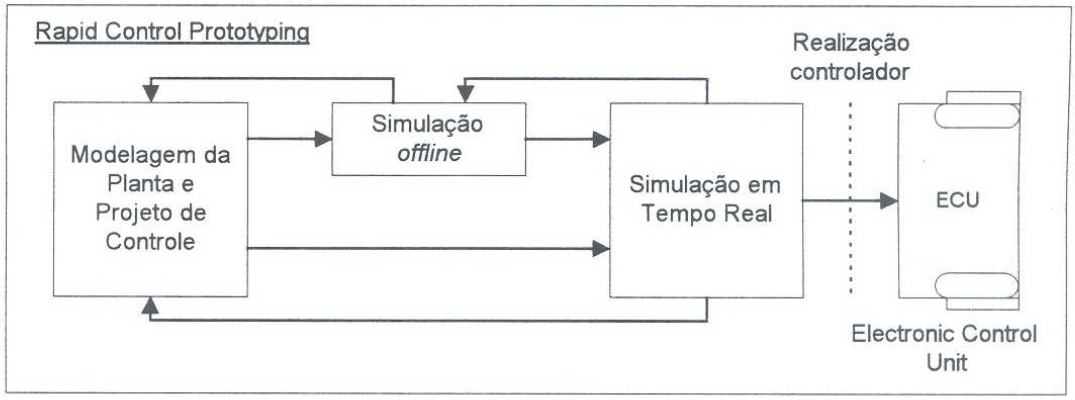


Figura 5 – Processo RCP

A simulação com HIL complementa este conceito. Uma planta pode ser subdividida em subsistemas e implementados computacionalmente através de modelagem matemática. No processo de interfaceamento do protótipo (planta) físico no ciclo de desenvolvimento do controlador, muitas vezes é interessante a participação de alguns elementos físicos e outros implementados na ECU. Situações de teste perigosas ou dispendiosas enquadram-se neste contexto. Elementos difíceis para modelagem e simulação permanecem fisicamente na malha. Com o desenvolvimento do ciclo os elementos matemáticos vão sendo substituídos por elementos físicos. Desta forma, simulações em tempo real com modelos e elementos físicos podem se realizados. A figura 5 representa este processo.

O processo de desenvolvimento de sistemas de controle é contemplado pela integração de recursos Matlab e Simulink e um *hardware* específico. A identificação de sistemas e modelagem baseiam-se no Matlab; o Simulink é utilizado para simulação *off-line*; o RTW (*Real-Time Workshop*) para geração automática de código C. Todas as especificações do sistema de aquisição (*I/O*) bem como as *interrupções* são definidas no ambiente Simulink. Isto se deve a uma estrutura desenvolvida para tratamento do código gerado pelo RTW para interfaceamento com o *hardware* específico. A figura 6 ilustra este processo.

As opções de *hardware* incluem placas DSP, placas Alpha de alta velocidade e soluções de multi-processadores. Juntamente com placas específicas de *I/O* o poder de processamento atende as especificações de tempo real requeridas.

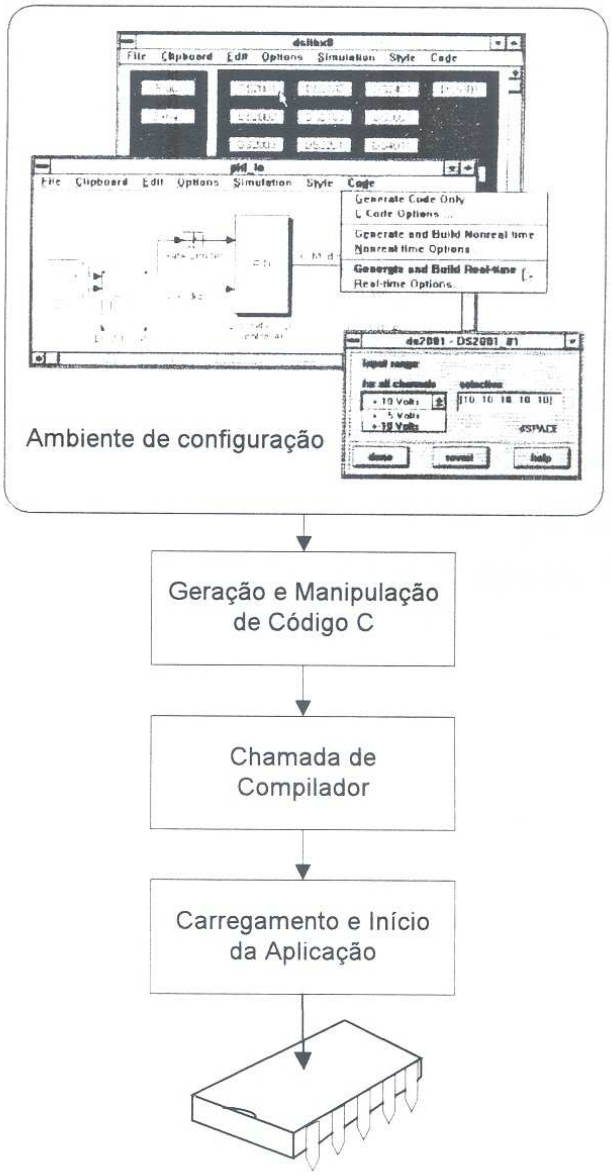


Figura 6 – Ciclo dSPACE

Um ambiente de desenvolvimento de interface gráfica do usuário (GUI) permite a configuração de painéis de controle e supervisão do experimento. Os parâmetros definidos no diagrama Simulink ou função Matlab são conectados a estes painéis. Nestes insere-se o conceito de Instrumentação Virtual as ser trabalhado no capítulo 4. Modificações em parâmetros do sistema podem ser feitos sem interrupção do experimento

sob condições de tempo real. Adicionalmente permite-se o desenvolvimento de interface com monitoração gráfica de sinais e recursos de armazenagem de dados.

- *Análise:* A plataforma proposta teve origem no laboratório de mecatrônica da Universidade de Paderborn, Alemanha. Portanto, a concepção de *hardware* e *software* se baseia nos conceitos RDP e HIL necessários ao ciclo de desenvolvimento de sistemas mecatrônicos. Nestes sistemas existe a preocupação de processamento em tempo real com sistema de aquisição e processamento de alto desempenho. Toda a solução voltada ao *hardware* foi desenvolvida sendo, portanto, uma solução proprietária. A simulação e projeto de controle é baseada nas ferramentas Matlab-Simulink e a simulação HIL utiliza-se do recurso RTW estendido por rotinas específicas ao *hardware*. A monitoração e controle do experimento é feito por interfaces externas ao ambiente Matlab com o conceito de Instrumentação Virtual. O VIEnCoD apresenta uma proposta semelhante[4] utilizando-se da metodologia de integração proposta, do conceito HIL e Instrumentação Virtual, mas sem *hardware* proprietário e maior flexibilidade na implementação de interfaces de monitoração e controle.

♦ ***WinCon***

WinCon é um aplicativo de 32 bits Windows que processa em tempo real o código de diagrama Simulink gerado através do RTW de forma a implementar controle em malha fechada e processamento digital de sinais em tempo real em um PC equipado com placa de controle e aquisição de dados. Mudanças de parâmetros e gráficos em tempo real podem ser feitos diretamente através de diagramas Simulink ou comandos Matlab. Vale também ressaltar a possibilidade de executar controladores independentes do ambiente Simulink e seus componentes. Outros aplicativos Windows podem ser utilizados simultaneamente sem prejuízo do funcionamento do controlador.

O Wincon consiste em sua estrutura de um cliente e um servidor referidos como WinCon W95Client e WinCon W95Server. Cada servidor pode comunicar com muitos

[4] Esta abordagem será detalhada no capítulo 4.

clientes. Um PC pode ter um cliente e um servidor operando ao mesmo tempo. O W95Server desempenha tarefas como : converter diagrama Simulink em código em tempo real executável em PC; compila e conecta o código usando Visual C++; descarregar o código executável para processar no cliente; controlar a operação do cliente; manter a comunicação do cliente com Simulink para mudança *online* de parâmetros; apresentar gráficos em tempo real; salvar dados em disco. O W95Client é um componente de *software* em tempo real que processar o código gerado em taxa de amostragem definida pelo usuário. A estrutura baseada em um PC é ilustrada na figura 7.

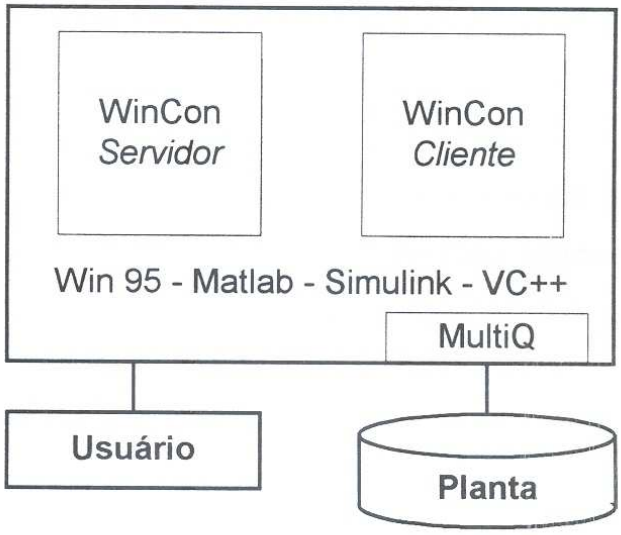


Figura 7– Estrutura WinCon

- *Análise:* A grande vantagem do Wincon é que o mesmo pode operar em ambiente Windows com frequência de amostragem de até 15 KHz, tornando-se uma alternativa para aos caros e dedicados *hardware* DSP. A filosofia no suporte ao desenvolvimento de sistemas de controle é análogo ao dSPACE: Simulink-RTW permitindo simulações com HIL. Entretanto, dispensa um *hardware* proprietário.

2.5.2 VisSim

2.5.2.1 Descrição

O VisSim é um programa baseado em Windows para a modelagem e simulação de sistemas dinâmicos através de interface baseada em diagrama de blocos. O ambiente de simulação provê projeto de sistemas lineares, não lineares, contínuos, discretos, variantes no tempo, híbridos, SISO e MIMO. Suporta o projeto hierárquico onde os subsistemas podem ser encapsulados em blocos criando novos sistemas. Algoritmos de controle definidos pelo usuário codificados em C, Fortran, Pascal e até Matlab (*m-files*) se enquadram nesta estrutura podendo ser encapsulados em blocos.

Ferramentas de linearização aproximam a dinâmica de um sistema não linear pela linearização em torno de um ponto de operação específico. Sistemas linearizados podem ser representados por equações de espaço de estados e função de transferência.

Técnicas clássicas para análise e projeto de compensadores são disponíveis, tais como Lugar das Raízes, Bode, Nyquist,... etc. Algoritmos PID pré-configurados podem ser parametrizados de acordo com a necessidade de projeto.

Recursos de otimização de sistemas lineares e não lineares sujeitos a restrições permitem, por exemplo, a determinação de parâmetros ótimos PID sob especificação de funções custo baseadas no tempo como erro em regime estacionário, sobressinal, tempo de subida,... etc. O método se baseia em algoritmo gradiente reduzido e generalizado conhecido como GRG2[5].

O VisSim apresenta recurso para implementação de simulação em tempo real com HIL, aquisição de dados e controle diretamente do Windows. A plataforma de *hardware* é aberta : placas *I/O* para PC, cartões de interface de controle de alta velocidade de motores, conexões de porta serial a CLP's (Controladores Lógico Programáveis) ou SDCDs (Sistemas Digitais de Controle Distribuídos). Além disto, diagramas podem ser

[5] Sua referência pode ser encontrado no site da *Visual Solution*, http://www.vissim.com, Out. 1999.

codificados em ANSI C permitindo a abertura a outra plataforma baseada neste compilador.

2.5.2.2 Análise

O VisSim com seu ambiente amigável de interface baseado em diagrama de blocos, aliado a especialização para modelagem e simulação de sistemas dinâmicos complexos, apresenta uma alta penetração no meio industrial. As facilidades e potencialidades na construção de protótipos e plataformas de teste com simulação HIL são análogas as ferramentas relacionadas ao Simulink. A disponibilização de condições em tempo real em simulações com HIL e abertura do ambiente a *hardware* somente foi contemplado pelo Matlab em sua última versão (5.3) através do *Data Acquisition Toolbox*. A estrutura baseada em diagrama de blocos interfaceada com sistemas de aquisição o torna muito semelhante com o *LabView* da *National Instruments*[6], ambiente de desenvolvimento de Instrumentação Virtual. Da mesma forma, o recurso de desenvolvimento de aplicativos *stand-alone* (executáveis, sem necessidade da licença instalada) veio somente com a versão 5.3 do Matlab, sob critérios especiais de aquisição impostas pela *The Mathworks*. A possibilidade de interface e integração com o Matlab o torna bastante atrativo para simulações de estratégias de controle complexas sob a forma de diagramas de bloco. A abordagem de identificação não apresenta suporte específico no Vissim. Em suma, é uma ambiente análogo ao Simulink, com maior direcionamento a sistemas de controle, maior facilidade operacional (recursos gráficos, implementação de protótipos com simulação HIL,... etc) e maior penetração na indústria.

A *Windward Technologies* é a empresa responsável pelo desenvolvimento dos recursos e ferramentas de otimização presentes no VisSim. Não é divulgado a existência de outras parcerias, ao contrário da *The Mathworks* com a plataforma Matlab.

[6] Sistema analisado no capítulo 4.

2.5.3 MOOMo - CAMeL - IPANEMA

2.5.3.1 Descrição

Para o desenvolvimento de sistemas de controle voltados a sistemas mecatrônicos é proposto um estrutura integrada composta de ambientes dedicados ao suporte de cada fase dentro de um ciclo proposto. Tal estrutura é ilustrada na figura 8.

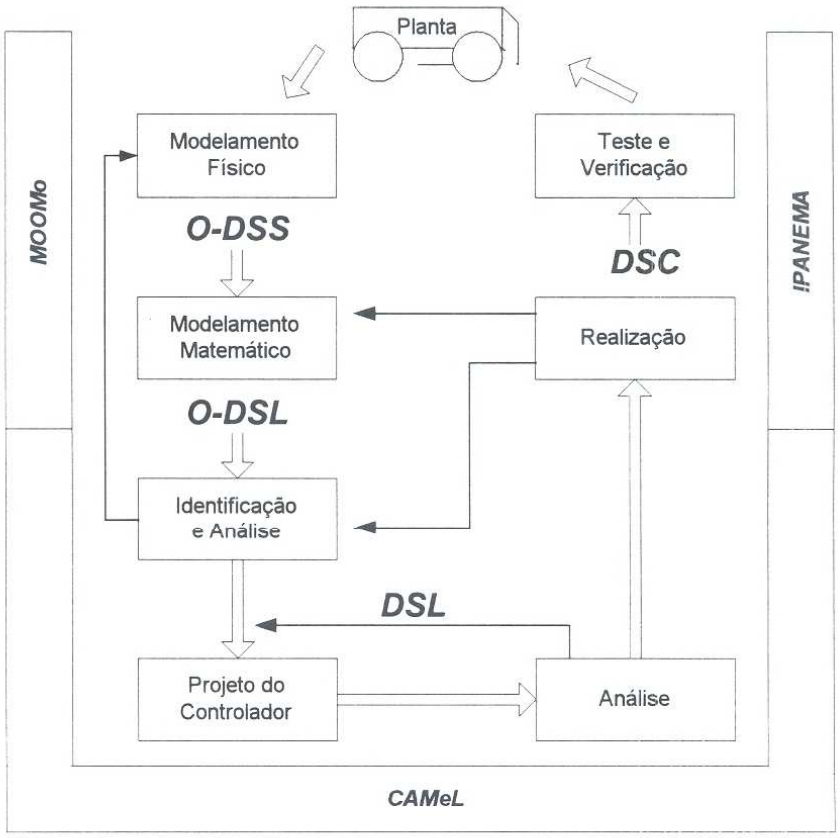


Figura 8 – Integração ferramentas sistemas Mecatrônicos

O projeto de sistemas mecatrônicos envolve a integração de elementos (funções, modelos) pertencentes a disciplinas diferentes como mecânica, elétrica, hidráulica e processamento de informação. Este integração multidisciplinar eleva a complexidade do sistema exigindo ferramentas apropriadas para a modelagem (MOOMo), análise e projeto de controlador (CAMeL) e realização e testes de validação do controlador (IPANEMA).

Na descrição de modelos e organização de sistemas mecatrônicos são disponíveis linguagens de descrição : *Dynamic System Structure* (**DSS**), *Dynamic System Language* (**DSL**) e *Dynamic System Code* (**DSC**). Os subsistemas individuais (classes no conceito de orientação a objeto) são modelados e interconectados por meio destas linguagens de descrição.

O primeiro nível cobre a descrição de disciplina específica (no senso da mecatrônica : engenharia de controle, de sistemas, ciência da computação) das diferentes partes de um sistema mecatrônico e suas interconexões [ZANELLA, 1996]. Cada disciplina técnica tem a sua maneira específica de descrição do sistema. A linguagem **DSS** é capaz de integrar todos estes formalismos de descrições específicas em um método para modelagem do sistema mecatrônico inteiro. De forma a analisar, simular e otimizar um sistema mecatrônico através das ferramentas do ambiente CAMeL, a descrição do sistema dever ser abstraída em um formalismo matemático – um sistema de equações diferenciais ordinárias (ODE) de ordem 1. A linguagem de descrição neste nível é a **DSL**.

O terceiro nível oferece a descrição do sistema orientado a *cálculo* e *máquina* na base da linguagem de descrição **DSC**. As descrições DSC são automaticamente gerados pela linguagem DSL e são processadas através do carregamento de uma ferramenta computacional de distribuição de forma a preparar uma implementação do sistema em *hardware* multiprocessador.

A seguir cada ambiente é descrito de forma mais específica dando a idéia da estrutura de suporte ao ciclo.

2.5.3.2 MOOMo

Destina-se a estrutura do processo de modelagem, desde a formulação do modelo (físico) específico a uma disciplina à derivação do modelo matemático. Para tal, combina elementos de orientação a objetos e teoria de grafos e prevê os pré-requisitos para integração de formalismos (algoritmos) para derivação do modelo matemático. Permite, portanto, uma descrição baseada em objeto de sistemas hierárquicos modulares.

Seu componente principal é a linguagem de descrição *Objective-DSS (Objective – Dynamic System Structure)* onde os sistemas mecatrônicos são definidas por classes específicas a disciplinas e verificados para correta sintaxe e semântica.

Os sistemas hierárquicos modulares podem ser montados através de elementos básicos e sistemas acoplados de acordo com um estrutura em árvore. Esta classes de elementos formam a base da descrição de modelo MOOMo. Este ambiente permite a geração de códigos para modelagem matemática em DSL (*Dynamic System Language*), linguagem utilizada no CAMeL.

2.5.3.3 CAMeL

O CAMeL (*Computer-Aided Mechatronic Laboratory*) é um ambiente CACSD *aberto* direcionado ao desenvolvimento de sistemas mecatrônicos. Suporta a análise, síntese e realização de sistemas lineares e não-lineares. É composto de uma linguagem de descrição de sistemas dinâmicos DSL (*Dynamical System Language*) e outras ferramentas, como : simulador de sistemas não lineares (SIMEX); otimizador de parâmetros de sistemas não lineares (OPIDEX); ferramentas para sistemas lineares (LINTOOL); simulador de sistemas lineares em tempo real (COSYMA) e um simulador de sistemas não lineares em tempo real (TRANSIENT) [BUSETTI, 1998].

O principal componente do CAMeL, a linguagem DSL, fornece uma plataforma padronizada de descrição de um modelo além de ser uma base para processamento computadorizado. As características importantes são:

- Combina conceitos de uma abordagem hierárquica através subsistemas com uma representação baseada em engenharia de controle orientada a blocos de descrição de modelos em espaço-de-estado. Em especial ela facilita a conexão de blocos via canais de entrada/saída e o encapsulamento de sistemas de informações dentro dos blocos;
- A descrição de sistemas dinâmicos não lineares é representada simbolicamente, associando a equação nomeada a entrada/saída, estados, parâmetros, equações diferenciais e equações de uma sintaxe precisamente definida. Limitações e

descontinuidades podem ser realizadas através de construções condicionais que ficam distribuídas nos diferentes níveis hierárquicos;

- O modelo descrito é preparado para processamento computadorizado (p. ex. simulação não linear) por um compilador de modelos interno ao programa. São verificadas a semântica e a sintaxe dos modelos formulados, com a emissão de mensagens de erros e com a conversão de modelos dentro de uma estrutura de dados integrados;
- A possibilidade de formulação livre, parametrizável de tipos de modelos, garante bibliotecas de modelos reutilizáveis e sem redundâncias;
- A complexidade e o tamanho do modelo não são afetados por nenhuma restrição no programa.

A linguagem DSL permite a descrição de sistemas dinâmicos através de sistemas de equações diferenciais de 1ª ordem:

$$\overset{o}{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{p}, t)$$
$$\mathbf{y} = g(\mathbf{x}, \mathbf{u}, \mathbf{p}, t)$$

Assim, um sistema pode ser descrito pela indicação de suas entradas $\boldsymbol{u}$, saídas $\boldsymbol{y}$, estados $\boldsymbol{x}$, parâmetros $\boldsymbol{p}$, e equações diferenciais e de saídas.

A linguagem DSL é a linguagem comum entre as ferramentas que constituem o ambiente CAMeL tais como : Pré-processador para geração de modelo, simulação não linear, identificação não linear e linear, otimização, simulação em tempo real e análise linear.

2.5.3.4 IPANEMA

Para o gerenciamento da complexidade dos sistemas mecatrônicos adota-se uma estrutura modular e hierárquica subdividindo o sistema em módulos de funções. As subestruturas criadas podem ter um projeto independente. Neste caso é adequado a

utilização de plataforma para simulação distribuída. O paralelismo é utilizado para proposta de estruturação, além da consideração de tempo real.

O IPANEMA (*Integration Platform for Networked Mechatronic Applications*) é um ambiente para simulação com *hardware-in-the-loop* distribuído e realização de sistemas mecatrônicos. Sua estrutura de *hardware* é baseada em um sistema multiprocessador. Supre duas necessidades : estratégia para organizar processos paralelos em um sistema; uma biblioteca de *software* que possa organizar o processamento de informação em tempo real.

O projeto de *software* é baseado em um modelo multicamadas, dispondo de uma camada de *software* entre o sistema operacional em tempo real e o processamento de informação da aplicação mecatrônica. Este *software* propõe-se a encapsular os serviços do sistema operacional visando a aplicação.

Diferentes processos são empregados para representar o processamento da informação. Estes processos são feitos pelos seguintes elementos ou classes (sob a metodologia de orientação a objetos):

- *Calculadores* : Processamento de informação (avaliação de equação diferencial e processamento de informação discreta).
- *Adaptadores* : Conexão dos elementos periféricos em uma simulação com HIL sob condições rígidas de tempo real.
- *Assistente* : Implementação de tarefas administrativas, por exemplo, início e parada de simulação e mudança de parâmetros do sistema mecatrônico.
- *Moderador* : Gerenciamento de todos os assistentes. Permite transparência da rede para o usuário.

2.5.3.5 Análise

Toda a plataforma descrita (MOOMo-CAMeL-IPANEMA) fornece um poderoso suporte ao ciclo de desenvolvimento de sistemas mecatrônicos. Como tais sistemas normalmente se apresentam sob alta complexidade exige-se uma ferramenta baseada em orientação a objeto que seja capaz de uma descrição física compreensível para todo o

sistema. Isto facilita a obtenção de um modelo matemático granular (subsistemas) onde qualquer parâmetro do sistema pode ser alterado sem dificuldades. As ferramentas de análise e otimização de controladores são encontradas no CAMeL que trabalha baseado na linguagem DSL. É um ambiente com os mesmos objetivos encontrados nas ferramentas de otimização e controle de outros pacotes comerciais, mas com uma outra forma de descrição do modelo. A possibilidade de simulação com HIL traz a necessidade de ambiente em tempo real adaptada a estrutura modular do sistema mecatrônico sob desenvolvimento. O IPANEMA é um ambiente para organização de processos paralelos com informação distribuída em tempo real baseado em *hardware* de multiprocessadores.

Em suma, é um ambiente sofisticado e dedicado a sistemas mecatrônicos, não sendo adequado a tratamento de outras propostas como, por exemplo, indústria de processos. Além disto, sua complexidade impede sua utilização no ensino de controle sendo propício para pesquisa avançada. Entretanto, os conceitos, estrutura de *hardware* e *software* e subdivisão do ambiente forneceram subsídios importantes na concepção e implementação do VIEnCoD.

2.5.4 Scilab

O Scilab é um pacote de *software* científico para computação numérica em ambiente amigável desenvolvido para aplicações de sistemas de controle e processamento de sinal. É constituído de três partes distintas : interpretador, biblioteca de funções (*procedures*) e bibliotecas de rotinas C e Fortran. A ferramenta chave na sintaxe do ambiente é a habilidade no suporte a matrizes. Demais ferramentas :

- Estruturas de dados : polinômios, matrizes de *strings*, listas, sistemas lineares multivariáveis,... etc;
- Interpretador e linguagem de programação avançados (análogo ao Matlab);
- Suporte de funções matemáticas;
- Ambiente aberto : fácil interfaceamento com Fortran e C via *online dynamic link*), e possibilidade de criação e adição de funções e bibliotecas ;
- Facilidades no tratamento e análise de sistemas não lineares, otimização linear e não linear e otimização quadrática;
- Capacidade de estrutura simbólica através de interface com Maple.

Algumas bibliotecas embutidas são de: Álgebra Linear, Controle (clássico, robusto,... etc), Otimização LMI, processamento de sinais, simulação, otimização, modelagem e simulação de sistemas híbridos.

A filosofia geral do Scilab é prover um ambiente com as seguintes características :

- Dispor de tipos de dados variados e flexíveis na sintaxe;
- Prover uma quantidade razoável de primitivas;
- Ter um ambiente aberto de programação onde novas primitivas são facilmente adicionadas;
- Dar suporte ao desenvolvimento de "*toolbox*".

O ambiente apresenta uma estrutura operacional e de recursos semelhante com o Matlab. Todos os recursos disponíveis neste ambiente são contemplados no Matlab, geralmente de forma mais desenvolvida. Naturalmente, pelo mesmo estar ainda em versões iniciais e com pouca penetração nos meios acadêmicos e industrial o leque de aplicativos em áreas específicas é inferior comparativamente ao Matlab. Em suma, é um pacote que se assemelha ao Matlab em suas versões anteriores mas com distribuição gratuita. Informações mais apuradas bem como o acesso ao *software* podem ser obtidos em seu site[7].

2.6 Comentários

Neste capítulo identificaram-se diversos elementos motivacionais para a concepção de um ambiente CACSD multifuncional que atenda as necessidades de ensino, pesquisa e serviços na área de controle. Tais elementos são advindos da avaliação das características de diversos ambientes com proposta CACSD existentes no tocante a sua funcionalidade didática (ensino), operacional e técnica (interface com elementos físicos, ambiente amigável; serviços) e científica (abrangência nos conteúdos de controle - clássico e avançado).

---

[7] *Scilab Group*, http://www.rocq.inria.fr, Out. 1999.

A utilização de ambientes com proposta CACSD no ensino de controle em auxílio aos métodos clássicos tem sido bastante acentuada, conforme observado anteriormente na análise da atual da produção científica. Esta prática tem se mostrado eficaz para treinar os estudantes a integrar os conhecimentos e observar os relacionamentos causa-e-efeito através das simulações. Uma das grandes vantagens do emprego da CACSD no ensino de controle é a facilidade que advém de se dispor de um ambiente integrado e padronizado para efetuar as diversas etapas de um projeto de controlador. A preocupação em desenvolver estes ambientes de forma a adequar a esta função (ambientes amigáveis com conteúdo educacional) tem despertado interesse de pesquisadores na área de controle.

Alguns pacotes computacionais (os mais difundidos) foram analisados de forma a verificar as características operacionais e recursos que pudessem torná-los mais atrativos que a plataforma Matlab – Simulink, utilizada no presente projeto. A integração deste ambientes com elementos de *hardware* (sistemas de aquisição, processadores) com preocupação de tempo real é a forte tendência no desenvolvimento dos ambientes CACSD. São as chamadas "soluções integradas" que fazem surgir novos conceitos e metodologias aplicadas ao desenvolvimento de sistemas de controle, como simulação com HIL. A plataforma Matlab, por exemplo, apresenta um elevado número de soluções integradas voltadas a assuntos específicos. Outro ponto observado é quanto as soluções de *hardware* dos sistemas - normalmente de forma proprietária, complexa e de alto custo.

Este estudo e avaliação conjunta - necessidades e ferramentas CACSD - juntamente com a revisão da produção científica atual, forneceram subsídios suficientes que delinearam a concepção e implementação do VIEnCoD, objeto do presente trabalho. Metodologias e conceitos foram desenvolvidos; os existentes foram adotados e adaptados à proposta do ambiente. Este esforço é colocado nos capítulos seguintes.

Capítulo 3

3 Ciclo de Desenvolvimento de Sistemas de Controle

3.1 Metodologia de Desenvolvimento

Muitas vezes as soluções clássicas de controle não se aplicam a sistemas ou processos com características específicas de desempenho e requerimentos de funcionamento. Os sistemas *Mecatrônicos* e muitos processos industriais enquadram-se neste perfil. Desta forma, estratégias de controle complexas sob uma metodologia de suporte ao processo de desenvolvimento de sistemas de controle são sugeridos [AMARAL et al., 1992].

A metodologia se baseia em dar suporte ao conceito proposto neste trabalho - ***Ciclo de Desenvolvimento de Sistemas de Controle*** **(CDSC)** - que engloba tarefas típicas desde a modelagem e identificação da planta ou processo à síntese do controlador. Neste ciclo os elementos físicos participam do processo de análise e simulação. A síntese do controlador e o conceito da simulação com *Hardware-in-the-Loop* são tarefas que envolvem estes elementos (planta e controlador) e são exploradas com mais detalhes neste capítulo. Assim, a aplicação em problemas industriais reais é otimizada de forma a aproximar a solução de controle à real necessidade da planta sem transtorno no processo de inserção do controle na malha.

A estrutura do CDSC pode ser observada na figura 9 e constitui-se das seguintes etapas:

- ***Identificação*** : Esta etapa corresponde a modelagem do sistema físico (planta[8]) através do conhecimento e desenvolvimento de um modelo físico, geração de um modelo matemático para uma descrição compreensível em um ambiente

[8] Denota-se "planta" à uma máquina, processo ou sistema, de forma mais abrangente.

computacional e subseqüente solução numérica deste modelo através de simulações para obtenção de parâmetros e índices de desempenho (propriedades) do sistema. Em muitos casos a tarefa de obtenção do modelo através de leis físicas não é trivial. Neste caso, torna-se necessário a determinação das grandezas físicas não observáveis a partir de medidas realizadas. Tal proposta, denominada de *Identificação* , visa a modelagem matemática de um sistema físico a partir de informações observadas (medidas) deste sistema. Maiores detalhes sobre este processo são vistos mais adiante.

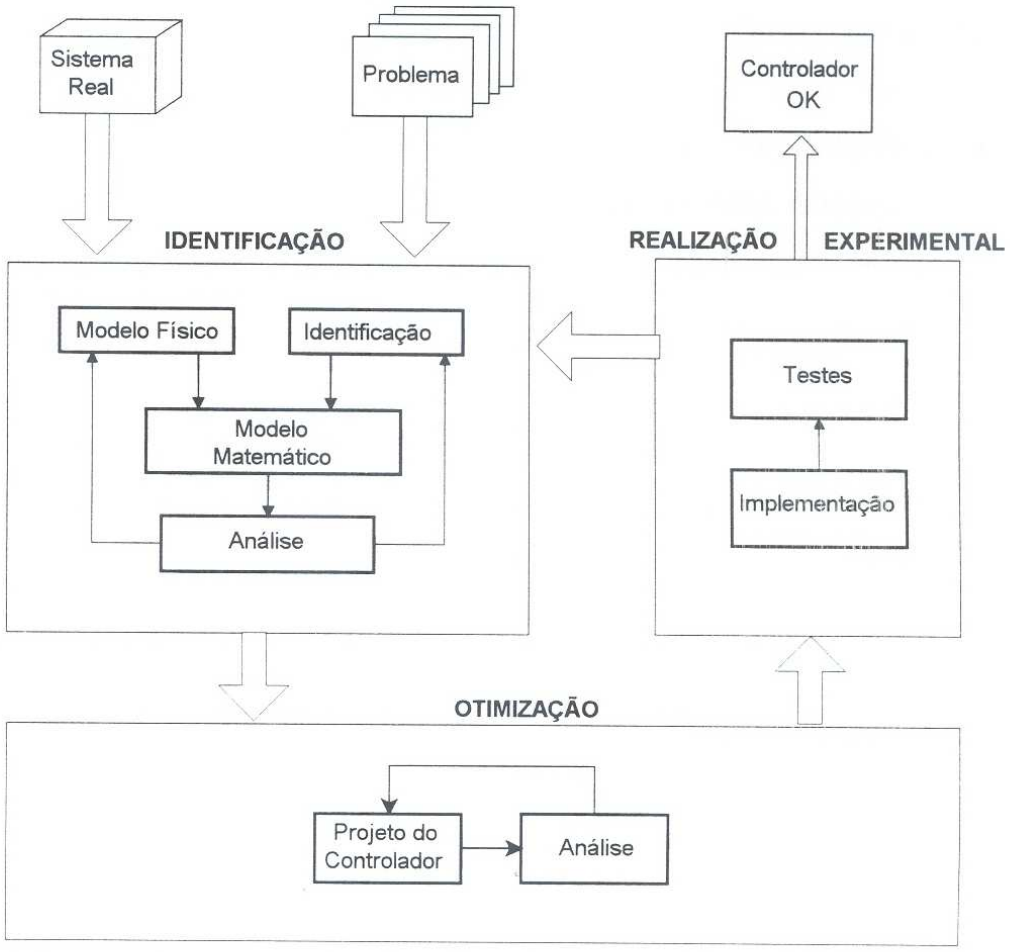


Figura 9 – Ciclo de Desenvolvimento de Sistemas de Controle - CDSC

- ***Otimização*** : Esta etapa é referente ao projeto do controlador baseada em uma estratégia de controle. São escolhidas técnicas para a síntese do controlador em função das características do modelo obtido e da filosofia que será empregada no projeto, por exemplo, o modelo matemático ou representação computacional obtida na fase anterior é inserido juntamente com um controlador PID em um

sistema realimentado (malha de controle), onde seus parâmetros (P,I e D) são obtidos por algoritmos de otimização em função de restrições impostas à resposta desejada do sistema no tempo a sinais de excitação. A malha de controle a ser sintonizada é configurada podendo apresentar elementos não lineares. Simulações são realizadas no domínio do tempo e da freqüência de maneira a validar os critérios de desempenho estabelecidos.

- ***Realização Experimental*** : Tem-se nesta etapa a implementação do sistema de controle. Primeiramente o controlador é realizado em um sistema computacional (algoritmo de controle) e conectado à malha de controle com o sistema real (planta). Neste ponto iniciam-se as simulações com *hardware-in-the-loop*, onde gradativamente são incorporados componentes reais na malha de controle. Tal conceito é mais presente em sistemas mecatrônicos onde, devido a complexidade da planta, a mesma é dividida em subsistemas que são representados computacionalmente, participando das simulações como elementos matemáticos. No decorrer do projeto estes elementos são substituídos por elementos da planta real. Este processo é representado na figura 10

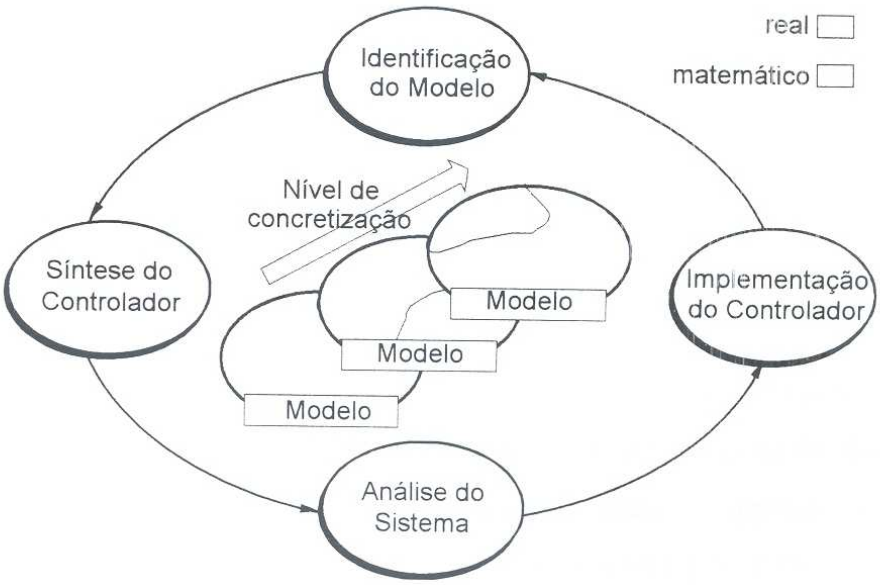


Figura 10 – Ciclo de desenvolvimento para simulação com *hardware-in-the-loop*

Em todas as etapas existem critérios de validação e testes que determinam a aceitabilidade do elemento sob modelagem e projeto. Assim, em qualquer parte do projeto onde houver um resultado indesejável, deve-se retornar à tarefa anterior de forma a redefinir nova estrutura, parâmetros ou estratégias que conduzam a resultados satisfatórios.

3.2 Etapas do Ciclo de Desenvolvimento

Os itens subseqüentes apresentam o desenvolvimento da teoria de controle relativa a cada etapa do CDSC implementadas e disponíveis no VIEnCoD.

A finalidade é a de colocar em evidência o conteúdo teórico explorado no ambiente, dando ao usuário (aluno, pesquisador ou engenheiro de aplicação) noção das possibilidades e fronteiras no cumprimento de determinada tarefa (identificação, projeto, controle,... etc). Não cabe aqui aprofundamentos e tratados sobre algoritmos, conceitos e teoria de controle avançados considerados, por exemplo, nas fases de Identificação e Otimização. Da mesma forma, os conteúdos de controle clássico serão superficialmente desenvolvidos, dado que se supõe que um usuário de ambiente CACSD, tenha algum conhecimento a respeito. Em acréscimo, o ambiente possui em suas interfaces menus intuitivos amparados por recursos de orientação e menus de ajuda específicos a cada operação, como será visto nos capítulos posteriores.

3.2.1 Modelagem e representação de sistemas

A representação de sistemas de controle podem ser feitas de diferentes formas que diferem na maneira em que os modelos são representados e em seu grau de abstração. Na figura 11 são encontradas três formas de representação : representação física, representação em controle e representação experimental [ZANELLA, 1996].

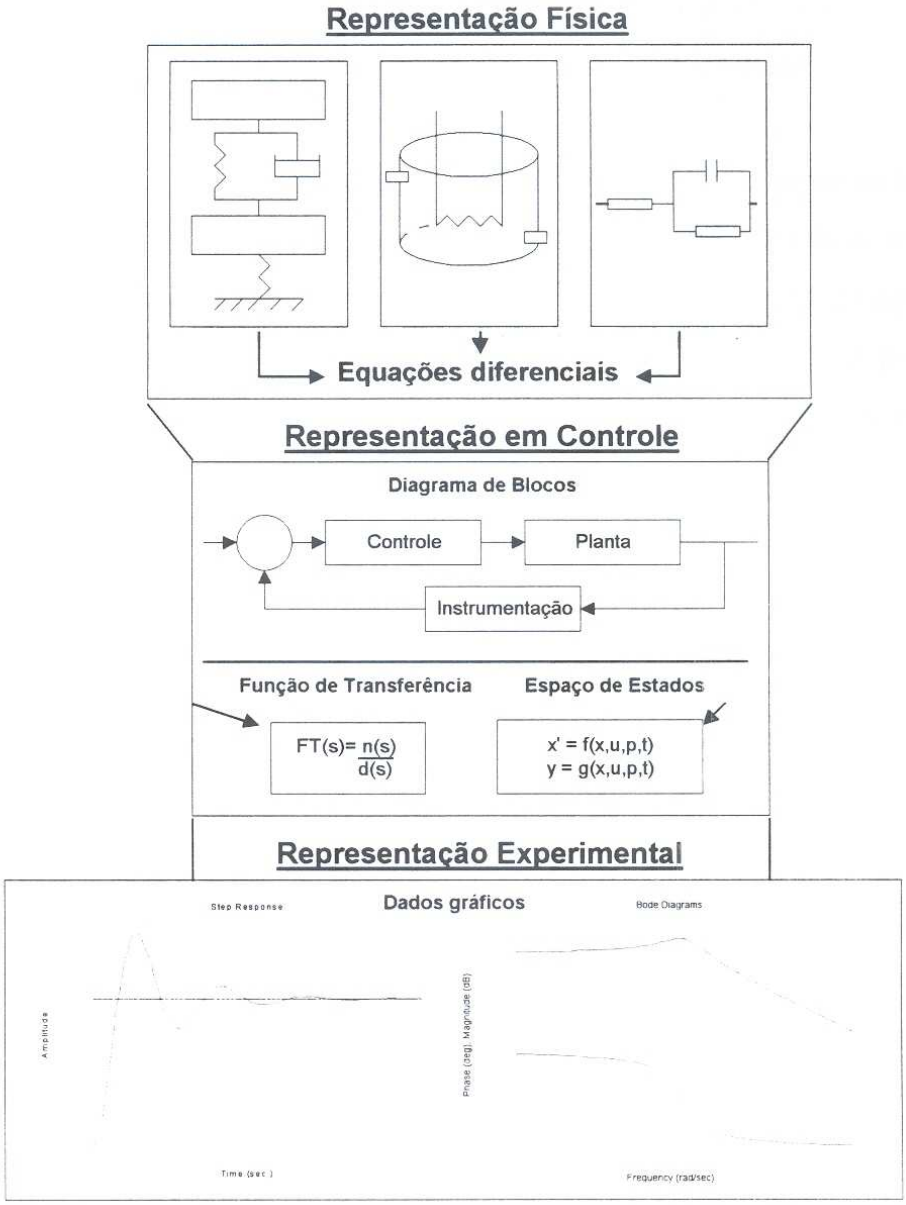


Figura 11 – Formas de representação de sistemas de controle

3.2.1.1 Representação Física e em Controle

Partindo de um sistema real, deriva-se um modelo substituto físico através de equações diferenciais. É obtida a denominada ***representação física*** do sistema. A dinâmica de muitos sistemas (mecânicos, elétricos, térmicos,... etc) são descritos através de equações diferenciais. A modelagem matemática de um sistema dinâmico é definida como um conjunto de equações que representam a dinâmica do sistema e que derivam das equações diferenciais. Os modelos dinâmicos podem ser classificados em:

contínuos/discretos, parâmetros distribuídos/concentrados, monovariáveis/multivariáveis, invariantes/variantes, lineares/não lineares, determinísticos/estocásticos.

As ***representações em controle*** baseados nos modelos matemáticos podem assumir diferentes formas, tais como : representação no espaço de estados, função de transferência e diagrama de blocos. Dependendo do sistema, uma forma de representação pode ser mais adequada do que outra. Sistemas não lineares, variantes no tempo, MIMO, problemas de controle ótimo, são tratados mais adequadamente por espaço de estados. De outra forma, para análise de resposta no tempo e freqüência a representação por função de transferência pode ser mais adequada. Quando a complexidade do sistema aumenta é conveniente a subdivisão do mesmo em subsistemas que representam e fornecem as várias relações de entrada e saída. Para tal é conveniente a representação por diagrama de blocos onde as inter-relações entre os componentes são visualizadas pelo fluxo de sinais presentes no sistema. A seguir, estas formas de representação em controle são abordadas.

♦ ***Função de Transferência***

A função de transferência de um sistema de equações diferenciais lineares invariantes no tempo é definida como a relação da transformada de Laplace da saída (função resposta) para a transformada de Laplace da entrada (função excitação) sob a hipótese de que todas as condições iniciais são nulas [OGATA, 1995]. Sua aplicabilidade, portanto, é limitada a representação de sistemas lineares invariantes no tempo.

Por exemplo, dado o sistema linear invariante no tempo definido pela equação diferencial :

$$a_0 \overset{n}{y} + a_1 \overset{(n-1)}{y} + \ldots + a_{n-1}\, y + a_n\, y \;=\; b_0 \overset{m}{x} + b_1 \overset{(m-1)}{x} + \ldots + b_{m-1}\, x + b_m\, x \qquad (n \geq m)$$

onde y é a saída do sistema e x é a entrada, tomando-se a transformada de Laplace de ambos os membros da equação tem-se a seguinte função de transferência :

$$G(s) = \frac{L[saída]}{L[entrada]}\Big|_{condições\ iniciais\ nulas} = \frac{Y(s)}{X(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \ldots + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n}$$

Vale também alguns detalhes relativos a esta forma de representação :

- Considera os elementos necessários para relacionar a entrada à saída, mas não fornece qualquer informação concernente a estrutura física do sistema. Muitos sistemas podem ser representados pela mesma função de transferência;
- Fornece uma descrição completa da dinâmica do sistema;
- Pode ser obtida experimentalmente através de dados (sinal de saída) obtido da excitação do sistema a sinais de excitação típicos (representação experimental).

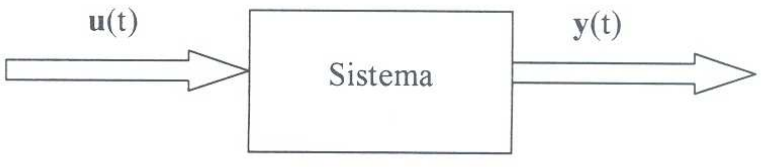
♦ ***Espaço de estados***

A tendência atual em sistemas de engenharia é para maior complexidade, devido principalmente às necessidades de tarefas mais complexas e de boa precisão. Estes sistemas são geralmente variantes no tempo, não lineares ou MIMO. Para tal, torna-se conveniente a sua descrição por espaço de estados.

Pela sua estrutura matricial e abordagem no domínio do tempo os métodos de espaço de estados são mais adequados para tratamentos computacionais. O aumento no número de variáveis de estado, variáveis de entrada ou saída não aumentam a complexidade das equações.

As técnicas convencionais (lugar das raízes e resposta em freqüência) não se aplicam ao projeto de sistemas de controle ótimos e adaptativos que são principalmente não lineares e variantes no tempo. O projeto de sistemas de controle via método de espaço de estados permite o projeto destes sistemas, tendo pólos de malha fechada especificados ou sistema de controle ótimos com respeito a índice de desempenhos desejados. Isto permite uma descrição matemática precisa da dinâmica do sistema.

O espaço de estado é representado pela estrutura abaixo, onde os sinais $\mathbf{u}(t)$ e $\mathbf{y}(t)$ são grandezas vetoriais :



Para tal temos as equações do espaço de estados:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \qquad \text{equação de estado}$$

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}, \mathbf{u}, t) \qquad \text{equação de saída}$$

Se estas equações são linearizadas sobre o estado de operação e se as funções vetoriais não envolvem *t* explicitamente (sistema invariante no tempo), tem-se :

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$

Onde :

$\mathbf{x}(t)$ = vetor de estado (nx1 vetor de estados de um sistema de ordem n)

$\mathbf{A}$ = n x n matriz de sistema

$\mathbf{B}$ = n x r matriz de entrada

$\mathbf{u}(t)$ = r x 1 vetores de funções de entrada do sistema

$\mathbf{y}(t)$ = p x 1 vetores de saídas definidas

$\mathbf{C}$ = p x n matriz de saída

$\mathbf{D}$ = p x r matriz que representa o acoplamento direto entre a entrada e saída

Neste sistema a saída $\mathbf{y}(t)$ para $t \geq t_1$ depende do valor $\mathbf{y}(t_1)$ e da entrada $\mathbf{u}(t)$ para $t \geq t_1$. O sistema dinâmico deve envolver elementos que memorizem estados anteriores sugerindo a participação de integradores cujas saídas podem ser consideradas como variáveis que definem o estado interno do sistema dinâmico. Assim, as saídas de integradores servem como variáveis de estado. O número de variáveis de estado para definir completamente a dinâmica do sistema é igual ao número de integradores envolvidos no sistema.

♦ ***Modelos paramétricos***

Os modelos paramétricos referem-se aqui a uma representação polinomial das funções de transferência, colocado sob forma de equação a diferenças linear. A utilização

destes modelos será bastante intensa na fase de *identificação* , sendo desta forma, abordado de forma mais detalhada neste item.

♦ ***Diagrama de blocos***

O diagrama de blocos permite uma maior granularidade na descrição do sistema quando é possível a subdivisão do mesmo em subsistemas. Assim, componentes do sistema podem possuir sua dinâmica (descrita através de equações diferencias ou função de transferência ou espaço de estados) encapsulados em blocos que explicitam as relações de entrada e saída. Desta forma, consegue-se a descrição de um sistema complexo (p. ex. sistema mecatrônico) de forma modular e hierárquica, permitindo com isto, maior nível de detalhe a mesma [ZANELLA, 1996].

Nos diagramas de blocos, o destaque está no fluxo de sinais do sistema composto pela interligação funcional dos blocos individuais ou subsistemas. Conforme será abordado posteriormente neste trabalho, esta forma de representação permitirá um ambiente de programação orientado a objetos adicionando facilidades na estrutura de bibliotecas de elementos. Uma referência sobre esta abordagem é visto em [DOYLE et al., 1996].

3.2.1.2 Representação Experimental

Na forma de ***representação experimental*** as propriedades do sistema são descritos por gráficos de onde se extraem seus parâmetros. Os dados são obtidos da resposta do sistema a sinais de excitação e com isto, o comportamento dinâmico do sistema é observado e um modelo matemático pode ser derivado. As ferramentas clássicas para análise no tempo e freqüência auxiliam este processo. São elas :

- ***Resposta no tempo*** : Na análise e especificação de critérios de desempenho de um sistema de controle muitas vezes se faz através de parâmetros no tempo tais como tempo de subida, tempo de acomodação, tempo de atraso, instante de pico, sobressinal máximo. Para tal, é necessário a aplicação de sinais de teste típicos à planta de maneira que a resposta do sistema possa ser analisada matematicamente. A escolha do sinal depende das características normais de

operação da planta, devendo este se aproximar desta condição. Os sinais de excitação mais comuns são : impulso, degrau, rampa e parábola. Na aplicação destes sinais à planta, obtemos o comportamento transitório e de regime permanente do sistema. Este último nos fornece o erro estacionário. Quando é possível extrair todos estes parâmetros da resposta do sistema pode-se obter sua função de transferência. Por exemplo, se um sistema de 1ª ordem que sofre excitação de um degrau e sua reposta leva dois segundos para atingir 63% de seu valor de regime (que é o dobro da amplitude do degrau aplicado). Sua função de transferência é a seguinte:

$$FT(s) = \frac{2}{2s+1}$$

Entretanto, a resposta experimental muitas vezes não fornece todas as informações suficientes para a determinação de sua dinâmica, decorrente de sinais de teste típicos. Neste caso, utiliza-se sinais específicos para aplicação da técnica de Identificação. A resposta do sistema à aplicação destes sinais (ruído branco, PRBS) é analisada por algoritmos de estimação onde obtém-se modelos paramétricos da planta.

- ***Resposta em freqüência*** : A resposta em freqüência é a resposta de um sistema em regime permanente com sinal de entrada senoidal. A relação entre a amplitude e defasamento do sinal de entrada e o sinal de saída para um determinado espectro de freqüência fornece a resposta em freqüência do sistema podendo ser visualizado através do gráfico de Bode (resposta experimental). Esta técnica estende o alcance da resposta ao tempo para plantas com características específicas onde a análise no tempo é dificultada tais como : sistemas com atraso de transporte de sinais, plantas com incertezas ou deficientemente conhecidas, sistemas de controle não lineares.

3.2.1.3 Comentários

Das representações experimentais abordadas, ambas fornecem aproximações do comportamento real. Algumas limitações que os modelos fornecem na descrição dos sistemas físicos são as seguintes:

- *Imprecisão nos parâmetros* : As repostas experimentais não fornecem com exatidão os parâmetros reais da planta. Isto se deve a alguns fatores tais como : as condições de teste (excitação através de sinais típicos) não reproduzem as condições normais de operação; a própria aquisição de dados e os parâmetros podem variar com o tempo.
- *Não linearidades* : A grande parte das técnicas de análise e projeto de sistemas de controle são aplicadas a sistemas lineares (onde o princípio da superposição pode ser aplicado). Na prática, muitos elementos físicos (sistemas eletromecânicos, hidráulicos, pneumáticos,... etc) envolvem relações não lineares entre as variáveis. Entretanto, muitas vezes é possível se identificar faixas de operação do sistema onde predomina um comportamento linear, permitindo, dentro desta região, a aplicação das técnicas de controle convencionais. Alguns tipos de não linearidades encontradas : saturação, zona morta, histerese, folga (*backlash*), atritos (estático, de Coulomb e outros), atraso de transporte, liga-desliga e outros.

3.2.2 Identificação de sistemas

Em muitos casos a tarefa de obtenção de um modelo através da utilização de leis físicas não é trivial e a representação matemática associada pode envolver complexos métodos numéricos. Em acréscimo, estratégias de controle adaptativo requerem a contínua atualização deste modelo para operação on-line.

Neste contexto, torna-se necessário a determinação das grandezas físicas que não observáveis diretamente a partir de medidas realizadas. Tal procedimento, denominado de *Identificação*, visa a modelagem matemática de um sistema físico a partir de informações observadas (medidas) deste sistema. Um sistema dinâmico pode ser conceitualmente descrito como na figura 12.

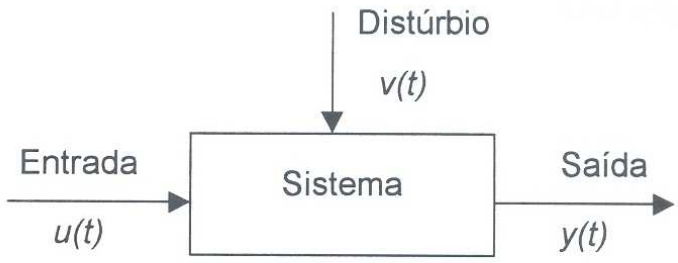


Figura 12 – Descrição de um sistema dinâmico

O problema de identificação de sistemas [ARRUDA, 1992], ilustrado na Figura 13, representa um conjunto de tarefas encadeadas de maneira a conduzir o processo de identificação de forma mais detalhada e controlada, permitindo a rápida identificação de problemas através de resultados parciais obtidos em cada etapa proposta. Para tal, é necessário um adequado suporte a nível de *hardware* (instrumentação), suporte gráfico e suporte a nível de algoritmos e recursos voltados à Identificação; todos contemplados pelo VIEnCoD, conforme será visto nos capítulos seguintes.

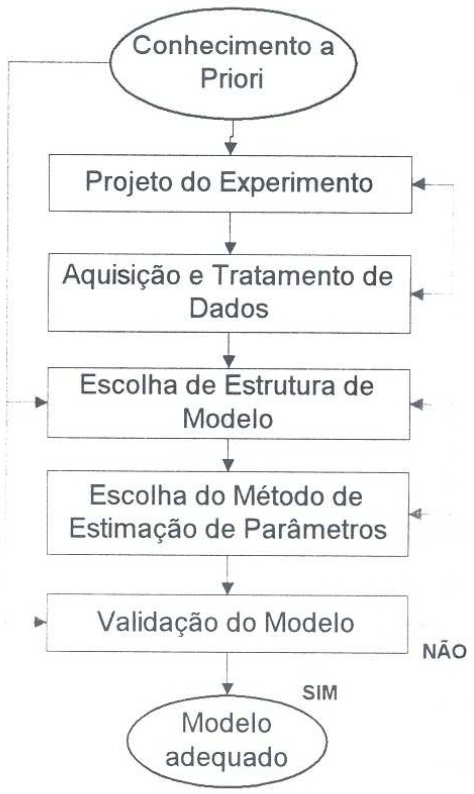


Figura 13. Processo de Identificação

Cada etapa do processo de identificação proposto é descrito a seguir.

3.2.2.1 Conhecimento a priori

São informações sobre a planta (experimentos anteriores, leis físicas que regem o sistema,... etc) que fornecem subsídio para escolha do modelo de estrutura mais adequada. Quando não se dispõe de nenhum conhecimento sobre a planta, técnicas especiais (p. ex. resposta ao impulso) são aplicadas fornecendo alguma informação.

3.2.2.2 Projeto do experimento

O projeto do experimento se traduz no planejamento e gerenciamento operacional da experiência na fase inicial onde alguns pontos devem ser atendidos, tais como : escolha e implantação de sistema de aquisição adequado, definição das características de aquisição (tempos de aquisição, **tipos de sinais**, características dos sinais, **intervalo de amostragem**, amplitude, tratamento, filtragem,... etc) e quantidade de realizações. Para tal, é necessário, baseado no conhecimento a priori da planta, verificar suas características físicas e operacionais.

O **sinal de excitação** utilizado em um experimento de identificação tem uma influência significativa no processo de estimação de parâmetros. A análise aprofundada desta influência é tratada em [SÖDERSTRÖM et al., 1989]. Os seus valores (parâmetros) definem a aproximação do modelo obtido do modelo físico real. Uma **condição importante** é a de que um sinal de ***excitação persistente*** seja utilizado na excitação do sistema. Isto implica em que todos os modos do sistema são excitados durante o experimento de identificação. Em outras palavras, o espectro do sinal de entrada deve satisfazer certas propriedades de forma a garantir que o sistema seja identificado. Ilustrando, o método dos mínimos quadrados falha na estimação de parâmetros quando o sinal aplicado é um impulso – o sinal não é persistente o suficiente. Um sinal é de excitação persistente de ordem n se sua matriz de covariância de ordem n é positivo definido; no domínio da freqüência esta condição é equivalente a dizer que a densidade espectral do sinal é não zero em pelo menos n pontos –para um sinal que tenha densidade

espectral estritamente positiva para todas as freqüências garantirá uma excitação persistente de ordem suficiente.

Em algumas situações a escolha do sinal é em função do método de identificação empregado. Por exemplo, a *análise transiente* requer um degrau ou impulso como entrada, enquanto a análise de correlação geralmente requer um sinal PRBS (*Pseudorandom Binary Sequence*). Em outras situações, entretanto, o sinal é escolhido de muitas maneiras diferentes tornando-se um aspecto importante no projeto de identificação de sistemas.

Uma regra geral que pode ser formulada, segundo [SÖDERSTRÖM et al., 1989] é : "o sinal de entrada dever possuir maior energia na banda de freqüência de interesse da aplicação pretendida para o modelo".

Cuidados devem ser tomados também quanto a amplitude dos sinais aplicados. A estimação dos parâmetros do modelo pode ser sem valia se a amplitude do sinal aplicado atinge a faixa de operação não linear da planta. Isto se aplica também para sinais de amplitude muito pequena onde podem ocorrer não linearidades do tipo zona morta. Em acréscimo, a precisão na estimação aumenta com o aumento da amplitude do sinal.

Com relação a taxa de amostragem adotada deve-se pressupor o seguinte : se a taxa é pequena pode-se omitir informações da dinâmica em alta freqüência; se for elevada os distúrbios podem ter influência elevada, além de que obtém-se pouca informação da dinâmica de baixa freqüência do sistema. Uma regra prática aproximada sugerida em [SÖDERSTRÖM et al., 1989] é a de que o intervalo de amostragem seja de 10% do tempo de acomodação da resposta ao degrau de um sistema.

Os sinais mais comuns utilizados para o propósito em questão são :

- ***Senóide*** : utilizado para o método de levantamento do modelo da planta através da sua reposta em freqüência. Um sinal de excitação senoidal de freqüência variável (valores dentro da faixa de freqüência de interesse) é comparado com a saída do sistema em termos da amplitude e defasamento. Isto resulta nos gráficos de Bode onde é comparado com padrões pré definidos através de aproximações

assintóticas. Existem muitos complicadores à aplicação deste método : imprecisões causadas pela instrumentação (resposta em freqüência deve ser plana, possibilidade de erros de medição), condições de não linearidades da planta (se sinal de excitação for de amplitude muito pequena haverá erros devido à zona morta; se muito grande haverá problema de saturação). Portanto é um sinal que resulta na obtenção de um modelo grosseiro do sistema servindo para um propósito de informação inicial (a priori).

- ***Impulso/Degrau*** : Sinais utilizados nos métodos de estimação não paramétrica onde os coeficientes de resposta ao impulso são obtidos diretamente dos vetores de entrada e resposta através da análise de correlação. Em geral o modelo obtido não é preciso, havendo diferença acentuada no ganho estático (na resposta ao degrau) mas fornecendo valor correto da constante de tempo do sistema. Vale ressaltar que a análise do transitório da resposta fornece alguns parâmetros do sistema (constante de tempo, ganho estático, fator de amortecimento,... etc). Entretanto, quando o nível de ruído presente no sistema é elevado fica difícil deduzir sobre as propriedades da dinâmica da planta. Isto justifica a análise de correlação.

- ***Sinais com propósitos específicos à identificação paramétrica*** : São sinais onde são avaliados e determinados os conteúdos espectrais através das funções de covariância e densidade espectral, de forma a se adequarem as necessidades particulares de uma determinada planta no processo de estimação em busca de um modelo com melhor desempenho. Este estudo pode ser encontrado em [SÖDERSTRÖM et al., 1989]. Dentre os sinais desta classe os principais são:
  - *Pseudorandom Binary Sequence* (PRBS) : Este sinal comuta entre dois valores obedecendo certo padrão tal que seus momentos de primeira e segunda ordem (valor médio e função de covariância) são similares ao *ruído branco*.
  - *Sinal Gaussiano Aleatório.*
  - *Sinal Soma de Senóides.*

3.2.2.3 Aquisição, observação e tratamento dos dados

As informações coletadas (dados medidos) que serão utilizados para estimar parâmetros de um modelo que caracteriza o comportamento de um sistema físico são freqüentemente corrompidas pela presença de incertezas (ruídos) que representam algum tipo de perturbação associado ao sistema observado. O conhecimento do tipo de incerteza presente no sistema real é de suma importância na avaliação de como estas incertezas afetam as variáveis estimadas.

Após a realização da aquisição, sob o suporte de condições de instrumentação adequados (imunidade ao ruído, especificações de impedância e resposta em freqüência adequados, resolução de conversores A/D/A necessários,... etc) procede-se a observação dos dados (vetores de excitação e resposta do sistema) plotados graficamente. Porções dos dados observados podem conter mundanças de nível, presença de distúrbios,... etc e que são inadequados para a estimação. O seu conteúdo em freqüência (espectro dos sinais) deve também ser observado com o auxílio de *periodograma.* Este gráfico representa o quadrado da transformada de Fourier. A proposta da análise dos dados desta maneira visa descobrir se existem porções de dados que não são adequados para identificação; se o conteúdo de informação dos dados é interessante na região de freqüência de interesse e para verificar a necessidade de tratamento de sinais antes da aplicação dos algoritmos de estimação.

O pré-processamento pode ser classificado em função do tipo de tratamento e preparação necessários aos dados para a estimação. Algumas formas de tratamento são as seguintes:

- *remoção de valores médios ou tendências lineares* : é sempre recomendável que sempre se remova pelo menos os valores médios atuantes sobre os dados – os resultados na estimação são melhores para a média zero no sinal;
- *seleção dos dados para identificação e validação* : um procedimento comumente utilizado é a separação da porção de dados utilizável em duas partes iguais – uma para o processo de estimação e outra para validação do modelo obtido;
- *remoção de faixas de freqüência indesejáveis ao sistema* : para remoção de ruídos de alta freqüência dos dados e direcionamento para que o modelo se

enquadre em faixas específicas de freqüência – no modelo utilizado para projeto de controle a banda de freqüência deve estar em torno da definida em malha fechada do sistema;

- *Re-amostragem* : feito através de interpolação (levando o sistema para taxa de amostragem maior) ou divisão (levando o sistema para taxas de amostragem menores).

3.2.2.4 Escolha de estrutura de modelo

De acordo com as classes de estruturas de modelo tem-se uma correspondente técnica de estimação adequada. Desta forma, os dados obtidos na aquisição poderão definir parâmetros mais apurados na estimação e o modelo obtido se aproximará mais do sistema real.

Em linhas gerais, a *escolha da estrutura de modelo e a ordem de seus polinômios*, recaem no objetivo da modelagem, ou seja, o conjunto de modelos deve-se adequar a proposta final. Ao final da estimação, o modelo obtido sofre validação onde critérios são aplicados de forma a avaliar a qualidade do modelo obtido. Os resultados da aplicação destes critérios fornecem uma realimentação sobre a escolha da estrutura de modelo. Outros fatores importantes influenciam na seleção da estrutura de modelo:

- *Flexibilidade* : Deve ser possível usar a estrutura de modelo para descrever os diferentes sistemas dinâmicos que são esperados na aplicação. Ambos, o número de parâmetros livres e a maneira que eles entram no modelo são evidentemente importantes.
- *Parsimônia* : A estrutura do modelo deve ser parsimoniosa, ou seja, o modelo deve conter o menor número possível de parâmetros requeridos para representar o sistema real adequadamente. Um modelo sobreparametrizado eleva o esforço computacional; um modelo subparametrizado pode ser impreciso.
- *Complexidade do Algoritmo* : A forma de estrutura selecionada para uma determinada técnica de estimação pode influenciar no esforço computacional requerido.

Os *modelos paramétricos* são representados por estruturas polinomiais que representam as funções de transferência tratadas no modelo. A forma geral de representação desta classe de estrutura de modelo é ilustrada na figura 14.

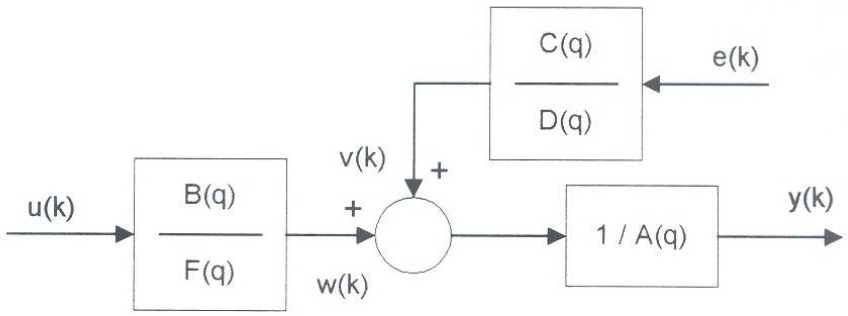


Figura 14. Representação modelos paramétricos

De onde obtém-se a estrutura de modelo paramétrica geral para sistemas SISO estocásticos:

$$y(k).A(z^{-1}) = w(k) + v(k)$$

$$y(k)A(z^{-1}) = \frac{B(z^{-1})}{F(z^{-1})}u(k) + \frac{C(z^{-1})}{D(z^{-1})}.e(k)$$

Onde :

e(k): ruído atuante no sistema - ruído branco - distribuição normal de média zero e variância σ^2

v(k) : término aditivo (soma dos ruídos de medidas e perturbações não mensuráveis)

u(k): sinal externo de controle

y(k): saída do sistema

z^{-1} : operador atraso

$$A(z^{-1}) = 1 + \sum_{i=1}^{na} a_i.z^{-1.i} \qquad C(z^{-1}) = 1 + \sum_{i=1}^{nc} c_i.z^{-1.i}$$

$$B(z^{-1}) = \sum_{i=1}^{nb} b_i.z^{-1.i} \qquad D(z^{-1}) = 1 + \sum_{i=1}^{nd} d_i.z^{-1.i}$$

$$F(z^{-1}) = 1 + \sum_{i=1}^{nf} f_i . z^{-1.i}$$

A escolha apropriada dos polinômios $A(z^{-1})$, $B(z^{-1})$, $C(z^{-1})$, $D(z^{-1})$ permite obter os vários tipos de modelos usados para representação de sistemas dinâmicos estocásticos. Fazendo-se algumas modificações no polinômio $B(z^{-1})$ – a fim de modelar o fato de que um sistema dinâmico tem um atraso "d" (*dead-time*), $z^{-d}B(z^{-1})$, para responder a um sinal de entrada – tem-se alguns tipos de estruturas comuns em sistemas de controle. São elas :

- **ARX (*AutoRegressive with eXternal input*) ou CAR (*AutoRegressive Controlled*)**

No modelo ARX temos : $C(z^{-1}) = 1 \quad e \quad D(z^{-1}) = A(z^{-1})$

Portanto,

$$y(k) + a_1 y(k-1) + \ldots + a_{na} y(k-na) = b_0 u(k-d) + \ldots + b_{nb} u(k-d-nb) + e(k)$$

ou

$$A(z^{-1})\, y(k) \;=\; z^{-d} B(z^{-1})\, u(k) + e(k)$$

E o vetor de parâmetros a serem estimados é:

$$\theta \;=\; \lfloor a_1\, a_2 \ldots a_{na} \quad b_0\, b_1 \ldots b_{nb} \quad , \quad d \rfloor$$

Este modelo se caracteriza pela influência no sinal de saída dos valores passados (regressivos) do próprio sinal de saída (auto). A necessidade de modelo do ruído é influenciada pela a relação S/R (sinal/ruído) do sistema. Quando esta relação é boa pode-se adotar uma estrutura ARX; do contrário sugere-se a adoção da estrutura ARMAX onde o mesmo é modelado pelo polinômio $C(z^{-1})$. De forma geral o modelo do ruído melhora a estimação do modelo.

- **ARMAX (*AutoRegressive – Moving Average with eXternal input*) ou CARMA**

Neste modelo temos : $A(z^{-1}) = D(z^{-1})$

Portanto,

$$A(z^{-1})\, y(k) \;=\; z^{-d} B(z^{-1})\, u(k) + C(z^{-1})\, e(k)$$

E o vetor de parâmetros a serem estimados é:

$$\theta \;=\; \lfloor a_1\, a_2 \ldots a_{na} \quad b_{01}\, b_1 \ldots b_{nb} \quad c_1\, c_2 \ldots c_{nc}, \quad d \rfloor$$

Este modelo possui em adição ao componente auto-regressivo o termo de "média móvel" onde o ruído tem sua participação ponderada em instantes passados na forma de uma "media ajustada".

- **ARIMAX (*AutoRegressive – Moving Average Integrate with eXternal input*) ou CARIMA**

Neste modelo temos : $D(z^{-1}) \;=\; (1 - z^{-1}) A(z^{-1})$

Portanto,

$$A(z^{-1})\, y(k) \;=\; z^{-d} B(z^{-1})\, u(k) + C(z^{-1})\, e(k)$$

A presença do termo integrador modela a influência de perturbações estocásticas de média não nula e /ou lentamente estacionárias, tais como: derivas, *off-sets*, *drifts* e *trends*.

- **OE (*Output-Error*)**

Neste modelo temos :

$$y(k) \;=\; z^{-d}\, \frac{B(z^{-1})}{F(z^{-1})}\, u(k) + e(k)$$

E o vetor de parâmetros a serem estimados é:

$$\theta \;=\; \lfloor b_{01}\, b_1 \ldots b_{nb} \quad f_1\, f_2 \ldots f_{nf}, \quad d \rfloor$$

Estrutura de modelo baseada no erro da saída do sistema.

- **BJ (*Box-Jenkins*)**

Neste modelo temos : $A(z^{-1}) = D(z^{-1})$

Portanto,

$$y(k) = z^{-d}\frac{B(z^{-1})}{F(z^{-1})}u(k) + \frac{C(z^{-1})}{D(z^{-1})}e(k)$$

E o vetor de parâmetros a serem estimados é:

$$\theta = \left[a_1\, a_2 \ldots a_{na} \quad b_{01}\, b_1 \ldots b_{nb} \quad c_1\, c_2 \ldots c_{nc}, \quad d\right]$$

A aproximação de Box e Jenkins consiste em levar em consideração as propriedades do erro de saída, modelado por um processo ARMA.

Os modelos de representação em ***espaço de estados*** para sistemas dinâmicos estocásticos apresenta-se sob a forma :

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + Ke(k) \\ y(k) &= Cx(k) + Du(k) + e(k) \end{aligned}$$

3.2.2.5 Escolha de método de estimação de parâmetros

Na escolha do método de identificação, a proposta de identificação é fundamental pois determina qual o tipo de estrutura de modelo é necessário e qual nível de exatidão é desejada. Se a proposta é o ajuste de um regulador PI, um modelo bastante simples é suficiente, entretanto, quando uma alta precisão é necessária, a proposta é a descrição de processos em detalhe para verificação de outros modelos teóricos. Os métodos de estimação a seguir são ordenados em grau crescente de precisão prevista e incremento da complexidade computacional :

- Análise Transiente;
- Análise de freqüência;
- Método dos mínimos quadrados;
- Método da variável instrumental;
- Método do erro de predição.

Cada um destes métodos apresenta melhor desempenho com determinadas estruturas de modelos conforme será visto mais adiante neste item. A experiência prévia

de outros métodos também é um fator que influi na escolha da estrutura de modelo e método de estimação. Em suma, deve-se manter o equilíbrio entre precisão e complexidade.

Outro ponto a ser observado é quanto a necessidade de identificação *on-line* da planta. Utilizado na sintonia automática de controladores e sistemas adaptativos, este tipo de identificação é necessário quando se deseja a adpatação dos algoritmos de estimação para a forma recursiva, por exemplo, o algoritmo dos mínimos quadrados recursivos.

Conforme citado anteriormente, os dados obtidos na aquisição podem ser corrompidos por perturbações (ruídos) caracterizando incertezas. De um modo geral, estas incertezas presentes nas informações coletadas são tipicamente caracterizadas como perturbações aleatórias aditivas com função densidade de probabilidade conhecida, como por exemplo, um ruído branco gaussiano. Deste modo, os vários métodos utilizados para estimar os parâmetros de modelos de processos incluem considerações estocásticas sobre a natureza das incertezas.

Os métodos de estimação podem ser distinguidos em dois tipos :

1. ***Métodos de estimação não paramétrica*** : Não impõem nenhuma condição de estrutura ao sistema sendo o mesmo linear. Na análise de correlação a seqüência de entrada do sistema é um ruído branco (excitação). Juntamente com a resposta do sistema – o par de vetores [u,y] é utilizado para obter-se funções de covariância e correlação. A função de covariância cruzada do par de vetores fornece a estimação da função de ponderação *h(k)* do modelo utilizado :

$$y(k) = \sum_{n}^{\infty} h(k)u(n-k) + e(k)$$

Na análise espectral o processo é análogo onde tem-se a descrição e tratamento dos dados no domínio da frequência. Através do espectro cruzado (transformada de Fourier da função de covariância cruzada) é possível estimar a função de transferência do modelo linear geral :

$$y(k) = G(k)u(k) + e(k)$$

Os métodos de estimação não paramétrica geralmente fornecem modelos com aproximação limitada fornecendo apenas informações iniciais sobre o sistema.

2. ***Métodos de estimação paramétrica*** : Uma estrutura de modelo específico é assumido e os parâmetros destas estruturas são estimados baseados nos dados. Isto abre uma possibilidade de estruturas de modelos bastante ampla, conforme visto nos modelos paramétricos, possibilitando a descrição do sistema de diversas maneiras. A abordagem na seqüência enfoca os métodos de estimação voltados as estruturas paramétricas vistas.

De acordo com os modelos e os critérios utilizados na escolha do melhor cálculo aproximado (estimador) dos parâmetros reais, tem-se vários tipos de estimadores descritos na literatura. A seguir faz-se referência a cada um dos métodos de estimação relacionados ao tipo de estrutura de modelo mais adequada para ser utilizado. Desenvolvimentos teóricos detalhados sobre estimação podem ser encontrados em bibliografia específica referenciadas no final deste trabalho [LJUNG, 1995] [SÖDERSTRÖM et al., 1989] [BOX et al., 1994] [CHAFOUK, 1997]. O enfoque colocado aqui é o desenvolvimento do conhecimento necessário à utilização das ferramentas de identificação bem como a interpretação dos resultados.

Um método comum e geral na estimação de parâmetros é o método de predição de erro (PE – *Prediction Error*) e *Máxima Verossimilhança* onde os parâmetros do modelo são escolhidos de forma que a diferença entre a saída do modelo (estimado) e a saída real medida é minimizado. Este método é avaliado para todas as estruturas de modelo abordadas. Para as estruturas ARX e espaço de estado black-box métodos baseados em correlação também são avaliados : *Variável Instrumental* (IV) e *Subespaço* (N4SID). Resumidamente tem-se para as estruturas vistas seus respectivos métodos de estimação:

- *ARX :*
  - *Método dos mínimos quadrados (LS)* : minimiza a soma dos quadrados da porção da equação a diferença relativa a $y(k).A(z^{-1})$ menos a porção relativa a $u(k)B(z^{-1})$.
  - *Variável Instrumental (IV) four-stage :* determina os coeficiente de $A(z^{-1})$ e $B(z^{-1})$ tal que o erro entre o polinômios de y(k) e u(k) tornam-se sem correlação com certa combinação linear de entradas. No primeiro estágio é utilizado o método dos mínimos quadrados cujo modelo obtido gera instrumentos para os demais estágios de estimação IV.

- *ARMAX :*
  - Um *critério de predição de erro quadrático robustificado* é minimizado usando algoritmo iterativo Gauss-Newton.
- *OE , Box-Jenkins, Estrutura generalizada:*
  - Os mesmos algoritmo de estimação utilizado nos modelos ARMAX com modificações na computação do erro de predição e gradientes.
- *Espaço de estado :*
  - Método baseado em subespaço que não faz uso de pesquisa iterativa. O algoritmo utilizado, *N4SID,* é descrito em [LJUNG, 1995]. Este algoritmo é complementado por uma fase separada dos mínimos quadrados lineares para re-estimar as matrizes B, D e X(0) das equações de espaço de estado.

3.2.2.6 Validação do modelo

A escolha da estrutura de modelo e a validação dos modelos obtidos após aplicação dos métodos de estimação acabam se tornando etapas complementares. Para a determinação de um modelo de estrutura apropriado é recomendado o uso da combinação de testes estatísticos e gráficos de sinais e variáveis relevantes. O modelo é considerado adequado se as resposta do modelo simulado e a do sistema real são suficientemente próximas (*closeness*). Os procedimentos que guiam esta escolha, baseados em [LJUNG, 1995] e [SÖDERSTRÖM et al., 1989], são colocados a seguir.

- **Função de Perda (*Loss function*)** : Definido em termos do erro de predição do modelo (soma normalizada dos erros de predição quadrática) sobre os dados destinados a validação. Desde que é desejável minimizar a função perda, o valor mais adequado para a ordem e atraso (*delay*) é provido por esta medida. Este procedimento é conhecido como *validação cruzada* (dados utilizados na estimação diferente dos dados utilizados para validação).

- ***Critérios*** : De acordo com a teoria de Akaike, em uma coleção de diferentes modelos, escolhe-se o com menor FPE (*Final Prediction Error*) ou AIC (*Akaike's Information Theoretic Criterion*). Ambos são critérios formulados por:

$$FPE = \frac{1+n/N}{1-n/N} * V \qquad AIC \approx \log[(1+2n/N) * V]$$

onde : n = número total de parâmetros estimados

N = comprimento dos dados

V = função perda (*loss function*) da estrutura em questão

Outro critério adotado também utilizado é o MDL (*Rissanen's Minimum Description Length*) que seleciona a menor estrutura que permite a melhor descrição global dos dados observados.

- ***Análise de erro de predição*** : A média e variância do erro de predição são computados utilizando os últimos N valores, e sua evolução durante a estimação de parâmetro é visualizada graficamente. É esperada que a variância reduza gradualmente a medida que a ordem é incrementada e aproxima-se dos valores reais.

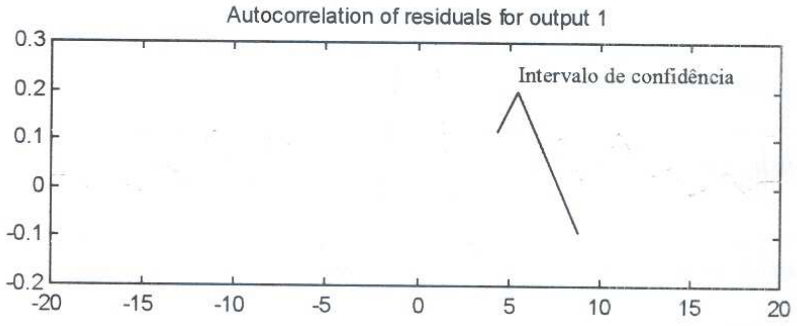


Figura 15. Processo de Identificação - Análise de Autocorrelação

- ***Correlação*** : A autocorrelação normalizada do erro de predição (*residuals*) e a correlação cruzada entre o erro de predição e a entrada de sinal são computadas para os N dados considerados. A análise gráfica destes valores é verificada em torno de um intervalo de confidência específico, conforme observado na figura 15. O aparecimento de picos fora do intervalo de confidência leva a conclusão de que o erro de predição não é branco e consequentemente ainda contém informação que não é representado no modelo, então é possível um incremento na ordem do mesmo. Simplificadamente, existe alguma informação em y(k) que foi originada de u(k-n) que não foi adequadamente descrita no modelo. Para a função de correlação cruzada especificamente, a correlação para *lags* negativos é

a indicação de realimentação da saída na entrada. Isto não significa que o modelo é deficiente; picos de correlação para valores positivos de *lags* significa que valores inadequados de atraso foram especificados.

- ***Pólos e zeros*** : Quando a ordem especificada do modelo é maior do que o valor correto, o modelo está sobreparametrizado, e o diagrama de pólos e zeros de G(z) indicará cancelamento de pares pólo-zero. Devido ao efeito de ruído, este cancelamento não é exato. Outra informação obtida deste diagrama é a escolha inadequada do atraso; a existência de zeros distantes do círculo unitário indica que o atraso especificado foi menor do que o correto valor.

- **Critério MSF (*Mean Square Fit*)** : O recurso de comparação direta da saída do sistema simulado pelo modelo identificado com saída do sistema real (dados experimentais – para validação) pode ser realizado pela raiz média quadrática entre as medidas. Portanto,

$$MSF = \sum_{K=0}^{n}\left[\frac{(y(k)-y_s(k))^2}{N}\right] \qquad y(k)_s : \textit{saída sistema simulado}$$

$$y(k) : \textit{saída sistema real}$$

- **Resposta Transiente** : Um critério de validação baseado na resposta transiente (degrau) pode ser utilizado através da comparação das repostas transientes do modelo paramétrico e do obtido com a técnica de correlação. A concordância das mesmas indica que as ferramentas utilizadas foram adequadas.

3.2.3 Projeto de controladores e otimização

De posse das características dinâmicas da planta obtida com a análise de sua resposta à aplicação de sinais de excitação específicos, parte-se para verificação de seu comportamento em malha fechada (sistema realimentado). O tipo e composição desta malha depende da complexidade do sistema, sendo que a estrutura base comumente utilizada é a estrutura *single loop* ilustrada na figura 16.

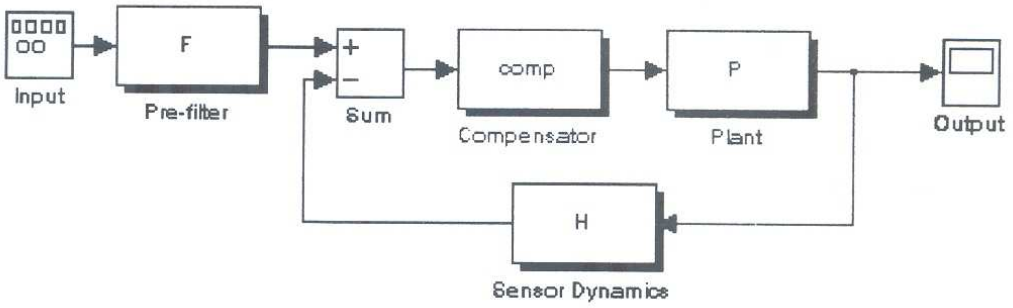


Figura 16 : Sistema de controle *single loop*

Nesta estrutura além da presença do controlador no ramo direto pode ser necessário a presença dos modelos do sensor e pré-filtro. O sensor determina a função de transferência do ramo de realimentação; se suas constantes de tempo são suficientemente pequenas comparadas com as outras constantes de tempo do sistema de controle, sua função de transferência se torna uma constante. Sinais indesejados ou ruídos cujos efeitos se desejam eliminar do sistema podem ser feitos através de pré-filtros. Por ser externo à malha não afeta a estabilidade em malha fechada da mesma. O tipo de filtro depende da faixa de freqüência de atuação do ruído com relação à largura de banda do sistema ou freqüência de interesse na dinâmica da planta, podendo apresentar estruturas diferentes (passa-baixa, passa-faixa,... etc)

A estrutura de controlador depende do tipo de sistema e das necessidades de desempenho para operação. Para sistemas LTI normalmente faz-se uso de estruturas convencionais como : controlador de avanço, controlador de atraso, controlador de avanço-atraso e seu caso particular, os controladores PID. Para tratamento de sistemas não lineares e variantes no tempo recorre-se a outros tipos de estruturas como algoritmos recursivos adaptativos (p. ex. controlador de variância mínima) e estruturas baseadas em conhecimento (controladores FUZZY).

Os controladores PID, segundo já comentado no capítulo 2, representam mais de 95% dos controladores industriais instalados no Brasil, definindo a importância em seu entendimento e configuração. Estes podem apresentar diferentes estruturas (algoritmos) sendo a estrutura mais comum :

$$Gc(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \qquad \text{ou} \qquad Gc(s) = K_p + \frac{K_i}{s} + K_d s$$

Onde : Kp – Ganho proporcional; Ti – Tempo Integral; Td – Tempo derivativo

Sua lei de controle é expressa por :

$$u(t) = K_p\left[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}\right]$$

Como não é possível a realização de um derivador puro, sobretudo quando se trabalha em baixa freqüência se adiciona o termo 1 / (1 + Ns), com N = Td/10 para reduzir seu efeito. Este termo representa um filtro de primeira ordem. A estrutura PID fica:

$$Gc(s) = K_p\left(1 + \frac{1}{T_i s} + \frac{T_d s}{1 + Ns}\right)$$

Cada termo (ação de controle) proporciona características específicas à dinâmica do sistema; o conjunto (controle PID) deve ser projetado para o atendimento pleno das necessidades de desempenho. As ações de controle podem ser resumidas em:

- *Ação proporcional* : agrega rapidez na resposta do sistema; o aumento demasiado do ganho pode levar o sistema a instabilidade;
- *Ação Integral* : proporciona precisão ao sistema diminuindo ou eliminando o erro em regime permanente na resposta do sistema;
- *Ação Derivativa* : ação de controle antecipatória; sua atuação é baseada na verificação da variação do sinal de erro evitando que o seu valor atinja valores excessivos aumento a instabilidade do sistema.

Baseado em critérios de desempenho especificados no tempo (tempo de subida, sobressinal, tempo de acomodação,... etc), na freqüência (margem de fase, margem de ganho, banda de passagem,... etc) ou critérios de otimização (minimização) de uma função erro (p. ex. sistemas de controle ótimo baseado em índices de desempenho quadráticos), define-se a técnica de projeto mais adequada para determinação dos parâmetros do controlador utilizado.

As técnicas convencionais normalmente se limitam a sistemas LTI e SISO, sendo o método do lugar das raízes (domínio do tempo) e o método da resposta em freqüência (análise de Bode e Nyquist) os mais utilizados. O método do lugar das raízes é um método gráfico para determinação das localizações de todos os pólos de malha fechada a partir do conhecimento das localizações dos pólos e zeros de malha aberta a medida que algum parâmetro (normalmente ganho) varia de zero até o infinito. Com ele obtém-se informações sobre da resposta transitória e em freqüência a partir da configuração de pólos e zeros do *plano s*. Os sistemas e componentes cujas características dinâmicas são fornecidas na forma de dados de resposta em freqüência, é mais adequado o projeto por esta abordagem. Também, quando existe a consideração de ruídos no sistema, o desempenho da resposta transitória pode ser especificada em termos da margem de ganho, margem de fase, largura de faixa,... etc.

Os métodos convencionais são baseados em procedimentos de tentativa e erro não produzindo, em geral, sistemas de controle ótimos. Para tal utilizam-se os métodos baseados em espaço de estados, por exemplo, o método de alocação de pólos e o método LGQ (*Liner-quadratic-Gaussian*).

A técnica de Ziegler Nichols é muito utilizada para sintonia de controladores PID baseada nas respostas ao degrau experimentais ou baseadas no valor de ganho proporcional (Kp) que resulta em estabilidade marginal com apenas a sua atuação [OGATA, 1993] [MOLLENKAMP, 1988]. É um técnica conveniente quando a dinâmica da planta não é precisamente conhecida. O método mais conhecido pode ser resumido da seguinte forma:

1. Retirar a atuação dos termos integral e derivativo ($T_i=\infty$ e $T_d=0$) e incrementar o ganho proporcional até o sistema apresentar oscilações mantidas. Este valor será o K_{cr}.
2. Medir o período de oscilação P_{cr}.
3. Os parâmetros PID serão : $K_p = 0.6\ K_{cr}$ $T_i = 0.5\ P_{cr}$ $T_d = 0.125\ P_{cr}$

As plantas com controladores PID determinados por estas regras exibem de 10% à 60% de sobressinal máximo na saída em resposta ao degrau; na média fica na faixa de 25%. Portanto, as regras de Ziegler Nichols fornecem valores iniciais aproximados do

parâmetros PID para um ajuste mais apurado através de outras técnicas, por exemplo, otimização. Em casos onde a planta apresentar integradores o método não se aplica.

3.2.3.1 Controle Digital

Com o desenvolvimento dos sistemas computacionais e de instrumentação eletrônica (p. ex. conversores A/D e D/A) os controladores digitais substituem hoje os controladores analógicos com grande desempenho e adicionando facilidades proporcionadas por recursos de *software*. Portanto, o estudo da teoria de controle voltado a sistemas discretos é a base para a análise e projeto destes controladores. Em um sistema de controle digital o bloco "*compensator*" (controlador) da figura 16 é implementado sob a forma digital conforme ilustra a figura 17. Da mesma forma o filtro pode apresentar-se sob uma estrutura analógica ou implementação computacional (filtro digital). Nesta figura temos os elementos : S/H (Sample / Hold) – amostrador e segurador (mantém o nível do sinal amostrado durante o intervalo de amostragem); ADC (*Analog Digital Converter*) – conversor analógico digital; DAC (*Digital Analog Converter*) – conversor digital analógico; H (Hold) – segurador.

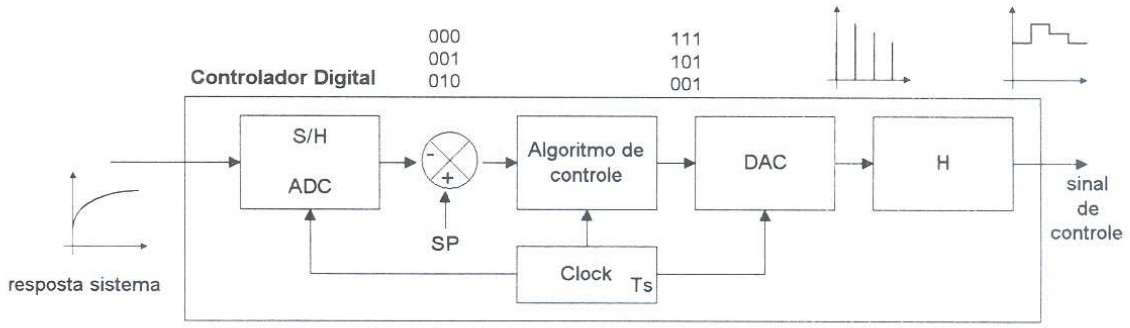


Figura 17 : Estrutura de um controlador digital

Todos os métodos convencionais e de espaço de estados são tratados sob a forma discreta (plano z) analogamente à abordagem no campo contínuo (plano s). No caso dos sistemas discretos existe a preocupação adicional na estabilidade proporcionada por um controlador em função da frequência de amostragem utilizada. Um controlador contínuo pode apresentar um comportamento estável, sendo que seu equivalente discreto pode ser instável. Pelo teorema de Nyquist a frequência de amostragem (fs) de ser $fs \geq 2.f$, onde f

é a máxima frequência do sinal a ser amostrado. Quando o comportamento dinâmico em baixas frequências é importante (p. ex. servossistemas) faz-se fs ≥ fator.BW , onde BW é a largura de banda do sistema e o fator assume valores práticos em torno de 10.

Segundo [BARCZAK, 1995], o projeto de um controlador digital pode ser realizado de várias maneiras sendo que dois métodos merecem destaque :

- *Projeto do controlador no campo contínuo e posterior discretização* : os métodos de projeto comentados anteriormente são aplicados considerando o sistema como contínuo. O controlador contínuo obtido (Gc(s)) é discretizado segundo o método de transformação escolhido : segurador de ordem zero, aproximação triangular, aproximação Bilinear ou método de Tutsin, transformação Bilinear com distorção de freqüência,... etc. Após a discretização é observado a sua estabilidade e comportamento dinâmico, ambos relacionados com o período de amostragem utilizado.
- *Projeto de controlador diretamente no campo discreto* : são utilizados os métodos de projeto discreto, obtendo-se diretamente o controlador.

O controlador PID digital pode ser obtido discretizando a sua equação de controle. O termo integral é obtidos através de aproximação trapezoidal e o termo derivativo pela aproximação da diferença de dois pontos. Temos, portanto :

$$u(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d \frac{de(t)}{dt} \right]$$

$$u(kt) = K\left\{ e(kT) + \frac{T}{T_i}\left[\frac{e(0)+e(t)}{2} + \frac{e(0)+e(2t)}{2} + \ldots + \frac{e\big((k-1)T\big)+e(kT)}{2} \right] + T_d \frac{e(kT)-e\big((k-1)T\big)}{2} \right\}$$

A transformada z da equação acima fica :

$$U(z) = K\left[1 + \frac{T}{2T_i}\cdot\frac{1+z^{-1}}{1-z^{-1}} + \frac{T_d}{T}\left(1-z^{-1}\right) \right].E(z) \quad = \quad \left[K_p + \frac{K_i}{1-z^{-1}} + K_d\left(1-z^{-1}\right) \right].E(z)$$

Onde as relações entre os parâmetros do modelo contínuo e discreto ficam :

$$K_p = K - \frac{KT}{2T_i} = K - \frac{K_i}{2} = ganho \quad proporcional$$

$$K_i = \frac{KT}{T_i} = \quad ganho\ integral$$

$$K_d = \frac{KT_d}{T} = ganho \quad derivativo$$

Sabendo-se que : K = ganho proporcional do modelo contínuo; T = intervalo de amostragem.

3.2.3.2 Otimização

A aplicação de algoritmos de otimização em sistemas de controle é interessante quando as ferramentas convencionais de projeto de controladores encontram limitações ou impossibilidade de uso. A presença de sistemas não lineares e variantes no tempo são elementos que contribuem para estas limitações. Auxiliam no projeto e identificação de sistemas de controle complexos, por exemplo, *gain scheduled,* controle multimodo, controle adaptativo, alocação de pólos e zeros de controlador,... etc.

Neste trabalho será dado enfoque a utilização de algoritmos de otimização para ajuste dos parâmetros de uma estrutura de controlador (determinação dos parâmetros ótimos), em função de especificações de desempenho definidas no tempo (máximos sobressinal, tempo de subida, tempo de acomodação,... etc) na presença de elementos não lineares (p. ex. saturação e limite de taxa de variação).

A figura 18 ilustra um sistema *single loop* com esta estrutura. O objetivo é o projeto de um controlador PID (variáveis de ajuste Kp, Ti e Td) de forma que o sistema em malha fechada encontre as especificações de rastreamento (especificações de desempenho no tempo comentadas) a um degrau aplicado à entrada. Os parâmetros iniciais para o processo de otimização podem ser obtidos através da aplicação da técnica de Ziegler Nichols que fornece valores aproximados, conforme mencionado anteriormente.

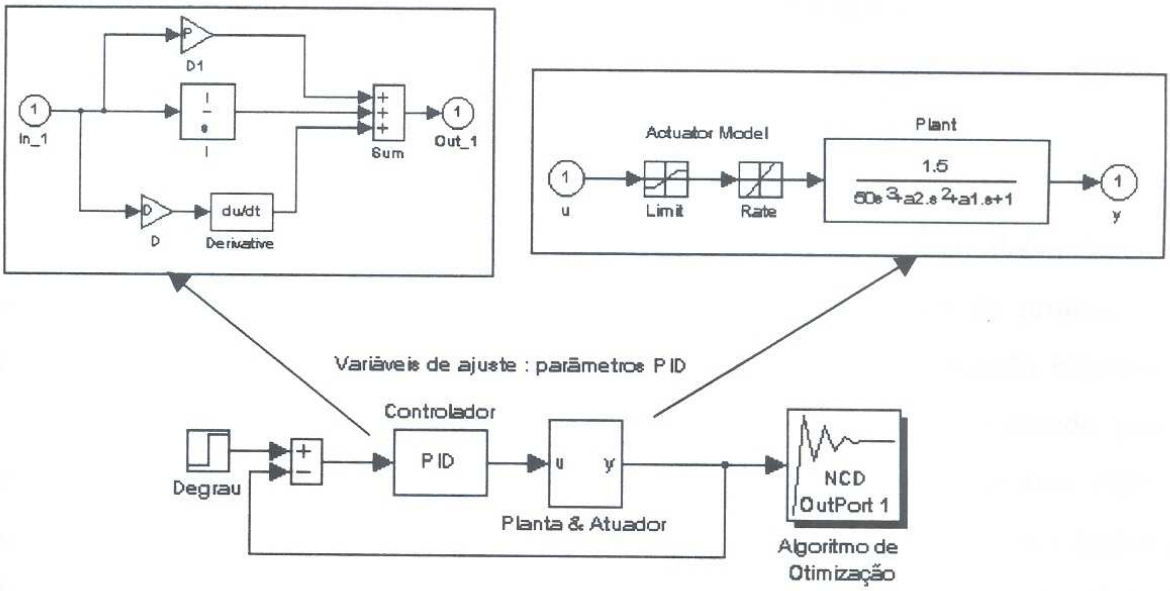


Figura 18 : Projeto de controlador PID – aplicação de otimização

♦ ***Otimização paramétrica***

O processo de otimização em questão denomina-se otimização paramétrica e é utilizada para determinação de um conjunto de parâmetros de projeto, $x = \{x_1, x_2,...,x_n\}$, que pode de certa forma ser definido como ótimo. Em um caso simples isto pode ser a minimização ou maximização de algumas características do sistema que são dependentes de x. Em uma formulação mais avançada uma função objetivo $f(x)$, a ser minimizada ou maximizada, pode estar sujeita a restrições na forma de restrições de igualdade, $G_i(x) = 0$ $(i = 1,...,m_e)$, ou restrições de desigualdade, $G_i(x) <= 0$ $(i = m_e + 1,...,m)$, e/ou limites de parâmetros, x_l, x_u.

Uma descrição de Problema Geral (GP – *General Problem*) é colocado como :

$$\begin{aligned} &minimize && f(x) \\ &x \in && \Re^n \end{aligned}$$

tal que :

$$\begin{aligned} &G_i x = 0, && i = 1,...,m_e \\ &G_i x \le 0, && i = m_e + 1,...,m \\ &x_l \le x \le x_u \end{aligned}$$

onde: x é o vetor de parâmetros de projeto, ($x \in \Re^n$), $f(x)$ é a função objetivo que retorna um valor escalar ($f(x)$: $\Re^n \rightarrow \Re$), e o vetor função $G(x)$ retorna os valores das restrições de igualdade e desigualdade avaliadas em x ($G(x)$: $\Re^n \rightarrow \Re^m$).

Uma eficiente e precisa solução para este problema não é somente dependente no tamanho do problema em termos do número de restrições e variáveis de projeto, mas também nas características da função objetivo e restrições. Quando a função objetivo e restrições são funções lineares da variável de projeto, o problema é conhecido como Programação Linear (LP – *Linear Programming*). Programação quadrática (QP – *Quadratic Programming* preocupa-se na minimização da função objetivo quadrática que é linearmente restringida. Para ambos os problemas LP e QP, procedimentos de soluções de confiança são prontamente avaliados. A maior dificuldade para solução está na programação não linear (NP – *Nonlinear Programming*) em que a função objetivo e restrições podem ser funções não lineares das variáveis de projeto. A solução de problemas NP geralmente requer um procedimento iterativo para estabelecer a direção de pesquisa a cada iteração maior. Isto geralmente é alcançado pela solução de uma LP, uma QP ou um sub-problema sem restrições.

Uma técnica de otimização encontrada em [BRANCH et al., 1996] transforma as restrições e saída do sistema simulado em um problema de otimização da forma :

$$\min_{x,\gamma} \quad \gamma \quad s.t. \begin{pmatrix} g < x > -\omega\gamma \leq 0 \\ x_l \leq x \leq x_u \end{pmatrix}$$

A variável $\boldsymbol{x}$ é a vetorização das variáveis de ajuste, enquanto $\boldsymbol{x}_l$ e $\boldsymbol{x}_u$ do limites maiores e menores das variáveis de ajuste. O vetor $\boldsymbol{g(x)}$ é a vetorização do erro de limite de restrição e $\boldsymbol{w}$ é a vetorização das ponderações nas restrições. O escalar γ impõe um elemento de relaxação no problema, que caso contrário impõem que os objetivos são rigidamente encontrados. Este tipo de problema de otimização utiliza um método de Programação Quadrática Seqüencial (SQP – *Sequential Quadratic Programming*), que soluciona um problema QP a cada iteração.

3.2.4 Realização do controlador

De posse do modelo discreto do controlador obtido pelos processos descritos anteriormente, deseja-se a implementação do algoritmo que represente a sua lei de controle. Este processo denomina-se de realização do controlador digital e são envolvidos elementos de *software* (algoritmo) e *hardware* (processador e conversores ADA). A metodologia utilizada neste trabalho torna viável implementações de controladores genéricos a partir de suas funções de transferência.

No campo de processamento digital de sinais, um *filtro digital* é um algoritmo computacional que converte uma sequência de entrada de números em uma sequência de saída de tal forma que as características do sinal são alteradas em um modo prescrito. Ou seja, um filtro digital processa um sinal digital permitindo a passagem de componentes de frequência desejáveis do sinal de entrada e rejeitando outros indesejáveis. Em termos gerais, um ***controlador digital*** é uma forma de filtro digital. A diferença fundamental reside em que no controlador digital o processamento de sinal deve ser em tempo real. Esta abordagem, bem como as características de hardware serão exploradas adiante neste trabalho.

A forma geral da função de transferência discreta do controlador é dada pela relação dos dois polinômios :

$$G(z) = \frac{U(z)}{E(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_m z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}}, \qquad n \geq m$$

Desta função obtém-se a respectiva equação a diferenças que representa a descrição computacional da lei de controle :

$$u(k) = -a_1.u(k-1) - a_2.u(k-2) + \ldots + a_n.u(k-n) + b_0.e(k) + b_1.e(k-1) + \ldots + b_m.e(k-m)$$

A estrutura PID abordada anteriormente, por exemplo, teria os coeficientes determinados em função desta estrutura genérica como segue :

$$G(z)=\frac{U(z)}{E(z)}=\left[K_p+\frac{K_i}{1-z^{-1}}+K_d\left(1-z^{-1}\right)\right]. \quad = \quad \frac{(K_p+K_i+K_d)-(K_p+2K_d).z^{-1}+K_d z^{-2}}{1-z^{-1}}$$

$$=\frac{b_0+b_1z^{-1}+b_2z^{-2}}{1+a_1z^{-1}+a_2z^{-2}}$$

ficando :

$$a_1 = -1;\ a_2 = 0;\ b_0 = K_p + K_i + K_d;\ b_1 = -(K_p + 2K_d);\ b_2 = K_d$$

Cada atraso z^{-1} pode ser representado em um diagrama como um bloco que representa uma unidade de tempo de atraso. Cada elemento pode estar multiplicado por uma constante que representa a ponderação para aquela amostra. A figura 19 mostra este elemento.

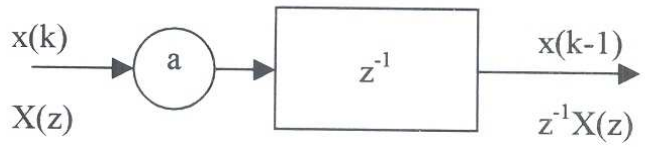


Figura 19: Elemento de atraso

Deste modo a estrutura genérica G(z) pode ser representada através de diagrama de blocos na forma canônica ou não canônica, conforme ilustra a figura 20.

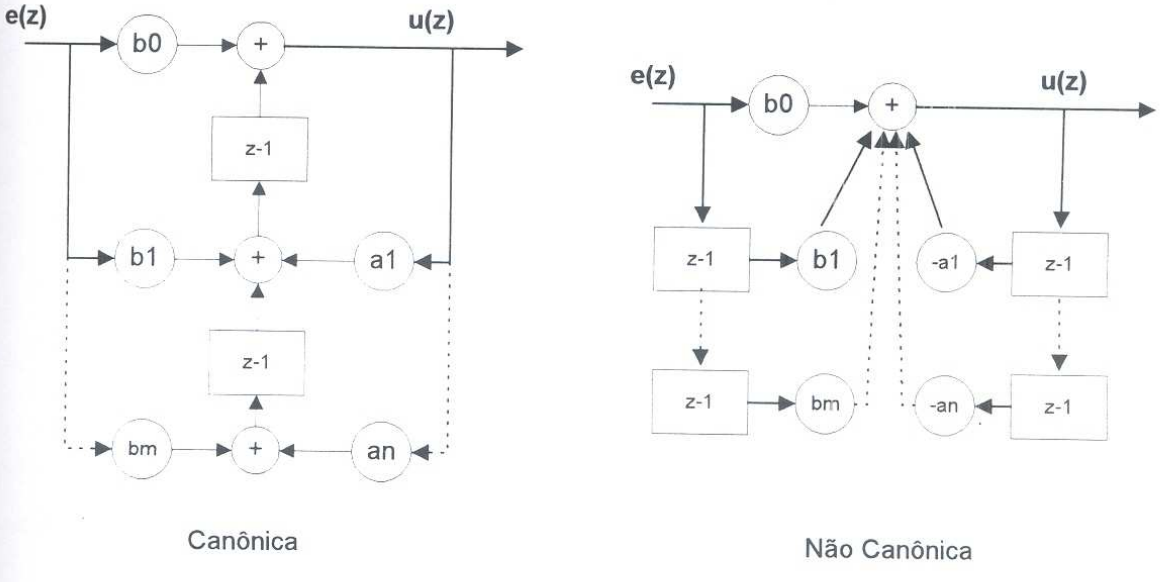


Figura 20 : Lei de controle implementada em forma de diagrama de blocos

3.3 Comentários

O CDSC foi proposto de forma a fornecer uma metodologia para estruturação de um ambiente CACSD implementado como um *toolbox* Matlab. Neste processo são abordados diversos tópicos de sistemas de controle, desde a teoria clássica à teoria avançada como Identificação de Sistemas e Otimização. Conforme já mencionado, a abordagem realizada não teve um aprofundamento detalhado, sendo para tal indicada bibliografia especializada. O objetivo foi de apresentar os tópicos com a adequada profundidade necessária à compreensão dos limites e potencialidades do ambiente VIEnCoD, que os suportará de forma integrada.

A forma de integração dos recursos correlatos aos diferentes tópicos de controle abordados no CDSC, deve ser feita em função dos objetivos que o ambiente propõe: ensino, pesquisa e serviços. O Matlab em seus *toolboxes* apresenta muitas ferramentas (algoritmos em *m-files*) de suporte, mas colocadas de forma isolada e inadequado suporte operacional (de utilização dos *m-files*). A estrutura de *software* e *hardware* do VIEnCoD é justamente criar um ambiente de integração e gerenciamento destes recursos segundo a metodologia imposta pelo CDSC. Este será o tema para os próximos capítulos.

Capítulo 4

4 Projeto do VIEnCoD

4.1 Integração de Ferramentas de Análise e Projeto

Conforme abordado no capítulo dois, os ambientes computacionais desenvolveram-se de forma bastante acentuada nos últimos anos em termos de recursos operacionais, ferramentas matemáticas e incorporação de conjuntos de algoritmos que desempenham atividades específicas – os *toolboxes* (Controle Robusto, Otimização, Processamento Digital de Sinais,... etc). Mais recentemente, recursos gráficos e visuais (GUI's) apresentam-se sob formas bastante elaborada e interativa dando suporte à utilização mais amigável dos algoritmos disponíveis nas diversas áreas. Esta é uma preocupação que tem desencadeado freqüentes atualizações de versões nos pacotes comerciais. Os sistemas de controle estão inseridos neste contexto, com diversos *toolboxes* específicos às diversas áreas de aplicação.

Juntamente com o aperfeiçoamento destes recursos computacionais, desenvolveram-se os sistemas de aquisição de dados com características técnicas melhores comparativamente aos custos. São placas de aquisição multi-funções, DSP's, barramentos de instrumentação VXI/VME,... etc. Nesta fase, surgiram sistemas de *software* para gerenciamento (configuração e operação) desta instrumentação de forma transparente com recursos de análise, tratamento e visualização avançados. O conceito de **Instrumentação Virtual** (*VI – Virtual Instrumentation*) trouxe uma nova filosofia ao sistemas de instrumentação permitindo maior flexibilidade na configuração e operação das plataformas com maior caracterização dos recursos à aplicação específica.

O que se observa, portanto, são duas plataformas poderosas e distintas – uma de análise e projeto de Sistemas de Controle e outra para desenvolvimento e gerenciamento da plataforma de instrumentação e aquisição de dados.

A idéia de integração destas plataformas na estruturação do VIEnCoD atende às diversas necessidades e deficiências expostas nos capítulos anteriores, relativas à ferramentas CACSD, destacando-se alguns pontos de suporte:

- Atendimento ao CDSC, explorado no capítulo anterior, onde sugere-se pela metodologia aplicada, a integração de elementos físicos ao ambiente de simulação;
- Suporte que ambas as plataformas proporcionam ao desenvolvimento de um ambiente CACSD multifuncional atendendo aos objetivos de um laboratório voltado ao ensino, pesquisa e serviços na área de controle;
- Serviços na área de controle junto a comunidade industrial onde se faz necessário dispositivos de aquisição, análise e armazenamento de dados, além de capacidade de interfaceamento aos diversos elementos existentes da planta instalada (CLPs, Barramentos de Campo,... etc).

Para atendimento de todos os objetivos e adaptação aos diversos contextos colocados é necessário uma metodologia adequada que norteie a integração das plataformas citadas. Para tal, é necessário o conhecimento das soluções existentes em cada uma, bem como suas capacidades e limitações. Esta abordagem é feita nos dois itens seguintes e após, a proposta de projeto do VIEnCoD é delineada.

4.1.1 Ambientes de desenvolvimento baseado em Instrumentação Virtual

O rápido desenvolvimento dos processadores paralelamente aos recursos computacionais em ambiente Windows 95/NT, permitiu que funcionalidades de instrumentos fossem colocados via *software* em PC. Desta forma, é possível o desenvolvimento de painéis personalizados de instrumentos (osciloscópios, geradores de funções,... etc) onde se procedem a configuração dos mesmos quanto a aquisição, tratamento e visualização de dados. Este conceito chama-se de *Instrumentação Virtual*.

A maior diferença entre a instrumentação clássica e a baseada em Instrumentação Virtual é a flexibilidade. A combinação de recursos de aquisição, análise e apresentação

de dados é feita de maneira bastante amigável permitindo adaptações e alterações contínuas na instrumentação sem prejuízo de tempo e custo. Existe atualmente uma ampla gama de soluções de *hardware* (placas DAQ baseadas em PC, barramentos VXI/VME,... etc) baseadas em instrumentação virtual cobrindo um espectro muito grande de aplicações. Cada componente vem com seu driver de comunicação (aberto) dispensando este tipo de preocupação. A integração, configuração e operação destes elementos que compõem a plataforma de instrumentação é feita em ambiente de desenvolvimento próprio.

Basicamente, existe duas filosofias que definem a forma de desenvolvimento que cada ambiente (*software*) suporta. Uma das filosofias é um ambiente de programação gráfica onde o sistema é desenvolvido baseado em blocos funcionais; não é necessário o conhecimento de sintaxes e códigos de programação, tornando o ambiente um montador gráfico. A outra filosofia é a programação baseada em linguagem de programação convencional (C, C++, Visual Basic, Pascal). Operações de baixo nível de análise, aquisição e apresentação são encapsuladas em bibliotecas de alto nível cujos códigos podem ser acessados. Ferramentas permitem a geração de código automático de GUI's desenvolvidas e de configuração de funções. Em suma, as potencialidades de ambos os ambientes é a mesma; muda-se apenas a forma de programação.

A seguir são colocadas os principais ambientes de desenvolvimento representantes de cada filosofia sendo analisado com maior profundidade o LabWindows/CVI 4.0, utilizado pelo VIEnCoD.

4.1.1.1 LabWindows/CVI

O LabWindows/CVI (*C for Virtual Instrumentation*) é um ambiente de desenvolvimento ANSI C de programação visual integrado de 32 bits baseado em arquitetura VISA (*Visual Instrumentation Software Arquitecture*). Possui bibliotecas de análise, drivers de instrumentos e ferramentas para interfaces gráficas do usuário (GUIs) intuitivas.

Sua estrutura funcional visa dar suporte a consecução de dois objetivos principais :

- Proporcionar ferramentas de desenvolvimento integrado para construir aplicações Windows de modo fácil e rápido. O LabWindows/CVI oferece:
  - um editor próprio para construir *Graphical User Interfaces* (GUI's);
  - *CodeBuilder*, que é uma ferramenta automática para geração de programas;
  - painéis de função para construir chamadas de função para cada uma das funções da biblioteca do LabWindows/CVI.

- Fornecer *framework* para fácil reutilização de módulos e bibliotecas de código. Os desenvolvedores podem englobar bibliotecas de funções em drivers de instrumentos com painéis de função customizados para geração de código e execução interativa de chamadas a funções de fornecimento de documentação *online* para as funções.

Sua arquitetura de *software* aberto permite o intercâmbio de código, funções, bibliotecas DLL 32 bits – arquivos *.obj*, *.lib*, *.dll* – entre os compiladores padrão C/C++ a saber:

- Microsoft Visual C++;
- Borland C++;
- Watcom C++ ;
- Symantec C++.

Os recursos LabWindows/CVI podem, desta forma, serem acessados por estes ambientes. O seu compilador possibilita a criação de arquivos executáveis e DLL's de 32 bits padrão Windows 95/NT. Este recurso permite que os drivers de instrumentos, colocados sob bibliotecas de funções, possam ser exportados a outro ambiente; da mesma forma os aplicativos desenvolvidos pelo usuário.

Existem ainda bibliotecas para programação a nível de sistema operacional e operações de baixo nível em *hardware*. Para desenvolver aplicações Windows de baixo nível o usuário pode invocar funções SDK diretamente do ambiente, e através delas acessar API's de programação do Windows. Na integração com *hardware*, é possível o acesso a localizações da memória física e perfazer I/O sobre o Windows 95 e NT.

As bibliotecas de I/O incluem sistemas GPIB, VXI, RS-232 e placas DAQ (*Data Acquisition*) além da disponibilidade de mais de 650 *drivers* para instrumentos de fabricantes diversos através de bibliotecas e interfaces de controle amigáveis. As funções do *driver* do instrumento suportam a sintaxe de comando do mesmo, protocolo de interface I/O, *data parsing* e *scaling*; sendo acessível e permitindo modificações para utilização de novos instrumentos. Os dados de uma aplicação podem ser compartilhados entre outros computadores ou aplicativos através de recursos DDE, TCP/IP, OLE além do controle remoto de aplicativos através da automação ActiveX.

Utilizando a biblioteca VISA, a interface de *software* independente endossado pela Aliança de Sistemas *VXIplug&play* , pode comunicar com instrumentos qual seja a conexão física. Por exemplo, a mesma aplicação pode ser utilizada para controlar um *link* serial ou *link* GPIB. No caso de VXI, a mesma aplicação pode controlar um instrumento de um controlador VXI embedded – um PC remoto utilizando um *link* MXI-2 – ou um PC utilizando um *link* GPIB.

Funções avançadas para análise e tratamento de dados são dispostas em bibliotecas:

- *Geração de sinal* : onda senoidal, rampa, pulso, ruído branco gaussiano e uniforme,... etc;
- *Windowing* : janelas de Blackman, Hamming, Hanning, Kaise-Bessel,... etc;
- *Filtros* : FIR, IIR, Butterworth, Bessel,... etc;
- *Estatística* : funções para tratamento estatístico
- *Processamento de Sinal* : FFT, Convolução, AutoCorrelação,... etc;
- *Ajuste de curvas* : Ajuste por Mínimos quadrados, exponencial,... etc;
- *Álgebra Linear* : funções para tratamento matricial.

Com relação a Controle de Processos Industriais e Automação de Fábrica, o LabWindows fornece uma gama de recursos que permitem funcionalidades características ao controle e supervisão do chão de fábrica. São *toolkits* reunindo bibliotecas específicas para :

- *Controle Estatístico de Processo* (*SPC – Statistical Process Control*);

- *Controle PID* : diversas funções implementam algoritmos PID com ferramentas *bumbless transfer* e *antireset windup;* compensadores avanço-atraso; controle *feedforward;* cascata *multiloop*,... etc;
- *Servidores para Automação Industrial* : dispositivos para comunicação com ferramentas industriais como instrumentação de processo, PLCs, controladores *single loop* e variedades de dispositivos I/O e aquisição de dados. Como exemplo temos interface Fieldbus *Foundation*; interface CAN; interface Profibus; servidores para controle de PLCs Allen-Bradley, Siemens,... etc;

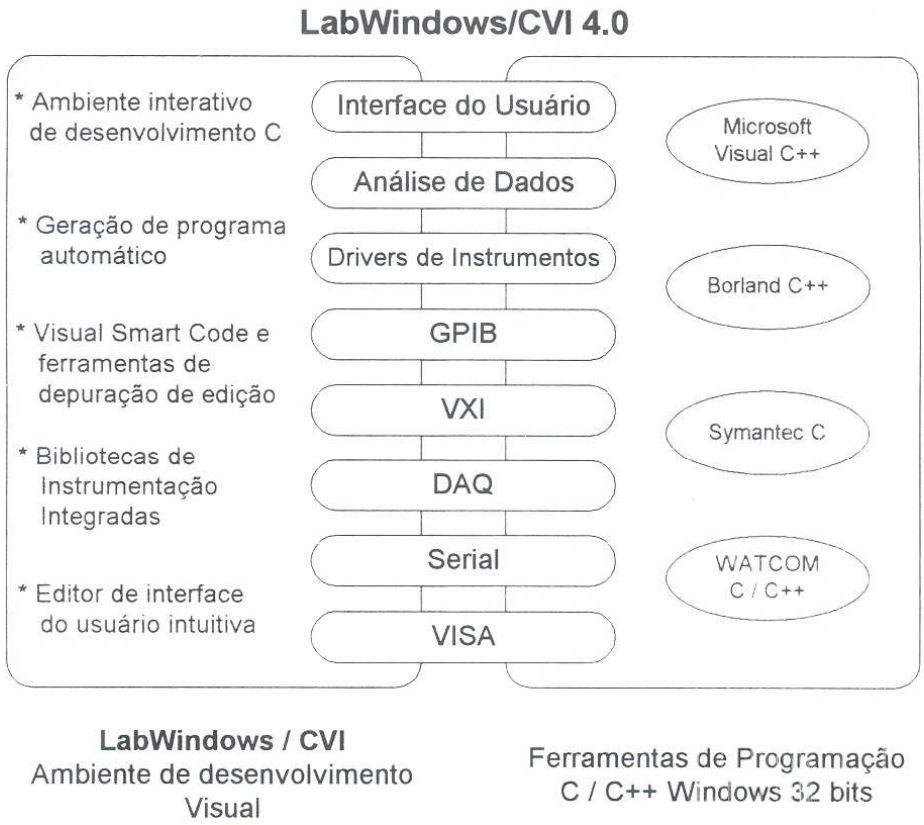


Figura 21. Estrutura do LabWindows / CVI

- *Desenvolvimento na Internet* : ferramentas para programação sobre Internet – envio de e-mail, arquivos FTP diretamente dos aplicativos.

Toda a estrutura comentada acima encontra-se representada na figura 21 dando uma noção global das possibilidades de recursos. Em suma, é um ambiente com características funcionais e operacionais voltadas ao desenvolvimento de aplicativos voltados a

instrumentação e aquisição de dados. As bibliotecas de funções expressam um pouco destas características e são colocadas no Apêndice I desta dissertação.

4.1.1.2 LabVIEW

O LabVIEW é um ambiente de desenvolvimento de programação gráfica baseado na linguagem de programação *G* – modelo de programação *dataflow*, para aquisição de dados e controle, análise e apresentação de dados.

O desenvolvimento de um programa baseia-se na montagem e interconexão de elementos de *software* chamados de VIs (*Virtual Instruments*) bem como a criação de painéis de interface do usuário. Todas as funções citadas no LabWindows/CVI são disponíveis e encapsuladas em blocos (*objetos*) : desde funções aritméticas simples à funções avançadas de aquisição e análise de dados, rede e operações em arquivos I/O .

A linguagem de programação *G* permite outra filosofia diferente da arquitetura linear das linguagens baseadas em texto. A execução ordenada no LabVIEW é determinada pelo fluxo de dados entre blocos e não pelas linhas seqüenciais de texto, permitindo a criação de diagramas que possuam a ocorrência simultânea de eventos. Esta característica o torna um ambiente multitarefa – múltiplas linhas de execução e múltiplos VIs executados simultaneamente.

A modularidade e hierarquia fazem parte do ambiente de desenvolvimento. Analogamente ao Simulink, sistemas podem ser compostos de vários subssistemas. No caso, vários VIs podem compor um novo VI com funcionalidade composta. As características são encapsuladas e adequadas as interfaces de entrada e saída.

Outro recurso disponível é a existência de compilador que gera código C otimizando seções críticas de tempo e código. Demais funcionalidades encontradas no LabWindows / CVI também são disponíveis em bibliotecas mas com maior quantidade de opções em *toolkits*.

4.1.1.3 Outros

Existem ainda outras plataformas baseadas no conceito de instrumentação virtual. Um exemplo é a fornecida pela *Hewlett Packard*, o ***HP VEE***, com ambiente de desenvolvimento gráfico análogo ao LabView voltado aos sistemas HP de instrumentação. Está em estudo a adesão da empresa ao *IVI Foundation,* promovido pela National Instrments, onde propõem-se uma padronização aos drivers de instrumentos.

O ***ComponentWorks***, da *National Instruments,* representa uma coleção de controles *ActiveX* de 32 bits para aquisição de dados, análise e apresentação para trabalhar com Visual Basic, Visual C++, Borland Delphi e Microsoft Internet Explorer. Desta forma, é possível o desenvolvimento de plataforma de instrumentos virtuais baseados nestes ambientes.

4.1.2 Recursos Matlab e Simulink

Um ambiente CACSD que atenda todas as fases do CDSC, em ambiente amigável voltado ao ensino-pesquisa-serviços, exige a escolha de uma plataforma que suporte toda a funcionalidade e objetivos propostos.

Neste sentido, o Matlab – Simulink da *The Mathworks* reuniu todos os requisitos sendo a plataforma adotada. As razões, previamente abordadas no capítulo 2, são novamente colocadas em ordem de relevância:

- Plataforma com maior penetração no meio acadêmico. Isto permite um intercâmbio maior de contribuições baseadas no ambiente;
- A maior parte das contribuições científicas na área de controle têm o Matlab-Simulink com ferramenta de apoio;
- Maior disponibilidade de *toolboxes* específicos à área de controle;
- Ambiente de programação de alto nível com forte estrutura visual – recursos gráficos avançados e dispositivos para elaboração de GUI's sem o conhecimento de código;
- Recursos amigáveis para simulação com HIL;
- Ambiente aberto – códigos C e Fortran.

Os recursos sob maior enfoque são desenvolvidos sumariamente a seguir. No item 4.2 a forma de integração destes recursos na estrutura do ambiente exigirá um maior detalhamento dos mesmos.

O Matlab-Simulink é um ambiente de computação técnica integrada que combina computação numérica, recursos gráficos avançados e linguagem de programação de alto nível. Existe elevado número de ferramentas e funções disponíveis para : análise de dados e visualização; computação simbólica e numérica; gráficos científicos e de engenharia; modelagem, simulação e criação de protótipos; programação, desenvolvimento de aplicações e criação de interfaces gráficas do usuário (GUI's).

Os *toolboxes* relativos ao desenvolvimento de sistemas de controle cobrindo as técnicas de análise e projeto clássicas, modernas e avançadas são :

- *Control System Toolbox* : algoritmos para a modelagem, análise e projeto de sistemas de controle clássico e moderno.
- *Fuzzy Logic Toolbox* : ambiente gráfico intuitivo para projeto de controladores inteligentes.
- *Nonlinear Control Design Blockset* : Otimização baseada no domínio do tempo para projeto de sistemas de controle lineares ou não lineares através do Simulink.
- *Robust Control Toolbox* : ferramentas especializadas para projeto de sistemas de controle robusto realimentado multivariável.
- *μ-Analysis and Synthesis Toolbox* : Ferramentas para projeto de controle robusto utilizando controle ótimo e o valor singular estruturado.
- *LMI Control Toolbox* : soluções para problemas de otimização no projeto de controle robusto.
- *QFT Control Design Toolbox* : conjunto de funções para o projeto prático de controladores robustos em realimentação.
- *Model Predictive Control Toolbox* : funções para o controle de processos multivariáveis na presença de condições limitadoras.
- *Neural Network Toolbox* : ambiente compreensível para simulação, projeto e pesquisa de redes neurais utilizando Matlab.

Alguns destes e mais um conjunto de outros *toolboxes* e aplicativos baseados no Matalab 5.2 e Simulink 2.2 foram utilizados efetivamente no projeto. São eles :

- *Control System Toolbox*;
- *System Identification Toolbox* : algoritmos para suporte do processo de identificação de sistemas descrito no capítulo 3.
- *Optimization Toolbox* : rotinas para para implementação de variados tipos de métodos para minimização e maximização de funções não lineares gerais. Será enfocado a utilização em problemas de mínimos quadrados não lineares em funções multiobjetivos.
- *Nonlinear Control Design Blockset;*
- *Matlab Compiler* : conversão de funções matlab (*m-files*) em funções DLL e código otimizado padrão ANSI C e C++ para utilização em aplicações *standalone*.
- *Real Time Workshop : geração de código C diretamente de diagramas em boloco Simulink.*
- *GUIDE* : ferramenta para criação de interfaces gráficas do usuário.

Na versão 5.3 do Matlab lançada recentemente, existem alguns aplicativos e *toolboxes* que apresentam grande valia à soluções em implementações[9]. São elas:

- *Matlab Runtime Server* : permite a geração de aplicações executáveis *standalone* desenvolvidas inteiramente no ambiente Matlab. *Com este aplicativo, um toolbox pode ser distribuído sob forma de solução integrada sem a necessidade do ambiente (licença Matlab) instalado.*
- *Data Acquisition Toolbox :* reúne um conjunto de ferramentas para controle e comunicação (via ambiente Matlab) com dispositivos de aquisição de dados. Os dispositivos de *hardware* suportados são : placas de aquisição de dados série E e 1200 da *National Instruments*; série VXI E1432/33/34da Hewlett-Packard; placas de som multimídia Windows. *Motivação na integração de sistema de aquisição diretamente no ambiente Matlab.*

[9] A versão utilizada no projeto é a 5.2 não suportando estes aplicativos. O lançamento desta versão data de período posterior à finalização deste trabalho.

4.1.3 Concepção VIEnCoD

A concepção do VIEnCoD foi o resultado do levantamento e estudo das características e recursos disponíveis nas plataformas LabWindows/CVI e Matlab-Simulink de forma a viabilizar a integração de um ambiente de desenvolvimento específico de instrumentação virtual a um ambiente de análise e projeto de sistemas de controle. Este ambiente integrado deverá estar sob condições de tempo real permitindo simulações com HIL.

A adequada aplicação de alguns recursos formam a estrutura da plataforma diferenciando-a das existentes pela caracterização multifuncional agregada – ensino, pesquisa e serviços.

O gerenciamento do VIEnCoD é todo baseado no ambiente Matlab caracterizando-o como um ***toolbox*** com proposta CACSD no atendimento ao CDSC. São rotinas e GUI's desenvolvidas para suporte de cada fase integrando-as de forma modular. O controle e a troca de dados com o sistema de instrumentação virtual desenvolvido no LabWindows/CVI é também absorvido neste ambiente. Alguns fatores levaram a esta concepção :

- Ambiente de programação amigável em linguagem de alto nível no Matlab;
- Forte suporte de programação visual na versão 5.2 do Matlab;
- Todo o processo de desenvolvimento e análise de controle é baseado em recursos e algoritmos Matlab e Simulink. Isto motiva uma estrutura baseada no mesmo sem a necessidade de implementações adicionais de interface entre aplicativos (DDE, AtiveX). O sistema de *hardware* funciona apenas como um módulo de aquisição de testes de validação;
- Estudo de ambientes com proposta CACSD baseados nesta concepção – p. ex. dSPACE;
- Motivação pelo grande número de aplicativos específicos sendo desenvolvidos diretamente em ambiente Matlab, conforme visto no capítulo 2;
- Diponibilização na última versão do Matlab (versão 5.3) do *dispositivo Matlab Runtime Server* permitindo a criação de aplicativos *standalone*;

- Facilidade de integração de novas contribuições ao VIEnCoD – o pesquisador de controle normalmente utiliza o Matlab e Simulink como ferramentas.

A execução de simulação com HIL é realizada pelos recursos de instrumentação virtual desenvolvidos em função da plataforma de aquisição adotada. O processamento destes recursos é realizado pelo mesmo *hardware* (PC) onde trabalha o ambiente de análise e projeto de controle (em processo *offline*) e sob a gerência do sistema operacional Windows. Esta estrutura impossibilita que condições de tempo real, necessárias a operação de sistemas de controle, sejam atingidas. Portanto, torna-se necessário uma atuação no sistema operacional de forma a definir prioridades nos eventos por ele processados e tempos de execução. Desta forma, condições de tempo real são determinadas permitindo a operação da lei de controle através da plataforma de instrumentação virtual com garantia no atendimento das constantes de tempo do sistema e intervalos de amostragem garantidos. Uma abordagem mais aprofundada é desenvolvida no item 4.2.3 adiante.

A disponibilidade de elevado número de drivers de instrumentos de diversos fabricantes no LabWindows/CVI agrega facilidades na escolha e desenvolvimento na estrutura de *aquisição,* evitando trabalhos de implementações neste sentido. Esta facilidade também se estende aos protocolos de comunicação com elementos industriais de controle como CLP (Controladores Lógico Programável), barramento Fieldbus, Profibus,... etc. Isto viabiliza a interface mais efetiva de uma ferramenta CACSD com elementos físicos reais presentes em um processo industrial. Segunda a concepção adotada, a estratégia de controle é implementada sob a estrutura de *hardware* baseada em instrumentação virtual.

Estas características tornam o VIEnCoD uma plataforma aberta em termos de *hardware* implicando em uma diferenciação perante outras plataformas onde existe uma estrutura específica de *hardware* disposta de microprocessadores, DSP's e sistemas de aquisição dedicados. A validação de controladores nestes casos se dá, p.ex, através de implementação de algoritmos em C que são compilados sob a estrutura definida pelo *hardware* proprietário.

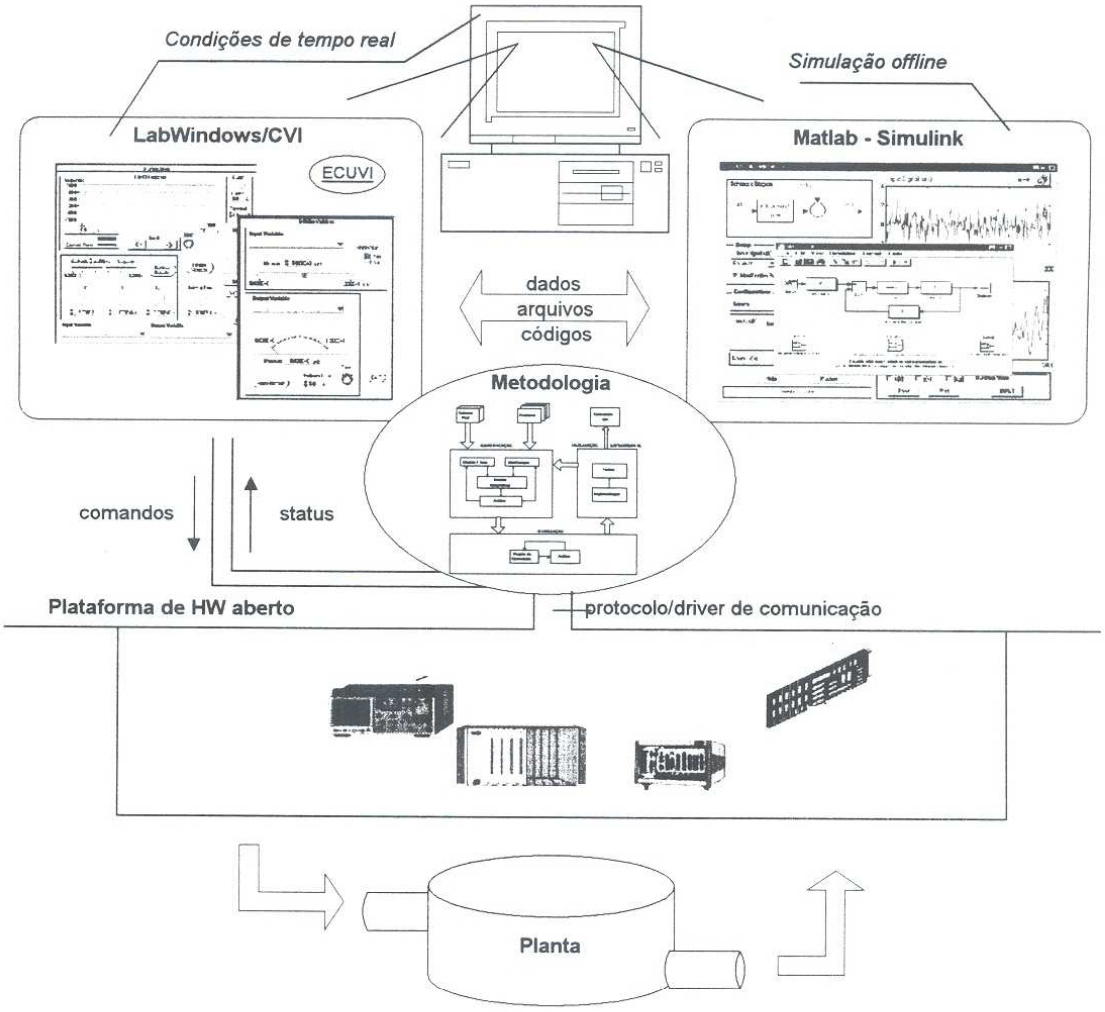


Figura 22. Concepção VIEnCoD

A estrutura em C de programação do LabWindows permite o suporte de algoritmos de controle cujos parâmetros e estrutura foram definidos pelo processo *offline* do CDSC (detalhado no item 4.2.3 adiante), para formação da ECUVI (*Electronic Control Unit Virtual Instrumentation-based)*, estrutura discutida no item seguinte.

Também é possível, com o tratamento preliminar de código, o suporte de rotinas desenvolvidas no Matlab e Simulink (tranformadas em código C pelo Matlab Compiler e RTW). Este procedimento é a base da estrutura de plataformas CACSD que apresentam dispositivos de *hardware* (processadores) à parte do ambiente de desenvolvimento (PC). Estes sistemas necessitam de recursos de *software* avançado para manipulação (sintaxe e adequação à estrutura de tempo real) do código C gerado pelo Matlab Compiler ou RTW ao *hardware* (processador) presente. É o caso, por exemplo, do dSPACE com o RTI (*Real Time Interface*) que representa uma extensão do RTW. A adoção pelo VIEnCoD deste

recurso não encontrou em sua concepção final, necessidades de implementação que justificassem este desenvolvimento. Este tópico é tratado em detalhes adiante no item 4.2.3.2.

A figura 22 ilustra de maneira global a concepção da estrutura do CACSD, onde as células principais de sua estrutura são colocadas. Os detalhes nas interfaces entre as plataformas, como a troca de dados, arquivos e intercâmbio de códigos são discutidos no item seguinte. Da mesma forma os recursos Matlab-Simulink e LabWindows/CVI previamente colocados, são explorados de forma mais específica e aprofundada para a descrição dos elementos funcionais do VIEnCoD.

4.2 Características de Projeto

Os aspectos operacionais e estruturais dos ambientes de computação numérica atingiram elevado grau de desenvolvimento pela contribuição da informática e engenharia de *software* aos métodos computacionais. Quando existe a proposta destes ambientes ao suporte da análise e desenvolvimento de sistemas de controle – ferramenta CACSD, é necessário que existam algumas características, métodos e funcionalidades (por exemplo, a programação orientada a objetos). A flexibilidade do ambiente de programação orientado a objeto no Simulink facilita o rápido desenvolvimento de classes de algoritmos de controle pela simples construção de blocos (p. ex. blocos de função de transferência): controle PID, *feedforward*, modelo interno, controle *anti-windup*,... etc.

Para atendimento de todo o CDSC, especialmente no suporte a sistemas complexos, é desejável que o ambiente apresente uma estrutura lógica e seqüencial que permita o acesso às tarefas pertinentes a cada etapa de forma individualizada e consistente à metodologia empregada. Por exemplo, o usuário deseja a otimização de um compensador mas já possui a modelagem da planta; a etapa de identificação não é necessária e o fluxo de projeto não é prejudicado. Neste processo são envolvidas muitas tarefas de cunho decisório como formulação do problema, validação de modelos, escolha da metodologia de projeto, testes e simulações,... etc. Este contexto torna necessário que o projeto do ambiente agregue algumas características :

- Ambiente modular e flexível;

- Interação com ferramentas de controle já avaliadas;
- Interação com usuário amigável e uniforme;
- Capacidade de representação de conhecimento.

Segundo [ARRUDA et al., 1994] a arquitetura utilizada por ambientes desenvolvidos com proposta CACSD é tipicamente composto por três partes :

1. Um conjunto de ferramentas para implementação de tarefas como simulação, análise e controle;
2. Uma interface para fazer a tarefa de especificação mais adaptada ao usuário final;
3. Um programa principal para coordenar as ferramentas e execução de interfaces.

O conjunto de ferramentas disponíveis (funções, subrotinas,... etc) sob características modulares, devem ser mapeadas e executadas segundo uma sequência de ações do usuário através do programa principal. Portanto, toda informação dos recursos do ambiente devem estar presentes em suas linhas de programa. Desta forma, toda modificação futura (inclusão de novas ferramentas, novas opções do usuário ou redução do sistema) requer atuação apenas no programa principal. Se estas modificações não são suportadas apropriadamente o ambiente CACSD é ineficiente e difícil de manter.

Em vista do rápido desenvolvimento, tanto no campo teórico (p. ex. novos métodos e algoritmos de controle) como nas composições de *hardware* (sistema de aquisição - instrumentação) e *software* (recursos gráficos avançados e sistema computacional eficiente), sugere-se a caracterização do ambiente CACSD como "aberto" [RUTZ et al., 1995]. "Aberto" neste contexto significa :

- Ser possível a formulação de sistemas das mais heterogêneas naturezas, pela padronização e linguagem de descrição consistente. O usuário pode fazer uso de bibliotecas de modelos existentes podendo também incrementá-la;
- Facilidades de conversão e integração de modelos;
- Para integração de programas, o usuário pode desenvolver seu próprio aplicativo e estabelecer uma interface padronizada e encapsulamento para estes componentes;
- O ambiente deve estar habilitado para suportar interfaces com outros aplicativos. A troca de dados deve ser bidirecional;

- A mudança na plataforma de *hardware* deve ser suportada sem trocas conceituais e transparência a nível de protocolo.

Todos os aspectos de projeto colocados são trabalhados segundo a disponibilidade de recursos e limitações de implementações baseada na concepção do VIEnCoD. Nos itens seguintes todo o projeto é detalhado.

4.2.1 Módulo gerenciador VIEnCoD em ambiente visual Matlab

Conforme já apresentado, a concepção de projeto do VIEnCoD baseia-se no conjunto de funções e rotinas desenvolvidas no ambiente Matlab destinadas ao gerenciamento do CDSC, caracterizando-o com um ***toolbox*** com proposta CACSD. Portanto, a execução de tarefas e simulações é toda suportada pelos diferentes algoritmos e recursos gráficos disponíveis e distribuídos entre os diferentes *toolboxes* já existentes (*Control Toolbox*, *System Identification Toolbox*,... etc). É elevado o número de funções com entrada de parâmetros e objetivos específicos. Apesar da disponibilidade de eficiente estrutura de *helps* , é necessário que a entrada dos parâmetros de cada função ou um conjunto integrado, seja suportada por GUI's intuitivas que carreguem conteúdo informativo e adequado tratamento de erros.

Previamente à definição desta atual estrutura, houveram trabalhos de implementação[10] com a adoção de uma concepção diferente. Nesta proposta todo o ambiente foi desenvolvido no LabWindows / CVI, que acumulava as tarefas de controle do *hardware* (aquisição e simulação com HIL) e o suporte da estrutura lógica e seqüencial do CDSC. Os recursos Matlab eram evocados através de comunicação DDE (*Dynamic Data Exchange*). Em virtude dos problemas e limitações deparados, esta concepção foi abandonada dando lugar ao atual trabalho.

Portanto, do levantamento das necessidades impostas por um ambiente CACSD com proposta multifuncional, partiu-se para um estudo aprimorado dos recursos (programação e ferramentas) Matlab/Simulink.

---

[10] ERZINGER, A.; JANSSON, J. *VIEnCoD - VXI-based Integrated Environment for Controllers Design*. Projeto Final do Curso de Engenharia de Computação, Pontifícia Universidade Católica do Paraná, 1997.

A pouca quantidade de referências bibliográficas com insuficiente aprofundamento sobre a programação avançada no Matlab impôs dificuldades no trabalho de criação e implementação do ambiente, mas compensado pelas vantagens já expostas. O principal auxílio neste processo de familiarização com o ambiente foi o estudo de aplicativos exemplos (estruturas simples) disponibilizados no site da *The Mathworks*[11], mas principalmente o estudo detalhado nas estruturas de programas existentes (p. ex.: *ltiview.m*, *ident.m*) através de recursos de depuração e verificação da funcionalidade de cada linha e comandos.

O envolvimento de conteúdos diversos de controle, desde tópicos da teoria clássica e moderna à teoria avançada de controle como Sistemas Discretos, Identificação e Otimização, exigiu estudo aprofundado da teoria e ferramentas presentes nos *toolboxes* específicos a cada área. Isto permitiu a disponibilização adequada[12] deste conhecimento de forma aplicada no ambiente. Esta tarefa representou a dificuldade de maior proporção no projeto e implementação do VIEnCoD.

Neste contexto, adotou-se uma metodologia de estudo para que fosse criado uma especialização maior sobre as potencialidades do Matlab de forma a otimizar a criação e implementação de idéias. Tal metodologia baseou-se no cumprimento das seguintes fases colocadas em ordem cronológica:

1. Verificação de todos os comandos de propostas gerais com o auxílio do *help* / *helpwin* juntamente com estudo dos manuais correlatos. Tais comandos abrangem todas rotinas à parte das específicas presentes nos *toolboxes* : manipulação de matrizes, operadores e conversão de tipos, manipulação de gráficos, funções matemáticas elementares, manipulação de arquivos,... etc;
2. Estudo de todos os comandos relativos a cada *Toolbox* utilizado no projeto. Foi a principal fase onde foram definidos todos os recursos para o suporte do CDSC. *Toolboxes* abordados : *Control Toolbox*, *Identification Toolbox*, *Optimization Toolbox*, *NCD Blockset*. Demais *toolboxes* relativos a sistemas de controle também foram analisados podendo ser utilizados em implementações futuras (p. ex. *Fuzzy Toolbox*);

---

[11] The Mathworks, http://www.mathworks.com, Out. 1999.

[12] Para atendimento dos objetivos propostos: educacional, pesquisa e serviços.

3. Verificação dos recursos Simulink;
4. Verificação de todos os demos existentes e outros obtidos externamente (p. ex. site da *The Mathworks*);
5. Estudo dos recursos de programação visual da forma citada anteriormente em acréscimo à verificação de todos os comandos correlatos;
6. Verificação dos recursos de API como : DDE, Compiler e RTW.

A implementação de maior complexidade e pequenas falhas existentes no ambiente foi determinada pelo tempo, sendo a validação das idéias apresentadas o objetivo principal deste trabalho.

4.2.1.1 *Toolbox* VIEnCoD – aplicação de recursos

O sistema Matlab é composto por recursos que fornecem suporte de alto nível à tarefas que compõem sua estrutura : linguagem, ambiente operacional, suporte gráfico, bibliotecas de funções matemáticas, interface de programa de aplicação (API). No capítulo 2 um *overview* destes recursos foi apresentado.

Para a criação de um ambiente com proposta CACSD foi necessário a utilização de grande parte destes elementos e de modo estruturado sob uma programação visual de alto nível. A estrutura do ambiente é obtida pela integração de *m-files (scripts e functions)* organizados em tarefas específicas (cada fase do CDSC), onde as sintaxes e trocas de dados são transparentes ao usuário através das GUIs e rotinas de programação adequadas. Assim, cria-se um conjunto de novas *m-files* (biblioteca) com objetivo específico caracterizando o VIEnCoD com um novo *toolbox* : *CACSD Toolbox*.

A seguir, as técnicas com maior destaque utilizadas no projeto são abordadas. Após, a estrutura de projeto de *software* é desenvolvida. As abordagens feitas visam elucidar a idéia estrutural do projeto e apresentar alguns dos recursos possíveis no ambiente de programação do Matlab, ainda pouco difundido e utilizado. Um conhecimento de base desta estrutura é fundamental para a continuidade do presente trabalho.

♦ ***Estrutura de programação de m-files***

O arquivos que contém linhas de programa em código Matlab são denominadas de ***m-files***. Estes podem ser de dois tipos diferentes : ***functions*** (aceitam argumentos de entrada e retornam argumentos na saída) e ***scripts*** (executam tarefas definidas sem necessidade de entrada ou retorno de parâmetros, por exemplo, chamada de GUI's).

A grande vantagem presente é que não é necessário a declaração de variáveis sem com isto perder a legibilidade de código e diminuido o tempo do desenvolvimento e depuração de um aplicativo. A identificação do tipo de dado é feito pela interpretação de sua sintaxe (p. ex.: x = {'Fução de transferência', 1 , [1 1 1] , 'vazão' , 'altura'} – a variável x é interpretada como sendo uma *cell array*).

No Matlab, a estrutura das variáveis são definidas através de ***classes*** e ***objetos***. A *classe* descreve a estrutura de um variável e indica o tipo de operações e funções que podem ser aplicadas a mesma. Um *objeto* é uma variável ou uma instância de uma variável. O contexto de ***programação orientada a objeto*** descreve um conceito na escrita de programas que enfatiza o uso de classes e objetos. O *Control Toolbox* define a classe ***lti*** e três subclasses para análise de sistemas LTI (*Linear Time Invariant*). A adição de novas classes se faz pela especificação de uma estrutura que provê armazenamento de dados para o objeto e a criação de um diretório onde ficarão as *m-files* que operam no objeto. Estas *m-files* são conhecidas com ***métodos*** para a classe. Maiores detalhes na criação de classes podem ser vistos em [The MathWorks, 1997c].

A manipulação e conversão de tipos de dados (classes) foi uma necessidade contínua na fase de implementação das ferramentas do VIEnCoD, sendo fundamental o conhecimento das diversas estruturas disponíveis e utilizadas no projeto. Primeiramente as estruturas gerais :

- *double (double array)* : matriz numérica de ponto flutuante dupla precisão; formato padrão no Matlab aplicado a constantes e vetores/matrizes de constantes. Ex.: num = 2; den = [1 1 1].
- *char (character array)* : define as *strings*. Ex.: *input_sys* = 'vazão' .

- *struct (struct array)* : conjunto de dados logicamente relacionados de diferentes tipos. Ex.: *data_plant* = { 'vazão' 'nível' 1 [1 1 1] } .
- *cell (cell array)* : lista de dados compostos. Ex.: *plant* = name: 'tanque' num: 1 den: [1 47] *type*: 'cont' *model*: 'tf'

Estruturas do *Control Toolbox* :

- *tf (tranfer function)* : função de transferência;
- *ss (state-space)* : espaço de estados;
- *zpk (zero-pole-gain)* : zeros, pólos e ganho em DC de uma função de transferência.

Ambas são formas de representação de modelos de sistemas dinâmicos; no caso sistemas LTI definido pela classe. Estas estruturas permitem a especificação de suas propriedades segundo a definição de campos de uma *cellarray*. Quando da especificação e visualização de modelos através de GUIs, figura 23, esta propriedade é utilizada na descrição detalhada do modelo. A função *get* permite o acesso destes campos.

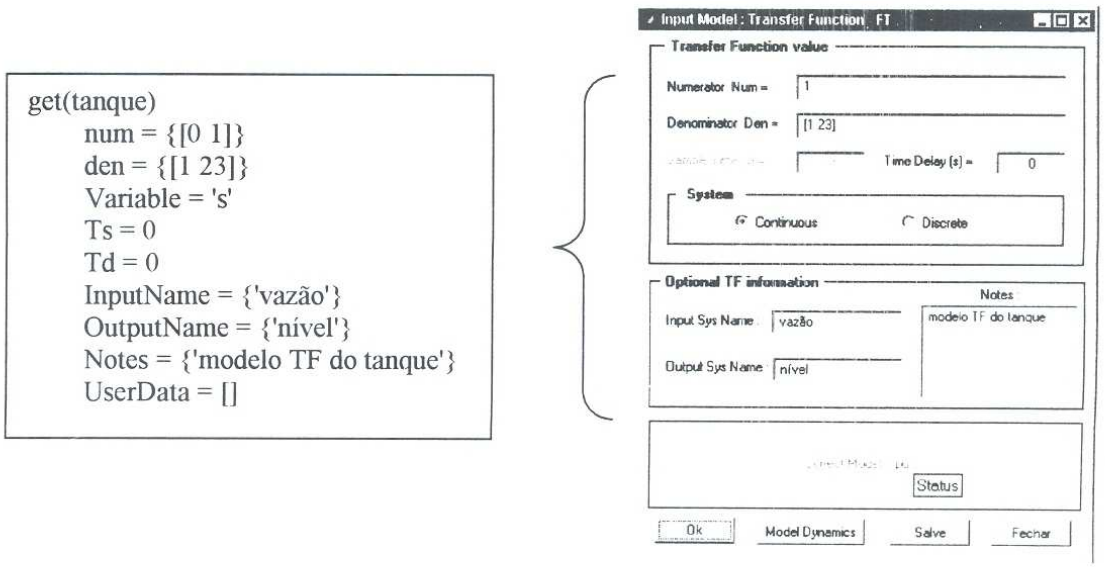


Figura 23 : Propriedades estruturas LTI no auxílio a representação de modelos

A figura 24 ilustra rotinas (etapa do CDSC) onde a manipulação (conversão) de tipos de dados (classes) é feita de forma intensiva. Vale ressaltar que os campos de entrada e saída de dados (elementos de uma GUI) apresentam formato de *string* (*char*

array) sendo necessário conversão para os demais tipos de dados para manipulação interna em programa.

Cada *m-file* possui uma área de memória , separada do *workspace* de base do Matlab (onde alocam-se as variáveis utilizadas), onde a mesma opera. Esta área é chamada de *function workspace*, com cada *function* contendo seu próprio contexto de *workspace*. Durante a utilização do Matlab, as únicas variáveis que podem ser acessadas são aquelas do contexto de chamada, que estão no *workspace* ou em outra função. A variável que é passada para uma *function* deve estar no contexto de chamada, sendo que a mesma retorna os argumentos de saída para o contexto de chamada do *workspace*. Através da definição das variáveis como globais, permite que as mesmas sejam acessadas por mais de um contexto de *workspace*.

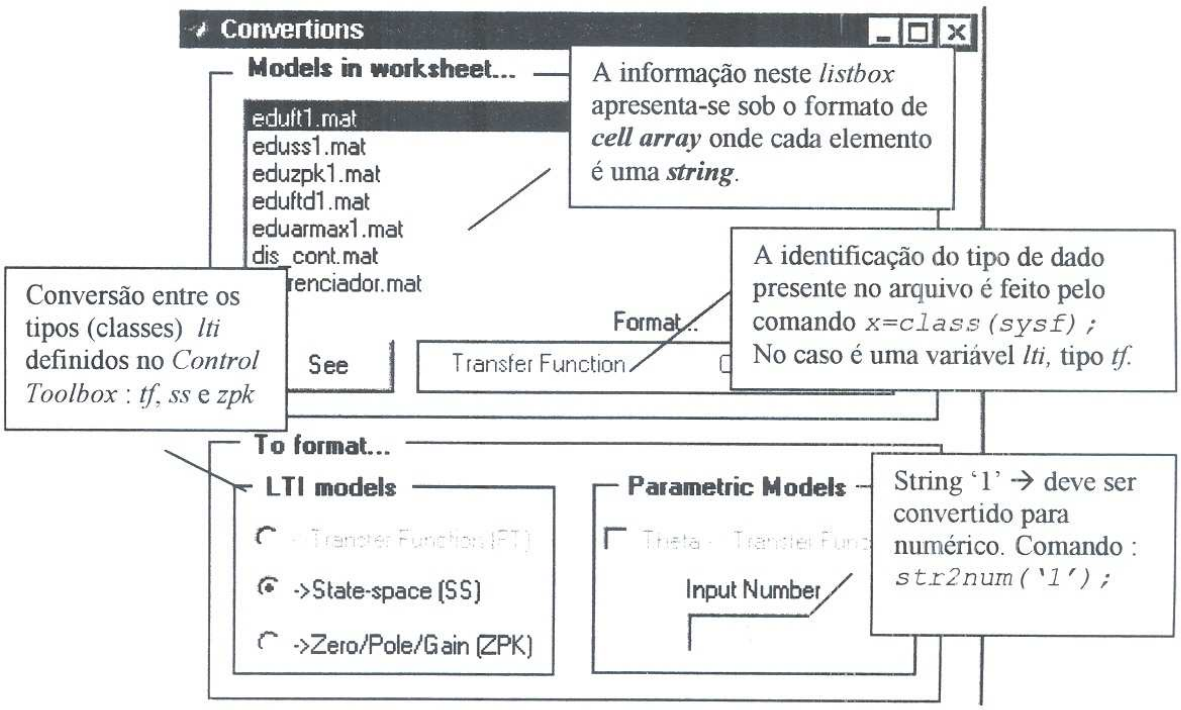


Figura 24 : Manipulação de tipos (classes)

A característica modular do VIEnCoD permitindo o controle lógico e sequencial entre eventos (etapas do CDSC) , implica na passagem contínua de variáveis entre *m-files* (*functions e scripts*). Quando é necessário o acesso do valor de uma variável (que sofre contínuas atualizações durante o CDSC) para execução de um *script*, sem a preocupação da pré-definição na estrutura de chamada da *function*, faz-se uso da definição de variável

global. Este perfil sugere sua caracterização como uma variável auxiliar, podendo ser acessada a qualquer momento sem a rigidez da estrutura de chamada das *functions*. Por exemplo, o VIEnCoD define alguns *flags* que determinam a estrutura de *layout* de algumas GUIs, em função da etapa do ciclo em que se está trabalhando. Estes *flags* são definidos como variáveis globais podendo ser acessados por qualquer contexto de *workspace*. São definidas como:

- *Path* da pasta de trabalho do usuário;
- Identificadores de elementos de GUIs;
- *Flags* de definição : por exemplo, identificação de planta, controlador, sensor e filtro para modelagem e tratamento diferenciado por rotinas.

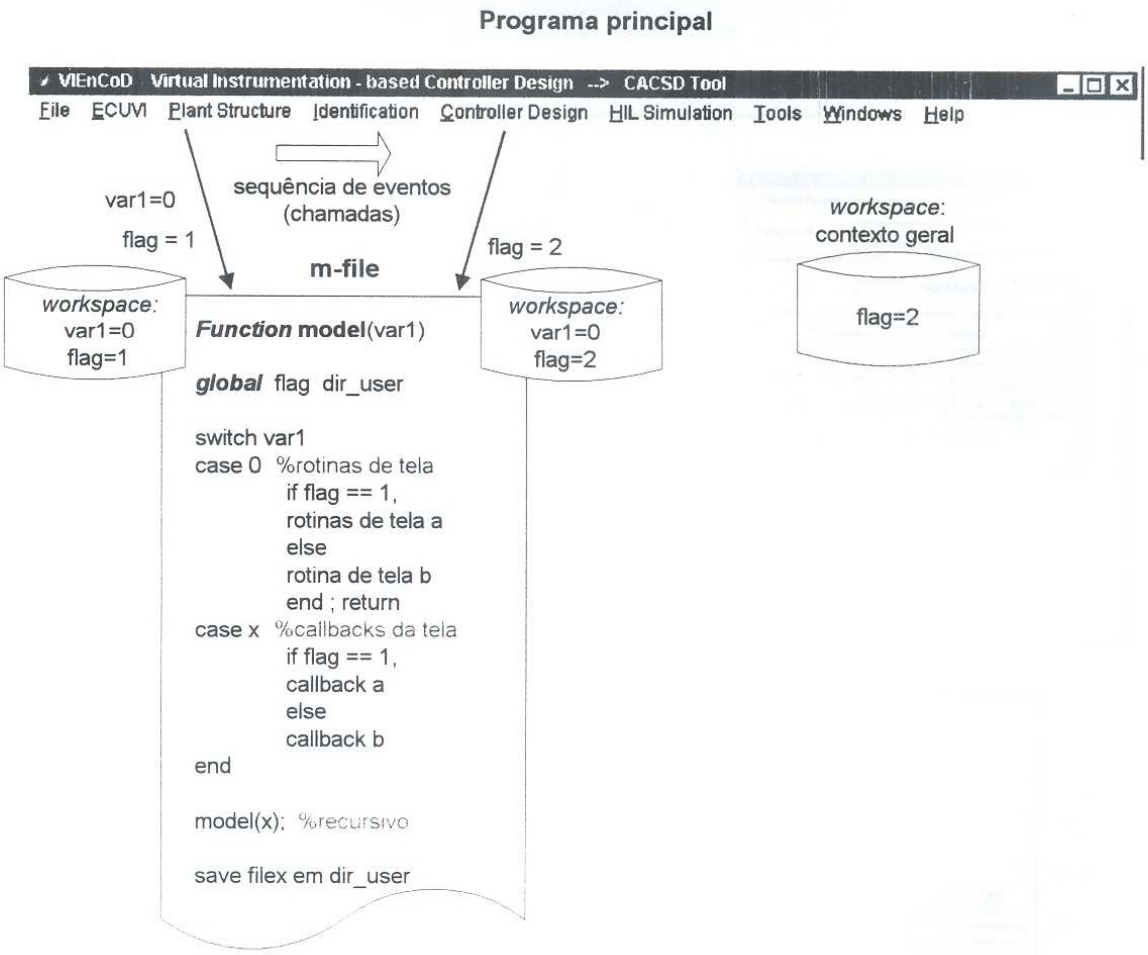


Figura 25 : Contexto de *workspace* e fluxo de dados

A passagem de variáveis através do contexto de chamada são utilizadas apenas para determinação da execução de rotinas de montagem de GUI ou execução de *callbacks*

desta GUI, utilizando-se apenas um *m-file* (esta técnica será descrita mais adiante). A figura 25 ilustra este processo implementado no VIEnCoD.

A **recursividade** é uma das ferramentas mais importantes na determinação da estrutura de *m-files*. Para aplicações isoladas, normalmente existe a divisão do programa em duas partes distintas (*m-files*) : uma com as estruturas de configuração dos elementos que compõem uma GUI (será visto mais adiante) e outra com as linhas de comando que executarão uma tarefa definida pelo elemento escolhido na GUI (***callback***). Quando a quantidade de elementos é pequena cada *callback* é encapsulada por uma *m-file*.

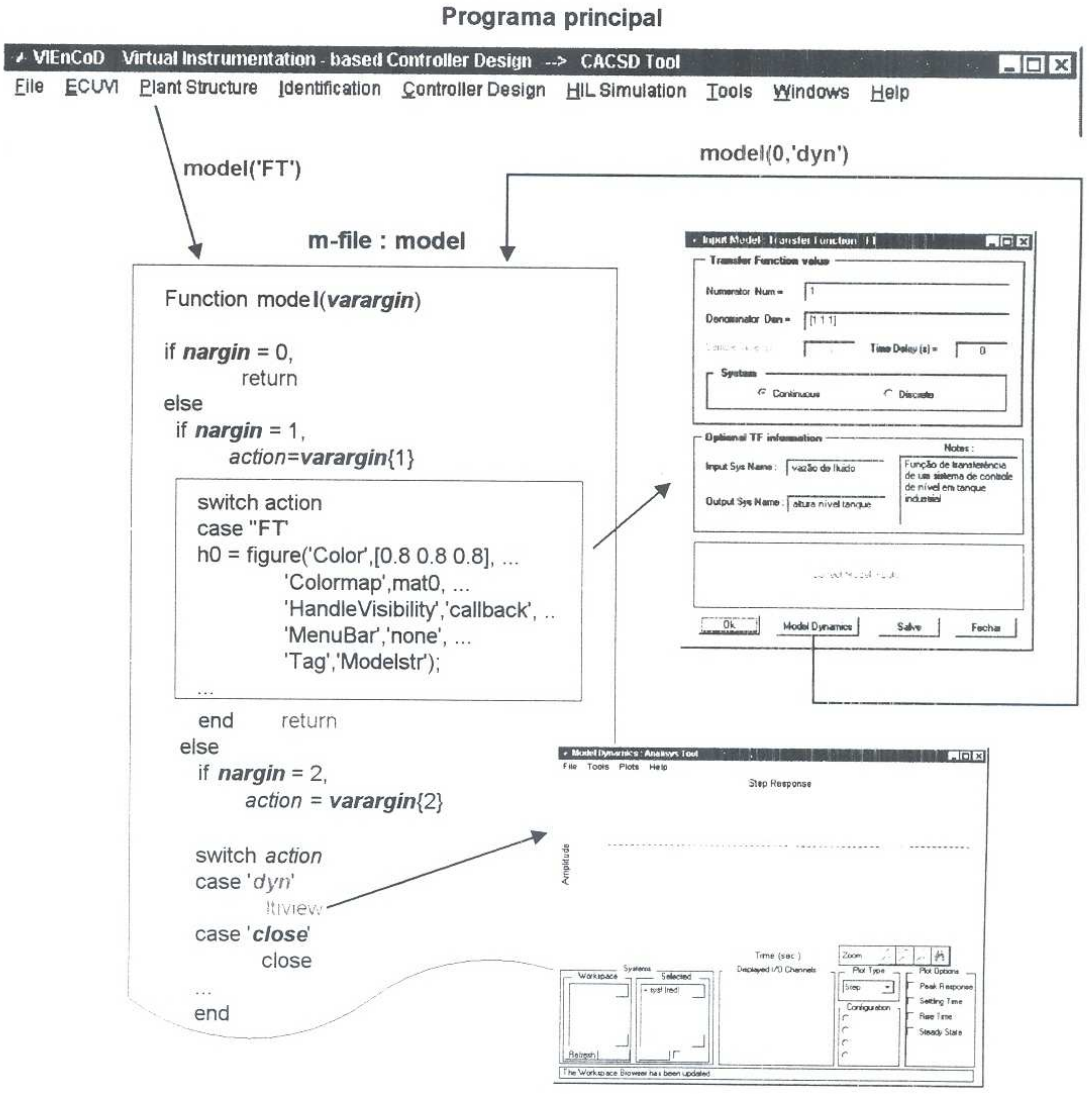


Figura 26 : Estrutura unificada de *m-file* com técnica recursiva

A estrutura do VIEnCoD é formada por uma quantidade razoável de GUIs que realizam tarefas inerentes a cada fase do CDSC. De maneira a tornar a estrutura de *m-files* simplificada e manter a característica modular, foi adotado a seguinte concepção : os comandos de configuração dos elementos gráficos de uma GUI e os comandos de execução das *callbacks* convivem no mesmo *m-file*. Para tal, é necessário a adoção de um procedimento que mantenha a legibilidade do programa diferenciando de forma clara suas estruturas. O uso da técnica recursiva apoiada pelas funções ***varargin*** e ***nargin*** possibilitam este objetivo. A figura 26 ilustra este processo.

Com o *varargin* permite-se o envio de qualquer número de parâmetros na estrutura de chamada de um *function*. O *nargin* determina o número de parâmetros enviados. A diferenciação para execução das linhas de configuração de GUI e execução de diferentes *callbacks* reside na aplicação destas funções juntamente com estruturas condicionais (no caso "*case*"), sob o seguinte padrão :

1. A não passagem de parâmetros não executa a *function* (nos casos onde o layout da GUI varia em função de uma escolha que é definida por parâmetro*)*: *nargin = 0, return*. Nos casos onde o aspecto da GUI é constante, os comandos de configuração são executados: *nargin = 0, ...comandos de config...., return.*
2. O envio de um parâmetro define a execução da GUI bem como a definição do seu layout (pelo parâmetro enviado): *nargin = 1, action = parâmetro1, ...comandos de config...., case action -> configs específicos, return.*
3. Um vez que a GUI está disponível para preenchimento de campos, procede-se a execução de uma determinada tarefa determinado por um botão, por exemplo. Esta callback chamará o mesmo programa (recursividade) enviando dois parâmetros : *nargin = 2, action = parâmetro2, case action -> callbacks específicos.* Desta forma, a estrutura condicional inicial *if nargin = 2* , faz saltar as linhas de configuração de GUI e executando apenas um *callback* especificado.
4. Qualquer outra chamada recursiva atua através do segundo parâmetro, cuja *callback* é selecionada através das estruturas condicionais.

Dentro da filosofia de utilização dos recursos Matlab adotada enquadra-se também a utilização de algumas interfaces gráficas (GUIs). Entretanto, de forma a permitir modificações e adaptações das mesmas para a adequada integração à estrutura VIEnCoD,

é possível que o(s) *m-file*(s) correlatos sejam definidos com ***private functions,*** mantendo os nomes originais. Para tal é necessário que estes *m-files* modificados residam em um subdiretório nomeado "***private***" . Assim, a "visibilidade" destes será apenas para *m-files* do subdiretório "pai", e com prioridade na chamada durante a pesquisa de *path* que o sistema operacional realiza. O aplicativo *ltiview.m* (para análise da dinâmica de modelos), por exemplo, localizado no *path* do *Control Toolbox,* não será mais utilizado se possuir uma versão modificada nas condições citadas acima. Esta técnica é útil quando da instalação do *toolbox* VIEnCoD em diferentes máquinas, onde se mantém as estruturas originais Matlab instaladas.

Apesar da forte estrutura de apoio à utilização das GUIs através de *hints* e *helps,* conforme será visto logo adiante, foi implementado uma estrutura de tratamento de erros em diversas situações, por exemplo, para entrada de sintaxe correta nos campos das GUIs. A estrutura que permite isto é: ***eval('comando','comando_secundário)***. Se a execução do '*comando*' incorre em erro, o controle do programa é direcionado à execução do 'comando_secundário'. Este pode ser um ***function*** **local** (*function* dentro da *m-file* para execução de rotinas internas) contendo rotinas para avaliação do erro ocorrido (através da função ***lasterr*** que indica qual o tipo do erro ocorrido) e rotinas para tratamento condicional do programa.

- ***Programação visual***

O Matlab através da interface GUIDE permite uma fácil criação de GUIs com geração automática de código. O processo de montagem do *layout* de uma nova interface baseia-se apenas na manipulação de elementos escolhidos : menus *pull-down* e *pop-up*, *push buttons*, *menus*, *radio buttons*, *check boxes*, *list boxes*, *sliders,* elementos gráficos (2D e 3D) e caixas para edição e diálogo. A especificação das propriedades de cada elemento bem como a definição da execução de tarefas específicas (*callbacks*) podem ser feitas através de ferramentas disponíveis nesta interface : editor de propriedades, editor de *callback*, editor de menus e ferramenta de alinhamento.

O projeto de uma GUI é uma atividade de grande importância na estruturação de todo o ambiente de *software*, em específico para caracterização como ferramenta

CACSD. A proposta multifuncional do VIEnCoD, intensamente abordada nos capítulos anteriores, exigiu que as interfaces sintetizassem toda a estrutura teórica de controle necessária àquela fase do CDSC, disponibilizando recursos de forma clara e intuitiva e com fácil percepção de objetivos. Para tal, adotou-se a metodologia sugerida em [The Mathworks, 1997a] e referenciada em alguns trabalhos como [ÅSTRÖM et al., 1998] [WITTENMARK et al., 1998], com as necessárias adaptações aos objetivos propostos. Uma abordagem semelhante é encontrada em [SILVEIRA, 1998] onde também se encontram contribuições. Nesta, a estrutura global do ambiente deve conter :

- *Familiaridade* : mesmo perfil de *layout* entre as interfaces;
- *Concentração* : uma idéia por módulo. A idéia corresponde a determina etapa do CDSC (ou uma tarefa desta etapa);
- *Simplicidade* : a interface deve apresentar-se de forma objetiva e com senso de unidade. As funcionalidades só devem ser adicionadas à interface se realmente necessárias ao conjunto.

Desde o levantamento das necessidades à fase final de utilização de uma GUI é necessário a adoção de uma técnica no processo de projeto. Tal processo é ilustrado na figura 27.

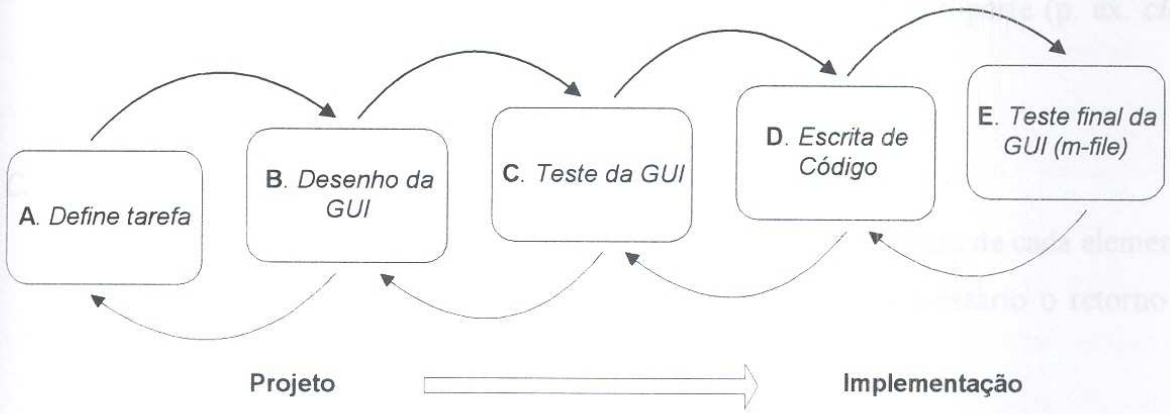


Figura 27 : Processo de projeto de uma GUI

Cada etapa representa a realização de tarefas definidas e sempre sofrendo realimentação na implementação:

A. Define tarefa

1. Etapa do CDSC a ser abordada : definição de um item do menu da tela principal. Ex.: Modelagem (*Plant Structure*);

2. Identificação e subdivisão de tarefas que contemplem toda a etapa: subdivisão em submenus. Ex.: *Plant Strucuture* → *Create Model*;
3. Levantamento de recursos para realização das tarefas. Ex.: estruturas *LTI* do *Control Toolbox*;
4. Avaliação das GUIs necessárias para suporte de cada tarefa: normalmente cada tarefa (submenu) é suportada por uma GUI; podendo também ocorrer mais de um submenu ser suportado por uma única GUI. Neste caso, existem estruturas condicionais no programa da *m-file* que determinam pequenas mudanças no layout da mesma GUI. Ex.: *Plant Strucuture* → *Create Model* → *TF(s), SS* ou *ZPK.;*

B. Desenho da GUI

5. Com base no conteúdo de controle necessário ao suporte da tarefa e no conhecimento das estruturas e sintaxes das funções envolvidas define-se os campos de entrada (p. ex.: *editbox, listbox*), comandos (p. ex. *pushbottons*), auxílio (*helps* e *hints*), aviso (caixas de aviso – p. ex. *msgbox*) e saída (p. ex. *caixas de texto, gráficos*);
6. Montagem dos elementos através do GUIDE;
7. Geração do *m-file* base : linhas de código da configuração dos elementos são obtidas automaticamente e definição de *callbacks* de pequeno porte (p. ex. *close* para o *pushbotton "Exit"*);

C. Teste da GUI

8. Executa-se a GUI (*m-file*) de base e verifica-se o funcionamento de cada elemento. É neste momento que podem surgir novas idéias sendo necessário o retorno ao item 5;

D. Escrita de código

9. O programa base é modificado de forma a comportar a estrutura de gerenciamento da GUI e dos *callbacks* (de maior porte) . Tal modificação é feita seguindo a concepção adotada em acumular em uma mesma *m-file,* as estruturas de configuração de tela e *callbacks*. As rotinas de cada *callback* são inseridas após a chamada de cada *"case"*. Vide figura 26, já abordada. Assim, é muito fácil a depuração, modificação ou adição de novas rotinas ou módulos;

E. Teste final da GUI

10. Faz-se todo teste da interface verificando o atingimento dos objetivos propostos.

Os elementos gráficos são configurados através de parâmetros que definem suas características visuais e operacionais. O conhecimento de cada parâmetro foi fundamental para implementação de diversos recursos funcionais aplicados às GUIs. Conforme já citado, existe deficiente bibliografia correlata sendo que o conhecimento obtido foi o resultado de estudo empírico baseado em estruturas já existentes. Portanto, vale no presente trabalho a colocação deste conhecimento adquirido de forma a constituir uma fonte de referência para futuros trabalhos. Durante o desenvolvimento do tópico "implementação de recursos" logo a seguir, alguns destes parâmetros e funções são citados de forma a explicitar a técnica de programação de GUIs através do controle dos mesmos. Primeiramente será visto um pouco das técnicas de programação com o ambiente Simulink.

- ***Técnicas de programação com Simulink***

O Simulink é um ambiente de programação gráfica destinado à modelagem, análise e simulação de sistemas dinâmicos, onde funções são encapsuladas em blocos com características de entrada e saída definidas. É particularmente útil para representação de sistemas complexos, onde toda estrutura é formada pelo acoplamento de subsistemas, cada qual possuindo sua representação e características próprias de entrada e saída.

Para representação e simulação de malhas de controle torna-se uma ferramenta mais amigável e eficaz do que a manipulação com funções Matalab. É baseado neste ponto que alguns aplicativos foram desenvolvidos para operarem com diagramas Simulink. Especificamente, para otimização de sistemas de controle operando em malha fechada, tem-se o NCD *Blockset* e também o *Optimization Toolbox*; ferramentas chave para suporte da etapa de otimização do controlador do CDSC proposto pelo VIEnCoD.

Neste sentido, o ambiente permite a participação de elementos Simulink de forma a trabalhar com os recursos e algoritmos Matlab de forma transparente. Por exemplo, a modelagem da planta, controlador e demais elementos de uma malha podem ser feitos

através da manipulação de blocos (configuração de parâmetros). Também no Simulink é possível a parametrização de modelos baseado na classe LTI, citado anteriormente.

O processo de integração dos blocos e diagramas Simulink ao recursos Matlab pode ocorrer de diferentes maneiras. O processo mais básico é que todo o parâmetro de configuração de qualquer bloco pode ser referenciado a uma variável presente no *workspace*. Tal variável pode ser um vetor de dados de excitação produzidos por um algoritmo desenvolvido em uma *m-file*, uma função de transferência obtida na fase de identificação,... etc. Assim, é possível a adição de estruturas de programas que envolvam arquivos *mdl* (arquivos Simulink). Existem também outras formas, tais como:

- Blocos "*From File*" e "*From Workspace*" : estes blocos são encontrados na biblioteca *Simulink Sources* e destinam-se a interfacear (leitura de) vetores em arquivo (*.mat) ou no *workspace* e disponibilizar na saída do bloco simulando um sinal de excitação;
- Do processo inverso ao anterior, os blocos "*To File*" e "*To Workspace*", escrevendo dados em arquivo (*.mat) ou no *workspace*;
- Blocos "*Fcn*" e "*Matlab Fcn*" : permitem o acesso a qualquer *function* do Matlab para geração de vetores (sinal) de comprimento configurado;
- *S-Function* : quando deseja-se a descrição do comportamento dinâmico de um sistema através de um algoritmo, o programa que o engloba é encapsulado em um bloco chamado *S-Function*. O programa pode ser feito em Matlab (*m-file*) ou código C (tratado como *mex-file* – C próprio para utilização no Matlab). Servindo para criação de blocos customizados Simulink apresenta algumas aplicações : incorporação de código C existente em uma simulação, descrição de um sistema através de conjunto de equações matemáticas, dentre outras.

O contexto de integração ao ambiente Matlab que estes recursos fornecem, necessitam que estruturas Simulink sejam previamente manipuladas para definição de variáveis, blocos, sistemas e subsistemas. A ligação ocorre por referência no programa (*m-file*) a cada elemento destas estruturas armazenadas em arquivo (*.mdl) que definem um sistema, por exemplo, um sistema em malha fechada com blocos controlador, planta, sensor e filtro conforme ilustra a figura 28. Quando se deseja total transparência às interfaces Simulink e manipulação de seus elementos para criação de um diagrama,

existem comandos Matlab que permitem este gerenciamento (criação, manipulação e configuração). A reunião deste comandos em um *m-file* permite a geração e manipulação automática de estruturas de forma programável. Este recurso foi implementado na configuração da malha de simulação da figura 28, através da criação e substituição de elementos escolhidos (comandos *add_block* e *replace_block*). Assim, evita-se que diversas estruturas diferentes sejam armazenadas (vários arquivos *.mdl) atendendo a casos específicos. A *m-file* responsável pela configuração da malha (através de uma GUI) utilizando esta técnica, é chamada pelo bloco *config*. Um bloco pode-se tornar um elemento executável pela declaração no campo "*Open function*" de suas propriedades. Assim, o usuário não precisa ter nenhum conhecimento operacional no Simulink.

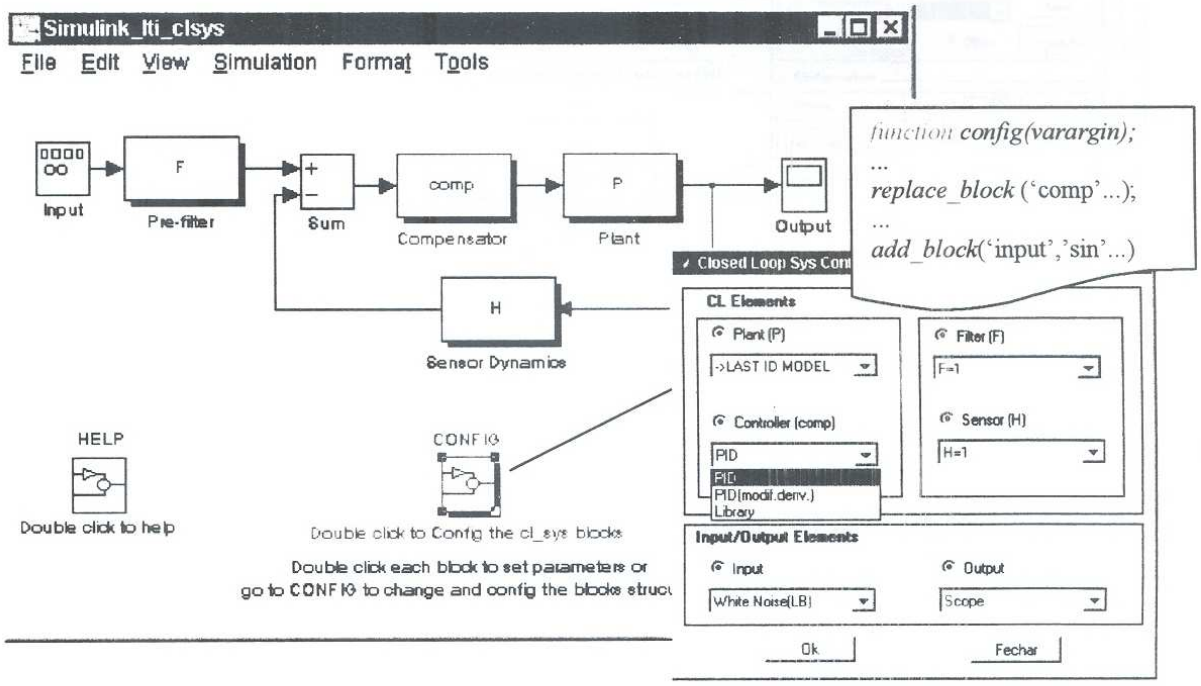


Figura 28 : Técnica de configuração de um diagrama por *m-file*

♦ ***Implementação de recursos***

Neste tópico serão exploradas algumas das principais implementações através da aplicação de diversos recursos disponíveis no Matlab e Simulink. Conforme já mencionado, uma maior complexidade ou melhora em determinados aspectos do ambiente são em detrimento do tempo. O objetivo é expor um pouco das potencialidades de programação através da validação de idéias, determinando um ambiente com base metodológica estruturada no CDSC.

A figura 29 representa uma interface desenvolvida para dar suporte ao sistema de análise e geração de sinais de teste e de excitação durante a etapa de identificação. Esta será utilizada como referência às implementações descritas.

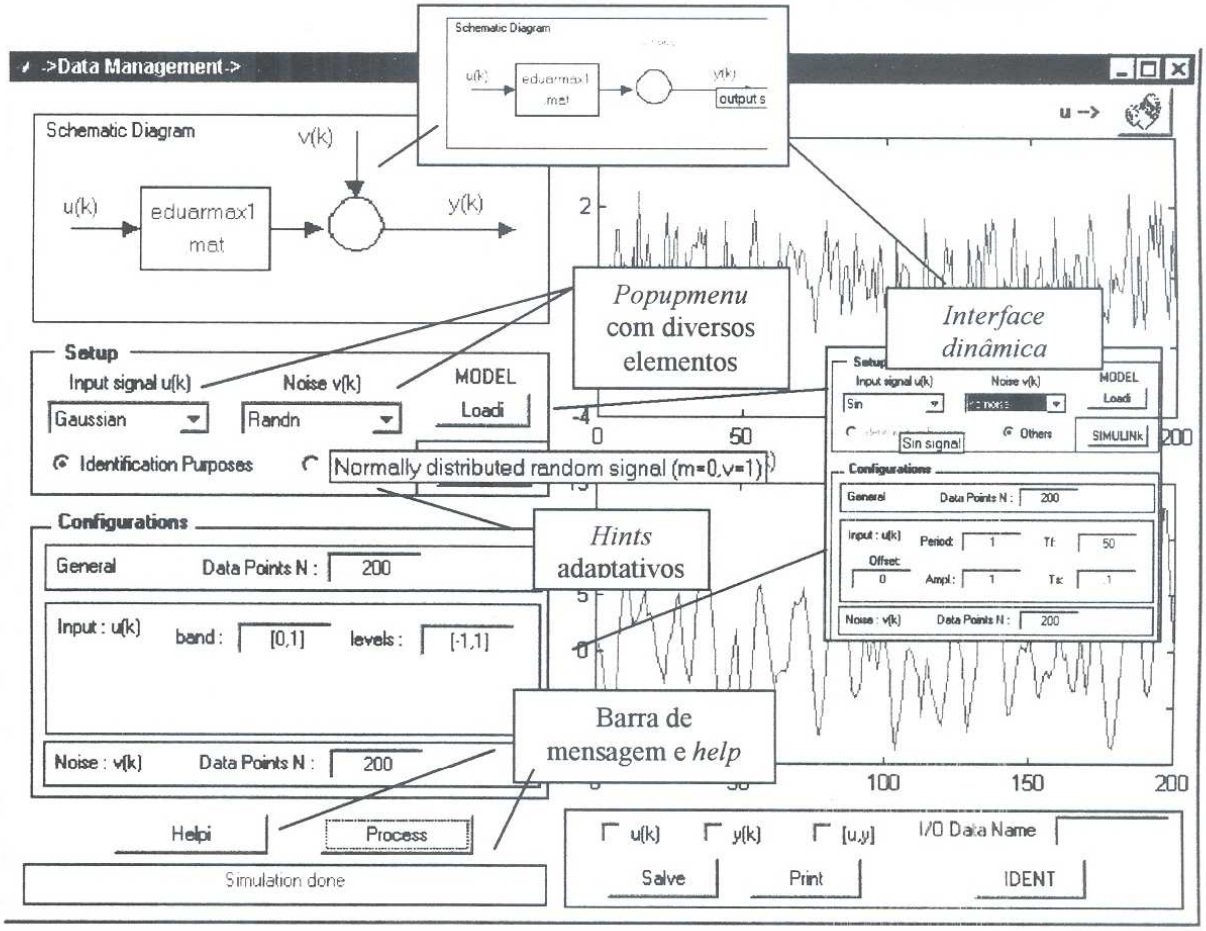


Figura 29 : Exemplo de aplicação de recursos

Os vários processos que compõe uma ambiente CACSD são bastante interativos e muito dependentes de uma cultura e conhecimento a priori. Existem muitos campos para entrada de parâmetros que não devem apresentar dificuldades no entendimento de sua sintaxe. Para tal, foi desenvolvido um suporte baseado em *helps* e *hints* adaptativos. O *hint* é uma estrutura que fornece uma informação sobre um elemento sobre a qual o cursor se encontra. Quando é relacionado com um *popupmenu* onde seus vários elementos necessitam de um rápido descritivo para auxílio na escolha, se faz necessário que o *hint* se adapte ao elemento selecionado mudando o conteúdo de sua informação e não apenas apresente uma informação genérica do *popupmenu*. Na figura 29 esta implementação é aplicada a identificação do campo "*input signal u(k)*". A maioria do elementos componentes de uma GUI apresentam *hints* (adaptativos ou não). Todas as GUIS

apresentam *helps* com informações que geralmente são sobre o conteúdo teórico correlato ou orientação de sintaxe.

Além dos *hints* e *helps* vale destacar um outro recurso utilizado para o suporte informativo das GUIs: as *barras de mensagem*. Atuam em complemento aos *hints* memorizando a última seleção do cursor e apresentam mensagens de aviso quando uma ação não é sucedida (p. ex. salvar um vetor de sinal sem entrar o nome para arquivo).

O elevado número de tópicos e itens de configuração para execução de uma tarefa aumenta a complexidade no projeto de uma GUI, exigindo-se a adoção de técnicas que mantenham as características desejáveis a interface: *familiaridade, concentração e simplicidade*. A *interface dinâmica* é um exemplo destas técnicas. A idéia se baseia na alteração dinâmica setorial do layout de uma GUI à execução de comandos através de seus elementos. O *popupmenu* "*input signal u(k)*" da interface da figura 29, altera dinamicamente seus campos na área "*configurations*" em função das características de cada sinal que necessitam diferenciação no tipo e quantidade de parâmetros. Também a figura "*Schematic Diagram*" sofre alterações mudando dinamicamente suas informações. Todo o processo de controle da GUI, incluindo o suporte da técnica de *interface dinâmica*, se dá através de uma única *m-file* com estrutura de programa recursivo especial abordado anteriormente. Os itens de configuração de elementos gráficos (p. ex. *editbox*) "*Visible*" e "*Enable*" são acessados para tornar o elemento visível e acessível respectivamente, conforme a situação. No caso da figura "*Schematic Diagram*" a atuação ocorre no item de configuração "*String*".

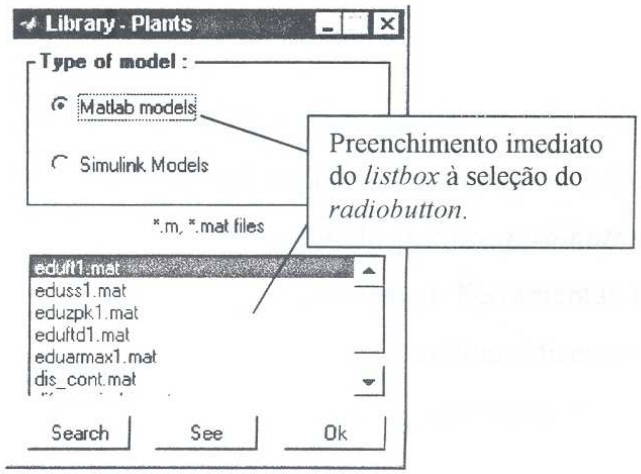


Figura 30 : Comando dinâmico

A extensão da técnica descrita é aplicada também para execução automática de tarefas (*callbacks*) através de elementos de seleção. A figura 30 mostra uma interface destinada a exibir a bibliotecas de modelos desenvolvidos em um projeto. Quando um *radiobutton* é selecionado o *listbox* é imediatamente preenchido com os modelos referentes à escolha, sem a necessidade da atuação de um comando secundário.

O ambiente permite também a disponibilização de barras de ferramentas que são agrupadas por grupos (segundo divisão dos menus) de forma a fornecer acesso mais amigável (através de *pushbottons* com figuras) e rápido (sem a navegação nos menus). A lógica de habilitação é a mesma aplicada aos menus e submenus. A figura 31 mostra a barra de ferramentas para tarefas da ECUVI.

Figura 31 : Barra de ferramentas

4.2.1.2 Estrutura de *software* do *Toolbox* VIEnCoD

Todo o *toolbox* VIEnCoD é gerenciado através de uma GUI principal que assume o papel de concentrar todas a etapas do CDSC de forma clara e intuitiva. Isto é feito através de menus dedicados e submenus com as tarefas específicas a cada fase e também com a técnica da *árvore dinâmica*.

A *árvore dinâmica* é a adaptação da técnica sugerida em [MUNTEANU et al., 1997] onde todo o conteúdo teórico de controle suportado pelo *software* é colocado sob a forma de um diagrama onde cada tópico é representado por um *pushbotton* que executa GUIs específicas ou telas com conteúdo teórico (hipertexto). Ferramentas de modelagem e análise (resposta ao degrau, diagrama de Bode, sistemas contínuo/discreto,... etc) podem ser acessadas de botões estruturalmente dipostos no *layout* que fornece uma visão global do sistema.

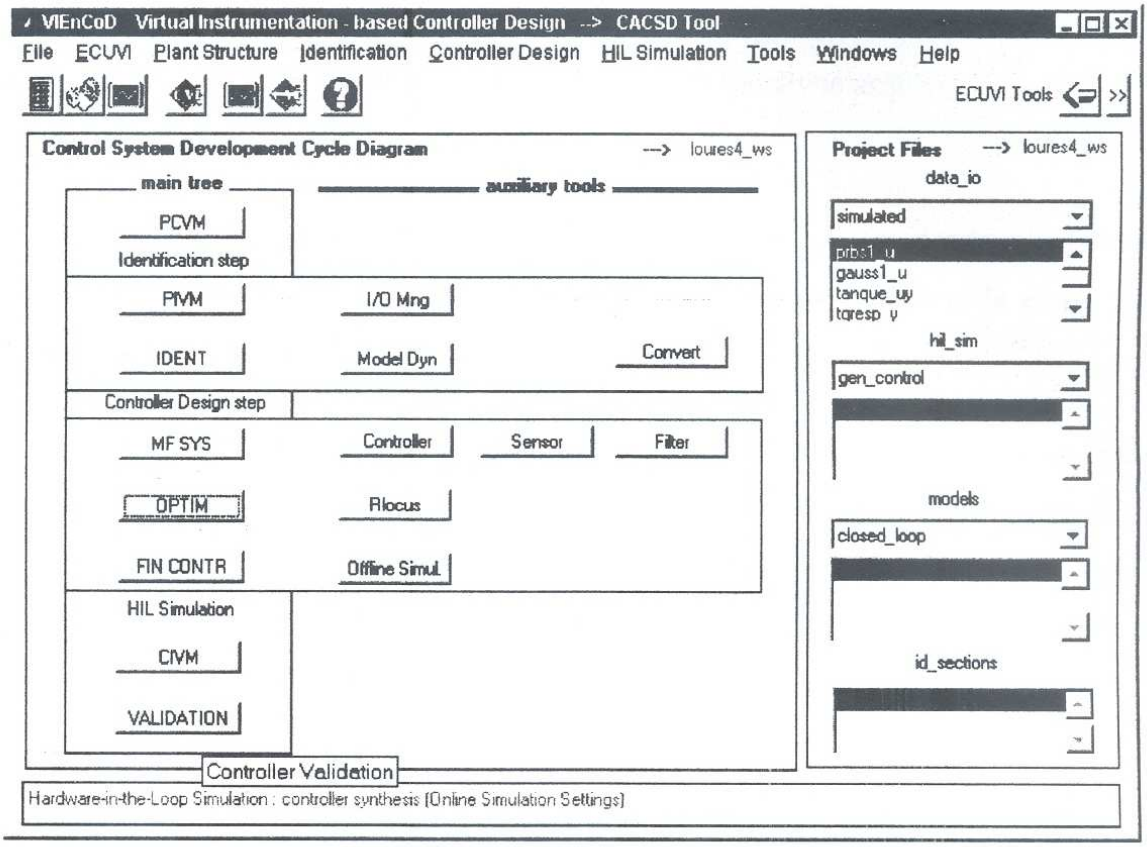


Figura 32 : Tela principal e a *árvore dinâmica*.

Em um ambiente com proposta CACSD a árvore dinâmica é um recurso interessante para acompanhamento do projeto do CDSC, visualizando etapas e ferramentas disponíveis. A figura 32 mostra a tela principal do VIEnCoD com implementação desta técnica. A árvore concentra os principais comandos de cada etapa do CDSC através de *pushbottons*, que são dispostos no *layout* que expressa de forma simplificada o diagrama estrutural do ambiente desenvolvido na figura 34 e 35 adiante. Desta forma, uma visão global do projeto é fornecida ao mesmo tempo que o usuário recebe informação sobre sua estrutura lógica e seqüencial através da habilitação dinâmica dos *pushbottons* no cumprimento de etapas.

Quando o ambiente é fechado a atual situação é memorizada através de *flags* permitindo a continuação do projeto em momento posterior sem perder as informações até então adquiridas (arquivos armazenados, estrutura da árvore mantida).

Em auxílio ao processo de acompanhamento e visão global do projeto, *browsers* mostram todos os diretórios da pasta do usuário com os respectivos arquivos (aquisições, modelos, seções de identificação, controladores,... etc).

A estrutura de *m-files* e arquivos que compõe o VIEnCoD foi desenvolvida de forma modular com caracterização funcional baseada no atendimento de cada etapa do CDSC, bem como as necessidades operacionais do ambiente.

O programa principal (*main_vien.m*) é estruturado de tal forma a expressar esta modularidade funcional onde todos os recursos desenvolvidos (*m-files*) são distribuídos em linhas de programa delimitada (classificada) em grupos (menus e submenus). Desta forma, a absorção de novas contribuições recai apenas na declaração dentro do programa principal da estrutura de novos menus associados aos *m-files* criados.

Quando da utilização do ambiente (execução do *main_vien.m*) é solicitado a criação de um novo projeto ou abertura de um existente, envolvendo-se os seguintes procedimentos com os respectivos objetivos:

1. São criadas todas as pastas de suporte onde são armazenados todos os arquivos gerados durante o CDSC. A figura 33 juntamente com a tabela 2 discriminam o padrão de pastas e a classificação quanto aos tipos de arquivos armazenados. A raiz desta estrutura é definida pelo usuário;

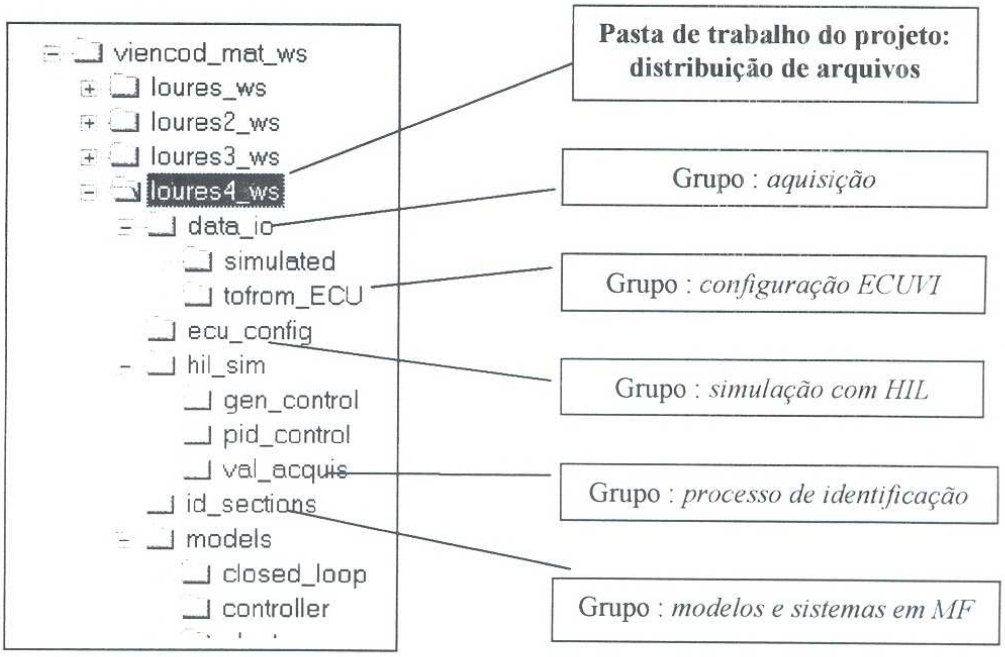


Figura 33 : Padrão de pastas de um projeto

Pasta de trabalho do usuário (projeto) : ***nome_ws***		
Pastas	**Tipo de arquivos**	**Sintaxe arquivos**
❑ ***data_io***	Grupo para arquivos de aquisição e simulação de sinais - módulo PIVM	
❑ ***simulated***	Vetores de sinais produzidos pela estrutura da GUI "*I/O Management*"	*nome_**u**.mat* :excitação *u(k)* *nome_**y**.mat* :saída sistema *y(k)* *nome_**uy**.mat* :ambos *[u(k) y(k)]*
❑ ***tofrom_ECU***	Todos os vetores de sinais gerados ou enviados à ECUVI. São arquivos em ascii.	*nome_**utecu.dat*** :sinal gerado e enviado para ECU *u(k)*, p.ex. PRBS *nome_**uyfecu.dat*** :aquisição feita pela ECU – para identificação *[u(k) y(k) Ts]*
❑ ***models***	Grupo para arquivos de aquisição ou simulação de sinais	
❑ ***closed_loop***	Sistema em malha fechada configurado para utilização com ferramentas de otimização: *single loop* (planta, controle, sensor, filtro); outro configurado pelo usuário	*nome_**clsys**.mdl* : diagrama Simulink
❑ ***controller***	Modelo do controlador	*nome_**ctr**.mat*: representação matemática (paramétrica ou classe lti) *nome_**ctr**.mdl*: diagrama Simulink (bloco)
❑ ***plant***	Modelo da planta	*nome_**id**.mat* : modelo paramétrico obtido na etapa de identificação. *nome.mat*: representação matemática (paramétrica ou classe lti) *nome.mdl*: diagrama simulink (bloco)
❑ ***sensor_filter***	Modelo do sensor e filtro	*nome_**sens**.mat, nome_**filt**.mat*: representação matemática (paramétrica ou classe lti) *nome_**sens**.mdl, nome_**filt**.mdl*: diagrama simulink (bloco)
❑ ***id_sections***	Processo de identificação realizado com a ferramenta *ident.m*	*nome.**sid***
❑ ***ecu_config***	Grupo para arquivos de configuração de experimentos (planta e ECUVI) : biblioteca de experimentos. Módulo PCVM	*nome.**vi***
❑ ***hil_sim***	Grupo para arquivos de troca com ECUVI para simulação com HIL - Módulo CIVM	
❑ ***gen_control***	Parâmetros controlador genérico *[a1...na b1...bn Ts]*	nome_***ctrgen.dat*** : obtido no CDSC nome_***ctrgen_m.dat*** : entrada manual pela GUI
❑ ***pid_control***	Parâmetros PID *[Kp, ti, td Ts]*	*nome_**pid.dat*** : obtido no CDSC *nome_**pid_m.dat*** : entrada manual pela GUI
❑ ***val_acquis***	Vetor de aquisição *[u,y Ts]* e gráfico de validação (sistema real x simulado)	*nome_**hilacq.dat*** : vetores de aquisição *nome_**graf_val.m*** : gráfico

Tabela 2 : Padrão de pastas e arquivos de um projeto

2. Um arquivo *.mat é criado funcionando como identificador do projeto. Sua estrutura é formada por *flags* e vetores dinamicamente alterados que armazenam a condição atual de desenvolvimento do projeto, por exemplo : identificação da etapa sob desenvolvimento (p. ex. lógica de habilitação de submenus e *pushbottons*), arquivos temporários (modelo identificado, aquisição,... etc) sob trabalho quando da saída do ambiente,... etc;
3. Definição de *path* para inserção dos diretórios do usuário na pesquisa automática feita pelo sistema operacional do Matlab.

A estrutura de *software* do VIEnCoD foi projetada de maneira a dar suporte ao CDSC cuja metodologia foi abordada no capítulo 3. As figura 34 e figura 35 apresentam esta estrutura onde o fluxo principal de dados obedece exatamente a metodologia proposta. Nestas figuras são exibidos todos os blocos funcionais do ambiente (aplicativos de cada etapa ou recursos operacionais) divididos por grupos (etapas do CDSC ou recursos operacionais). Vale ressaltar que os dados presentes no fluxo principal são representados por arquivos temporários espelho dos arquivos já armazenados na pasta do usuário. Desta forma, ao fechamento do VIEnCoD as informações processadas são armazenadas juntamente com o *status* do CDSC (qual fase se encontra em em que tarefa) permitindo o posterior retorno com transparência na recuperação dos dados.

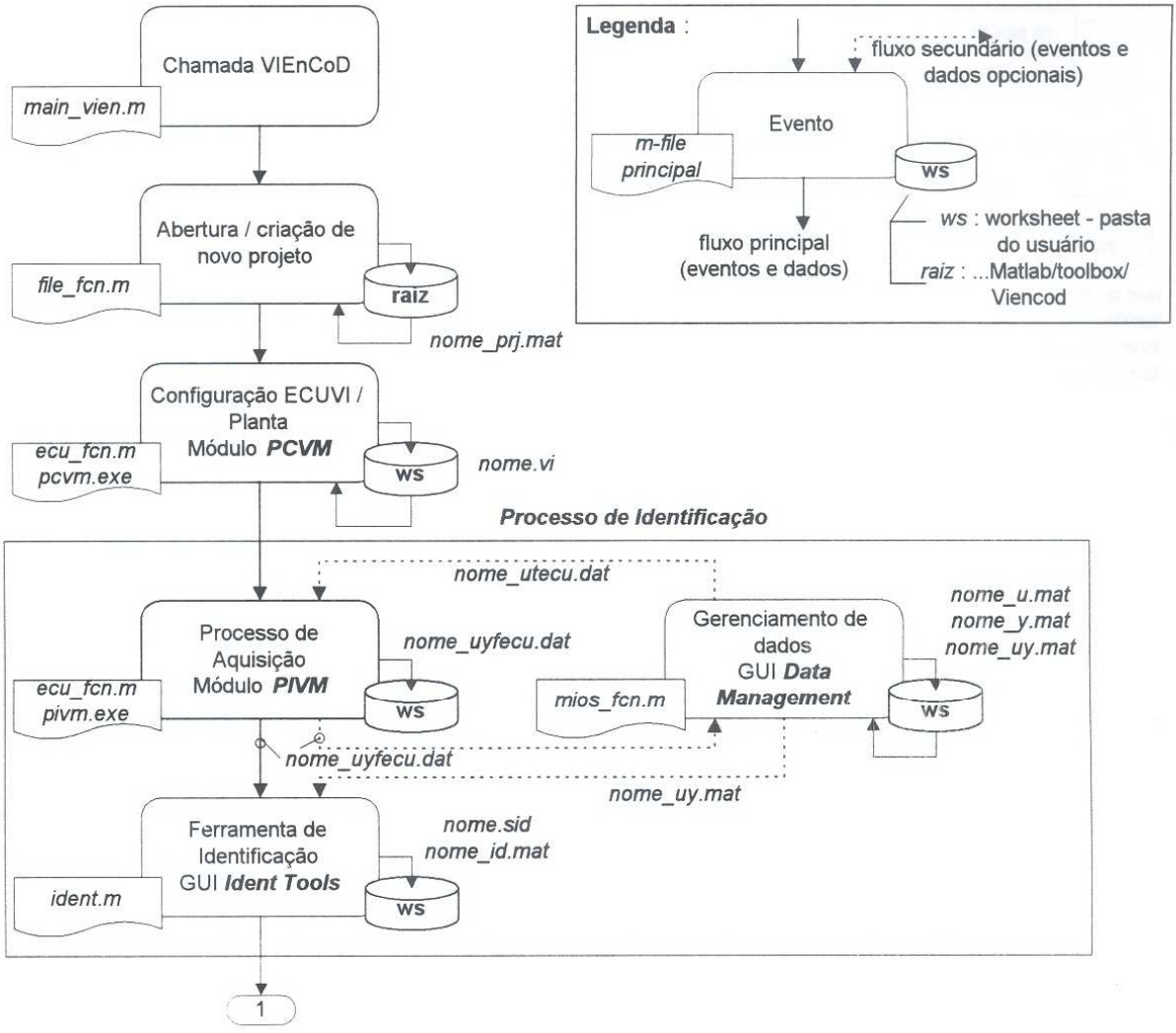


Figura 34 : Diagrama estrutural do VIEnCoD

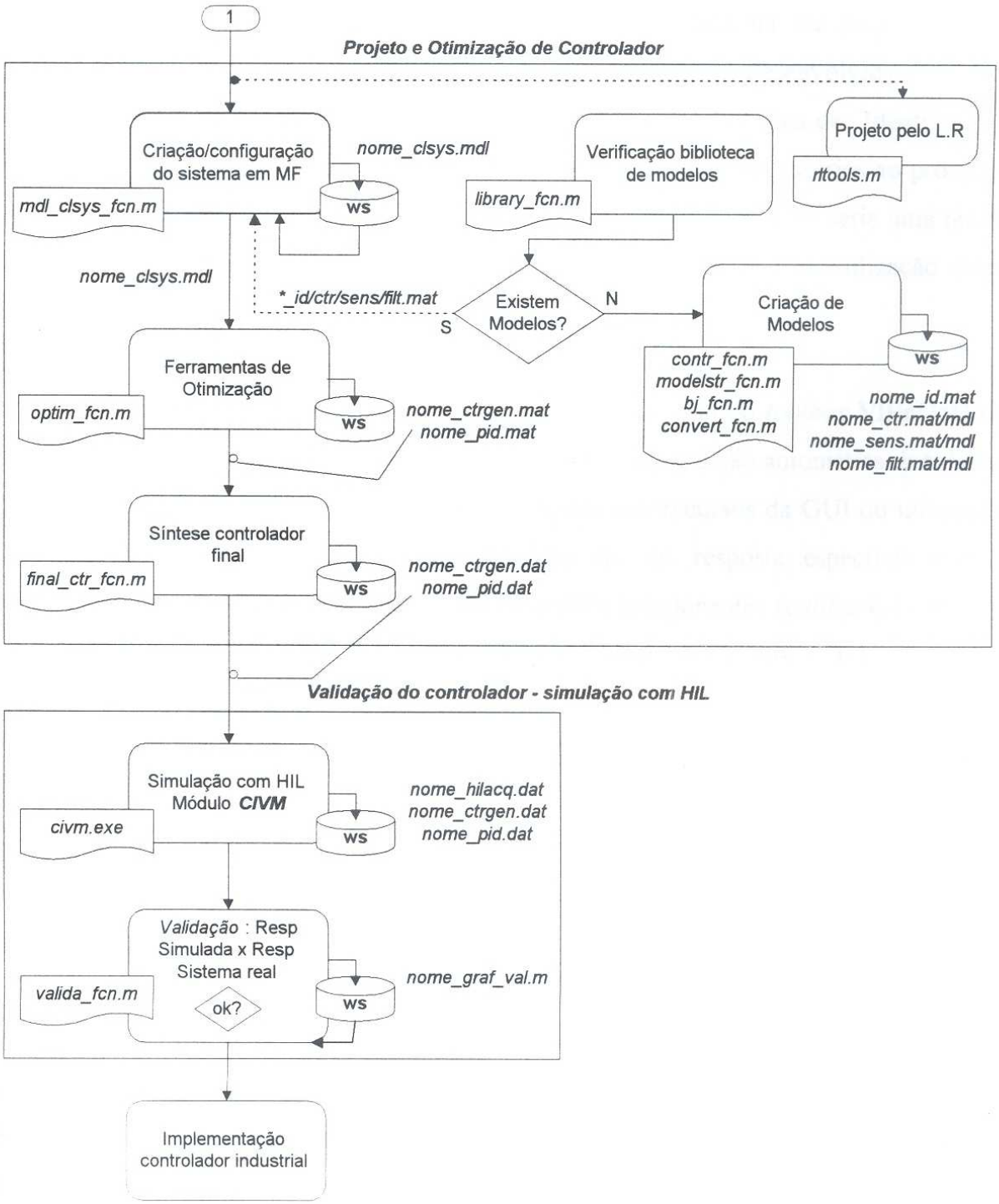


Figura 35 : Diagrama estrutural do VIEnCoD

4.2.1.3 Comentários adicionais

A etapa de identificação é fundamentalmente embasada na utilização da GUI gerenciadora dos recursos do *Identification Toolbox,* biblioteca desenvolvida pela *The Mathworks* em parceria com Dr.Ljung, um dos grandes nomes na área de Identificação de Sistemas. Tal interface agrega todos os recursos e ferramentas necessárias ao processo de identificação descrito no capítulo 3. O desenvolvimento de outra GUI seria uma tarefa redundante com resultados iguais ou inferiores. Portanto, optou-se pela utilização desta ferramenta.

Entretanto, para integração à estrutura lógica e sequencial do *toolbox* VIEnCoD tal interface necessitava algumas adaptações, por exemplo, na inserção automática dos dados experimentais (vetor de excitação e saída do sistema) nos recursos da GUI ou utilização individualizada de algumas de suas ferramentas (p. ex. resposta espectral) sem o desligamento com a estrutura global. Todos os *m-files* componentes (*callbacks*) relativos a cada elemento da GUI foram analisados (variáveis globais, estruturas,... etc).

Uma sequência de tarefas visando a identificação de um modelo foi simulada onde a troca de parâmetros e arquivos foi varrida através da depuração de todos os *m-files* encadeados envolvidos. A conclusão final obtida é que toda a estrutura de utilização de *functions* é gerenciada pela GUI com controle sobre os eventos efetuados através de seus elementos não permitindo o acesso de uma ferramenta isolada. O evento de utilização de cada elemento da GUI pela atuação direta (*mouse* ou teclado) ativa *flags* e uma lógica entrelaçada de estruturas condicionais que habilitam a operação adequada de cada *function*. Estes *flags* controlam o funcionamento de uma complexa "máquina de estados". Em determinada fase do estudo tinha-se oito *m-files* em depuração e um mapa de passagem de variáveis de aproximadamente 25 elementos. Sem a obtenção de êxito, decidiu-se pela busca de soluções substitutivas.

A solução adotada foi a utilização de interfaces criadas com o auxílio nos procedimentos iniciais de utilização da GUI, como o caso da inserção dos dados experimentais feitos pela ECUVI. O Matlab contém comandos próprios (*msgbox*) para tal com fácil configuração, não necessitando o desenvolvimento através do GUIDE. A figura

36 mostra o instante de chamada da GUI onde a interface de instruções é obtida pelo comando "*msgbox*".

Demais problemas de implementação não apresentam fatores relevantes aos objetivos do ambiente, sendo normalmente, objeto para melhorias sem mudança de conceitos.

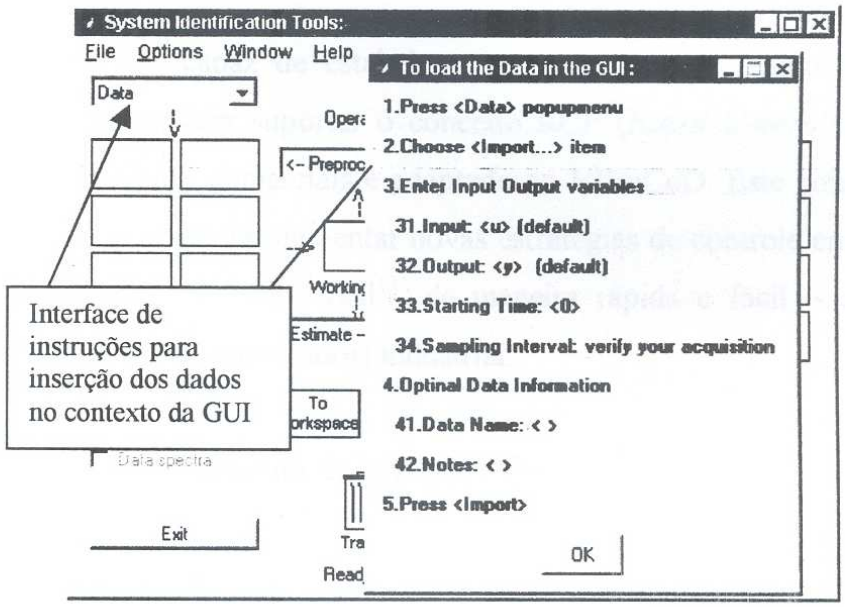


Figura 36 : Manipulação inicial da GUI do *Identification Toolbox*

4.2.2 Plataforma de aquisição e controle baseada em Instrumentação Virtual

A plataforma de aquisição integrada ao ambiente de análise e projeto de sistemas de controle deve possuir algumas propriedades como fácil configuração, controle, acesso de dados e monitoração do *hardware* escolhido e utilizado, através do PC. Para tal, é necessário a utilização de um ambiente de desenvolvimento de Instrumentação Virtual que permita a implementação do ambiente de *software* de suporte, com manutenção das características do ambiente de gerenciamento do *toolbox*.

Definida a plataforma de aquisição (*hardware* – barramento VXI, placa DAQ,... etc) que atenda as características imposta pelo sistema (tratamento de sinais, sinais analógicos e digitais, constantes de tempo – velocidade de conversão e processamento,

resolução,... etc) desenvolveram-se as GUIs específicas ao suporte de etapas do CDSC - a Identificação e Simulação com HIL. Criaram-se, portanto, sistemas modulares para o suporte da configuração do sistema (planta e plataforma de aquisição), identificação e validação do controlador. Estes sistemas são caracterizadas como módulos ou instrumentos virtuais que compõem a ECUVI (*Electronic Control Unit Virtual Instrumentation-based*), estrutura detalhada mais adiante.

A ECUVI deve ser capaz de estabelecer os requisitos de tempo necessários a simulação com HIL e também suportar o conceito RCP (*Rapid Control Prototyping*), sugerido em alguns sistemas comerciais e adaptado ao VIEnCoD. Este conceito implica na capacidade do ambiente em implementar novas estratégias de controle em estrutura de *hardware* com condições de tempo real[13], de maneira rápida e fácil – antes de sua implementação em estrutura (controlador) industrial.

A seguir é abordado a estrutura de *software* da ECUVI.

4.2.2.1 Estrutura ECUVI

O emprego da denominação ECUVI (*Electronic Control Unit Virtual Instrumentation-based*) foi o resultado da adaptação da concepção de *software* e *hardware* visando a simulação com HIL, de algumas plataformas com proposta CACSD apresentadas no capítulo 2, por exemplo, a plataforma DSPACE.

A ECUVI é a denominação de toda estrutura de suporte para simulação com HIL que é composta por elementos de *hardware* (plataforma de aquisição e controlador) e *software* (supervisão e controle da instrumentação e implementação do controlador). A concepção do VIEnCoD determina a composição de *hardware* da ECUVI baseada em plataforma aberta de instrumentação e o PC (processador) como controlador. É neste *hardware* que são processados os elementos de *software* – aplicativos (desenvolvidos em ambiente de Instrumentação Virtual) de monitoração e controle da instrumentação e suporte para lei de controle sob processo de validação. Ambos os processos operando em condições de tempo real[13] sob ambiente operacional Windows através do RTX,

[13] Tópico a ser abordado nos itens 4.2.3.3, 4.2.3.4 e 4.2.3.5 adiante.

abordagem a ser detalhada no próximo item (4.2.3). Esta estrutura é definida, portanto, pelo termo ECU (elementos de *hardware*) – VI (elemento de *software*). Esta estrutura é indicada na figura 40 no item seguinte e figura 22 do item 4.1.3 determinando sua participação.

Conforme observa-se nestas figuras, a atuação da ECUVI se dá através dos chamados processos *online*, onde os requerimentos de tempo real são exigidos. Estes processos caracterizam tarefas relativas a duas etapas do CDSC : Identificação e Simulação com HIL. Este contexto sugere a modularização da estrutura ECUVI em partes funcionais correlatas a função desempenhada. São os módulos :

- **PCVM** – *Plant Configuration VIEnCoD Module* : é o módulo responsável pela configuração do sistema sob projeto – sistema de aquisição e especificação das grandezas físicas da planta e sinais envolvidos. Apesar de ser um processo *offline*, representa a interface de entrada e armazenamento dos dados utilizados pelos demais módulos da ECUVI.

- **PIVM** – *Plant Identification VIEnCoD Module* : módulo responsável pela supervisão e controle de todo o processo de aquisição dando suporte aos requisitos para obtenção de realizações necessárias ao processo de Identificação e modelagem.

- **CIVM** – *Controller Implementation VIEnCoD Module* : módulo responsável pela implementação do controlador (lei de controle) que deve executar em condições de tempo real em simulação com HIL. Todo o controle e supervisão desta simulação é feito através deste módulo.

Para cada módulo formam desenvolvidos aplicativos que encapsulam rotinas de interface com os dispositivos de *hardware*, lógica do processo, estrutura de suporte da lei de controle e GUIs. A descrição mais detalhada das características e funcionalidade de cada módulo é dedicada ao capítulo 5 deste trabalho.

Estes aplicativos, desenvolvidos em LabWindows/CVI têm sua estrutura de programa formada pelos elementos indicados na figura 37. Tal estrutura compõe o projeto

do módulo cujas informações (arquivos e localização) são armazenadas no arquivo *nome.prj*. São eles :

- *nome.c* : arquivo fonte;
- *nome.h* : arquive de *include files*; estrutura criada quando a definição das variáveis e *callbacks* das GUIs (nome.uir);
- *nome.uir* : arquivo de interface do usuário; projeto do *layout* das GUIs;
- tkvx*.fp : arquivo do driver do instrumento utilizado;

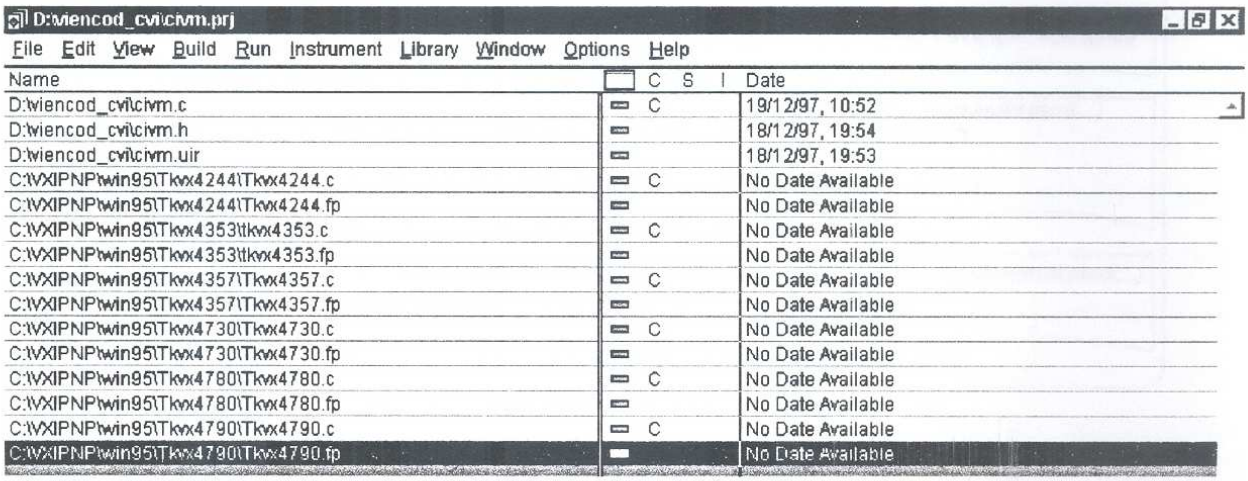


Figura 37 : Estrutura do projeto de um módulo da ECUVI

Cada módulo apresenta a mesma estrutura, sendo "nome" referido aos três componentes da ECUVI : *pcvm.**, *pivm.** e *civm.** . O VIEnCoD em sua versão inicial apresentava todos os arquivos relativos a cada módulo concentrados em uma única estrutura (*viencod.prj*). Tal concepção foi em função da tentativa de implementação de todo o ambiente CACSD em LabWindows/CVI, conforme já mencionado. Após a definição do ambiente como *toolbox* iniciou-se um trabalho de modularização da enorme estrutura formada visando o atendimento das tarefas bem específicas : configuração do sistema, identificação (aquisição) e simulação com HIL.

A estrutura, recursos e técnicas utilizadas no projeto das GUIs são análogas as utilizadas para suporte do processo *offline* do ambiente, em ambiente Matlab. Uma GUI apresenta diferentes elementos aos quais são atribuídas funções denominadas de *callbacks*. O código destas *callbacks* (podendo ser gerado através do CodeBuilder)

encontram-se no arquivo fonte *nome.c* sob a estrutura mostrada na figura 38 . As técnicas de figuras dinâmicas, com redefinição automática do campos e *layout* da GUI em função do contexto, são também aplicadas, conforme será visto no capítulo 5 deste trabalho.

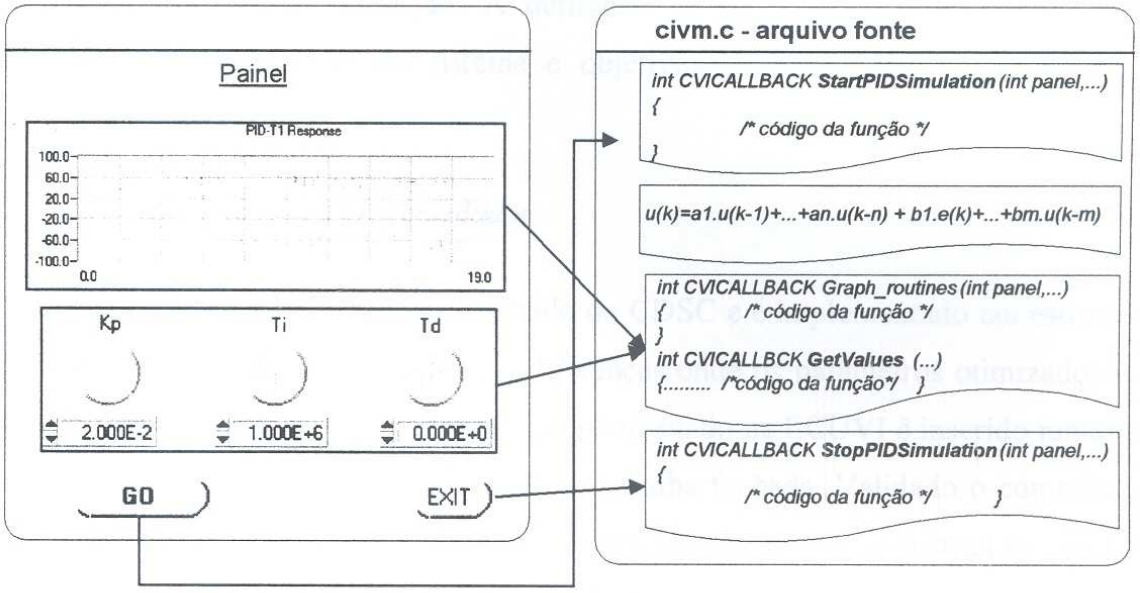


Figura 38 : Estrutura de programa sob conceito de *callbacks*

Os módulos componentes da ECUVI promovem a troca de dados entre si e entre eles e a estrutura de arquivos do *toolbox* definido anteriormente. Este fluxo de dados bem como a geração de arquivos é ilustrada nas figuras 34 e 35 do item 4.2.1.2.

4.2.3 Considerações sobre simulação com HIL

A metodologia de simulação com HIL (*hardware-in-the-loop*) tem sido amplamente utilizada como ferramenta de suporte ao desenvolvimento de sistemas de controle. A sua utilização em sistemas de controle complexos, como sistemas mecatrônicos, é observada em plataformas como o CAMeL e dSPACE, vistos no capítulo 2.

Nestas, a estrutura física do sistema de controle é decomposta em subsistemas funcionais. Estes subsistemas podem ser modelados matematicamente e encapsulados em estruturas funcionais (ex.: blocos, função de transferência) onde são bem definidas suas

interfaces definindo condições de acoplamento. Portanto, elementos físicos coexistem com elementos implementados computacionalmente durante as simulações.

A configuração de um sistema com a adoção de simulação com HIL permite o interfaceamento de modelos desenvolvidos (planta ou controlador) com elementos físicos reais com propósito de validação. A definição da maneira de utilização da técnica depende das características do sistema e objetivos propostos. As possibilidades são descritas a seguir.

♦ *Planta – modelo do controlador*

O modelo do controlador é o resultado do CDSC e é implementado em estrutura de programa (C) sob a forma de equação a diferenças onde os parâmetros otimizados é que definem a lei de controle. Tal controlador implementado na ECUVI é inserido juntamente com a planta física (real) para a simulação em malha fechada. Validado o controlador, a lei de controle (parâmetros) são transferidos para um dispositivo industrial (p. ex. CLP). Esta técnica é conhecida também como validação do controle *online* e é a utilizada na atual implementação do VIEnCoD.

♦ *Controlador implementado – modelo da planta*

A lei de controle presente na ECUVI é validada através da simulação em malha fechada com o modelo da planta e não a real. Ou ainda, em sistemas complexos, a planta pode apresentar-se em subsistemas em elementos físicos e modelos. Esta forma de validação do controle é conhecida como validação *offline*, onde não existe a presença física da planta na malha de simulação. Esta forma de simulação não é implementada pelo VIEnCoD pelos seguintes fatores : - normalmente os testes são feitos com protótipos não apresentando riscos de operação; - o controle implementado na ECUVI é apenas o modelo discretizado do modelo do controlador obtido no processo de otimização e já simulado *offline*.

4.2.3.1 Processos no VIEnCoD

A atual implementação do VIEnCoD distingue basicamente dois processos, envolvidos no CDSC proposto, com relação as propriedades de operação e participação

de elementos físicos na malha : processo *offline* e processo *online*. Ambos são descritos a seguir.

♦ ***Processo offline***

Referencia-se ao ambiente de análise e desenvolvimento abrangendo todas as tarefas destinadas desde a identificação ao projeto do controlador. O processo inicia-se com o recebimento dos vetores de realização feitos pelo módulo PIVM; em seguida os mesmos são tratados para utilização na identificação e validação do modelo matemático; de posse do modelo parte-se para definição e otimização de uma estratégia de controle. A finalização do processo ocorre quando os parâmetros do controlador são enviados à unidade ECUVI para o processo de validação. Todas as etapas do CDSC são feitas sem a inserção de elementos físicos não havendo mais nenhuma realimentação para propósitos de validação, ou seja, o CDSC desde a identificação da planta ao projeto do controlador é em *offline*.

Este processo impossibilita, portanto, a implementação de estratégias de controle adaptativas onde é necessário a identificação da planta em períodos pré definidos (p. ex. estimação paramétrica através dos mínimos quadrados recursivos) de forma a atualizar os parâmetros do controlador para a adaptação às mudanças no sistema (p. ex. controlador de variância mínima generalizado). A figura 39 ilustra este diagrama onde cada bloco apresenta facilidades de implementação pelo VIEnCoD, porém sem o amparo da recursividade possibilitada pelas condições em tempo real.

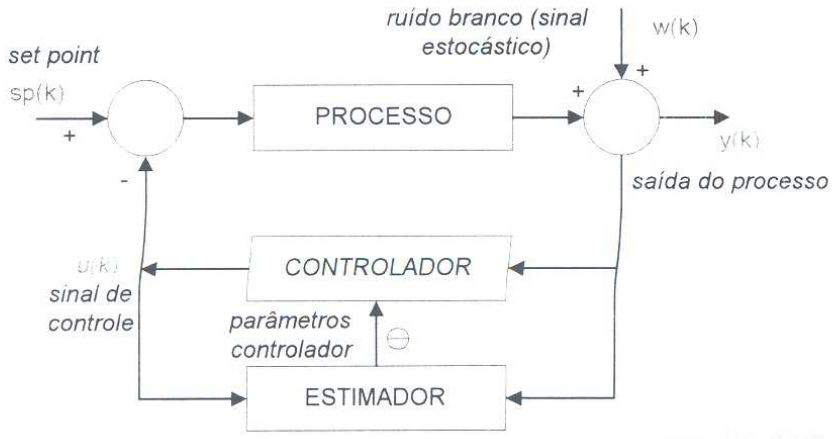


Figura 39 : Sistema de controle adaptativo – processo recursivo *online*

♦ ***Processo online***

O processo dito *online* se restringe a etapa de validação do controlador onde os parâmetros do controlador obtidos em processo *offline* são inseridos em uma estrutura de programa (algoritmo) na ECUVI que é executado em condições específicas de tempo real. Como o processador da estrutura de *hardware* da ECUVI implementada é o PC, onde trabalha o Windows 95/NT, faz-se necessário a utilização de artifícios que gerenciem o sistema operacional da máquina estabelecendo escalonamento das prioridades de eventos. A descrição mais detalhada deste gerenciamento é feita no item seguinte (4.2.3.2). A adoção deste recurso substitui com bom desempenho a utilização de placas DSP que ainda apresentam custo elevado.

Esquematicamente tem-se :

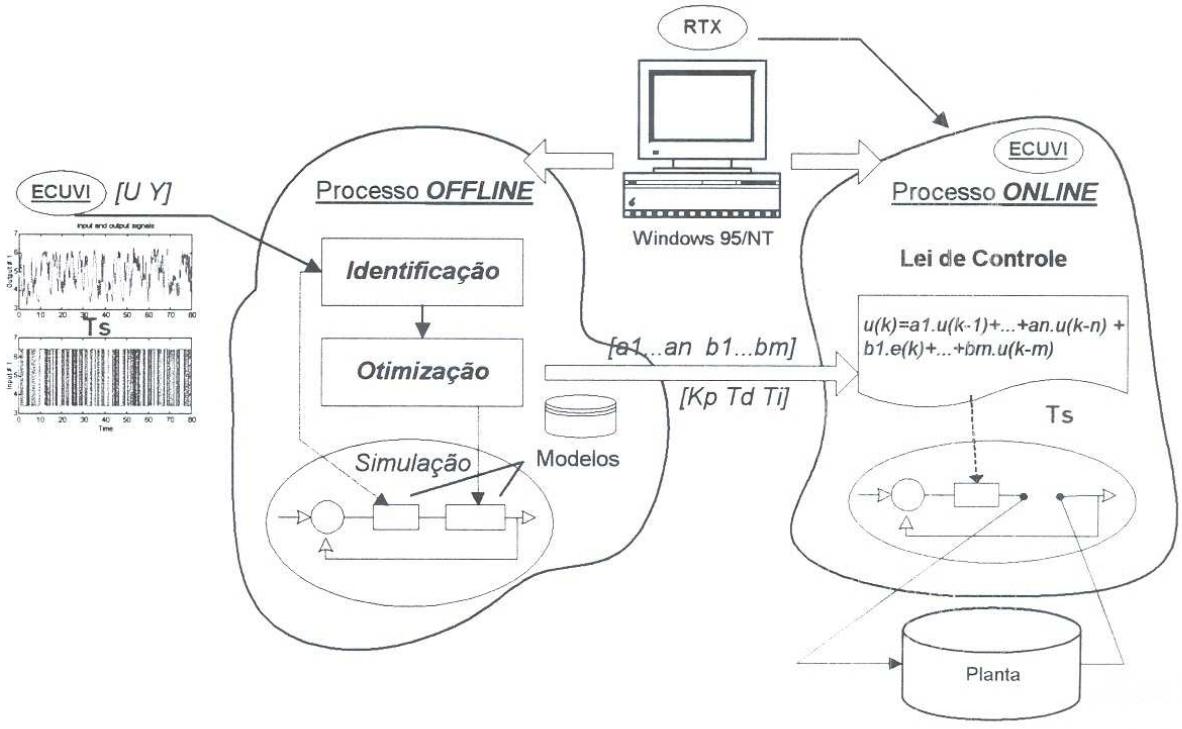


Figura 40 : Estrutura operacional do VIEnCoD

4.2.3.2 Processos - simulação com HIL em revista

As propriedades colocadas acima revelam condições que limitam a aplicação adequada da simulação com HIL em suas várias formas. A utilização das rotinas presentes no CDSC formam um processo *offline* à parte sem qualquer interação com

elementos de *hardware*. Isto limita a flexibilidade do ambiente a implementação de sistemas mais complexos onde a planta apresenta subsistemas físicos e modelados participando da malha durante as simulações. Esta mesma limitação se aplica a sistemas adaptativos onde é necessário a utilização das rotinas presentes no processo *offline*, de forma recursiva.

Sob este contexto, a aplicação do HIL se restringe apenas à tarefa de validação do controlador obtido no processo *offline* de otimização e implementado pela ECUVI, onde juntamente com a planta física, faz-se a simulação do sistema em malha fechada.

Um outro ponto importante relacionado à técnica HIL é a forma de ***implementação da lei de controle***. Durante a fase *offline* de projeto e otimização, uma estrutura de controlador é definida em uma das formas disponíveis de representação (espaço de estados, função de transferência,... etc) e seus parâmetros são otimizados em função de critérios de desempenho. Tal processo foi abordado no capítulo 3. Em seguida o modelo é discretizado obtendo-se sua respectiva função de transferência no plano z e consequentemente, a equação a diferenças que descreve a lei de controle. A estrutura genérica que descreve esta lei de controle é implementada na ECUVI onde apenas os parâmetros do controlador, [a1 a2 ... an b1 b2 bm], são recebidos do processo *offline*. A implementação de uma estratégia de controle que não possa ser descrita por uma estrutura polinomial A/B (p. ex. um algoritmo recursivo implementado em *m-file*), encontra obstáculos nesta forma de implementação. Uma solução é a utilização dos recursos Matlab Compiler/RTW que permite a geração de código C de qualquer estrutura Matlab (*m-files*) e Simulink (diagramas).

A seguir são abordados os recursos diretamente relacionados a possibilidade de implementação da simulação com HIL integral e sob condições de tempo real. Após é feita uma revisão na fase atual de implementação e estrutura do VIEnCoD com a proposição de soluções.

4.2.3.3 Ambiente de simulação em tempo real

Conforme colocado, o VIEnCoD apresenta uma solução CACSD integrada de uma plataforma de instrumentação virtual (sistema de aquisição) à uma plataforma de análise e

projeto de sistemas de controle. Ambas as plataformas compartilham a mesma estrutura de *hardware* (PC) onde trabalha o Windows NT.

A implementação de sistemas de controle com simulação com HIL exige condições específicas quanto ao tempo (desempenho) de processamento (p. ex. sistemas com constantes de tempo pequenos exige processamento rápido) e garantia nos tempos aplicados (p. ex. período de amostragem) implicando determinismo ao sistema. Tais condições definem as características de tempo real e que não são encontradas no sistema proposto fazendo-se necessário a adoção de recursos ao nível de sistema operacional (OS – *Operational System*). Este contexto, de suma importância no projeto, é avaliado neste item onde são desenvolvidos de forma simplificada os conceitos fundamentais de sistemas em tempo real e as características de OS para suporte através do Windows NT, no campo da aplicabilidade em sistemas de controle.

Sistemas em tempo real são frequentemente classificados como "**rígidos**" (***hard***) e "**flexíveis**" (***soft***). Os sistemas rígidos são requeridos, sem falha, para satisfazer todos os requerimentos de resposta no tempo a todo momento sob qualquer circunstância e qualquer combinação diferente de eventos. São sistemas associados a *processos críticos* que se não realizados dentro de um tempo final de execução (deadline), têm efeitos catastróficos ao sistema. Um exemplo de um sistema de tempo real rígido é o sistema de controle de um reator de uma planta geradora nuclear.

Já um sistema de tempo real flexível está relacionado um desempenho médio onde requer-se que a maioria (não a totalidade) dos casos tenham os requerimentos de resposta no tempo atendidos. São sistemas associados a *processos essenciais* (possuem um tempo final para execução – *deadline*; se não cumprido não existe efeito catastrófico) e *processos não essenciais* (não acarretam prejuízo maior ao sistema quando não cumprido dentro de unidades de tempo pré-estabelecidas).

Controle em tempo real é a habilidade de necessariamente e sem falhas, responder a um evento dentro de um período de tempo garantido. Por exemplo, na execução da malha de controle ilustrada na figura 41, deve-se garantir que a saída do controlador responda ao valor de entrada medido dentro de um intervalo de tempo específico,

conhecido como *tempo de ciclo de malha de controle*. Se este tempo é constante, então o sistema de controle é estável e com comportamento determinístico; se varia não existe garantia quanto a estabilidade do sistema. Portanto, fica implícito no nome de "sistema em tempo real" a necessidade de controle de tempo fornecido para que seja lidos os sensores e atualizadas as saídas do sistema.

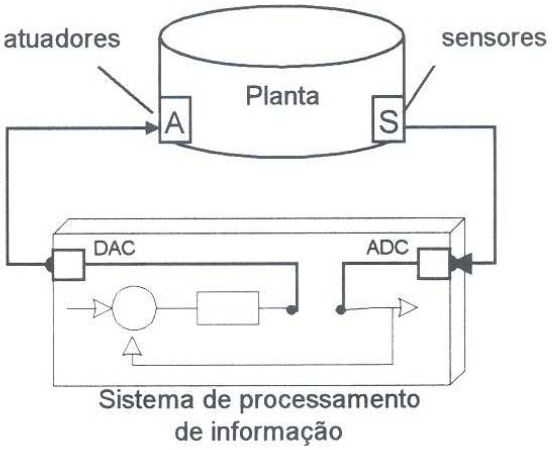


Figura 41 : Sistema de controle clássico

O diagrama de tempos mostrado na figura 42 mostra os parâmetros que definem o desempenho de um sistema de tempo real. Os tempos envolvidos neste diagrama são:

- **Ts** : Período de amostragem desejado.
- **TL** : Tempo de latência. O tempo consumido para paralização de todas as tarefas de primeiro plano e início da seção de tempo real.
- **Te** : Período de amostragem efetivo. Tempo decorrido entre a última amostra e a próxima.
- **Tc** : Atraso de computação. Tempo gasto para execução do código na rotina de serviço de interrupção.
- **Tf** : Tempo de primeiro plano. Tempo remanescente para tarefas de primeiro plano antes que a próxima interrupção ocorra.

Quando uma interrupção ocorre, em **Ts** fixo, o processador pára a execução do que esta fazendo (tarefa de primeiro plano) e inicia a execução do controlador de tempo real ou a Rotina de Serviço de Interrupção (ISR – *Interrupt Service Routine*). O tempo tido entre a ocorrência de uma interrupção de hardware e o atual início da ISR determina o

período de latência **TL**.O ideal é que este tempo seja o menor possível sendo mais importante o fato de que o mesmo seja o mais constante possível. O tempo entre o início da execução de uma ISR e o início da próxima representa o período de amostragem efetivo **Te**. Este valor deve ser o mais próximo possível de Ts. Se TL é constante, então Te=Ts e a ISR é atrasada por TL. O tempo que isto leva para executar a ISR é o atraso de computação – **Tc** e deve ser menor que o período de amostragem efetivo Te ou o sistema não evolui como esperado. O tempo entre o final de duas execuções de ISR é o tempo remanescente em primeiro plano, **Tf**, para o processador desenvolver outras tarefas como a manutenção de GUIs e permitir que outros programas sejam executados.

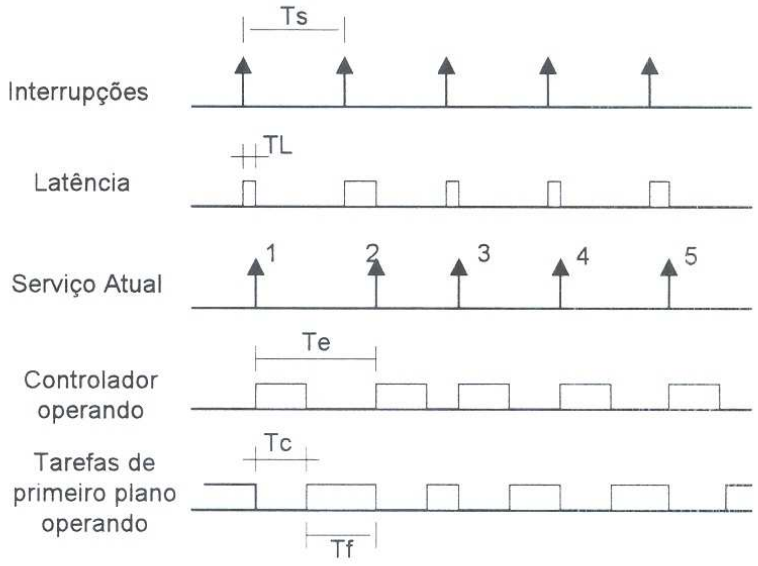


Figura 42 : Diagrama de tempos em sistema de tempo real

4.2.3.4 Controle em tempo real baseado em Windows NT

A plataforma Windows hoje apresenta-se de forma expressiva sobre os computadores pessoais PC, existindo um grande número de aplicativos de controle (aquisição, análise e projeto) baseados nesta plataforma. Os ambientes de suporte do VIEnCoD LabWindows/CVI e Matlab representam este conjunto. O Windows em sua estrutura natural não é um sistema operacional de tempo real, limitando aplicações com simulação com HIL que é uma tendência atual na composição de um pacote CACSD. Entretanto, através da manipulação nos elementos que o compõem é possível a determinação de condições de tempo real bastante satisfatórias. O seu projeto inclui

elementos RTOS (*Real Time Operational System*) suficientes para gerência de prioridades e escalonador preemptivo. Esta abordagem é realizada a seguir.

♦ ***Escalonador e Prioridades***

Em um sistema operacional multitarefa, o trabalho do ***escalonador*** é selecionar dentre um conjunto de tarefas presentes a mais apropriada para execução. É o mais central componente em um RTOS. Um escalonador que não seleciona corretamente as tarefas para execução pode causar a perda de importantes prazos finais de execução.

Um sistema em tempo real normalmente consiste de tarefas em tempo real (ou tempo crítico – como alarmes de paralização de emergência) e tarefas de não tempo real (como atualização de gráficos em tela). Se ambas estas tarefas estão prontas para operação, a tarefa em tempo real é escalonada primeiro. O escalonador distingue estas tarefas através de ***níveis de prioridade***. O nível de prioridade é um número associado a cada tarefa que representa essencialmente a importância relativa da tarefa. No Windows NT, por exemplo, níveis de prioridade vão de 0 à 31. Esta granularidade na definição de níveis de prioridade permite flexibilidade e controle de como as tarefas são priorizadas.

Em acréscimo às prioridades, um escalonador em tempo real deve também oferecer ***escalonamento preemptivo*** : o sistema operacional pode temporariamente parar a execução de um tarefa para a execução de outra. Isto é muito importante em um sistema em tempo real, onde o sistema operacional pára (*preempt*) a tarefa corrente quando uma tarefa de prioridade superior está para executar.

O Windows NT possui um escalonador preemptivo de encaminhamento de prioridades (*priority-driven preenptive scheduler*). Possui 32 níveis de prioridade dividido em duas classes – tempo real e dinâmica. Tempo real inclui os níveis de prioridade mais elevados (16 à 31). Dinâmica inclui os níveis de prioridade inferiores (0 à 15).

Com o conhecimento de como o escalonador e prioridades trabalham no Windows NT, pode-se utilizar o API (*Application Program Interface*) Win32 para construir

aplicações utilizando múltiplos processos e tarefas. Definindo diferentes prioridades aos processos e tarefas envolvidas consegue-se um controle sobre o sistema.

Uma abordagem muito comum em sistemas de tempo real é como as GUIs afetam as características de tempo real - se as mesmas degradam e se existem atividades de mouse ou teclado. Por exemplo, em condições normais de operação do Windows, uma tarefa que levaria para execução uma média de 1.5 ms, pode levar acima de 60 ms com atividades em paralelo de mouse e teclado. Estas condições dependem do tipo da máquina e das tarefas carregadas. A definição de prioridades de tempo real podem tornar imune a atuação destes dispositivos durante a execução de uma tarefa prioritária.

Uma condição imposta neste contexto é que deve-se evitar a execução de toda a aplicação em prioridade de tempo real evitando consequências, como por exemplo, mouse e teclado travados. *A melhor solução é a separação dos componentes de tempo real dos demais (p. ex. GUIs) e execução dos processos de tempo real em prioridade de tempo real.*

♦ ***Interrupções***

Interrupções são componentes de suma importância para sistemas de tempo real. Uma interrupção é um mecanismo que informa o sistema de tempo real que algum evento externo solicita para ser executado. O sistema deve ser capaz de responder a estes eventos rapidamente. Uma interrupção é tratada por uma rotina de serviço de interrupção (ISR – *Interrupt Service Routine*), uma rotina de *software* automaticamente executada pelo sistema operacional quando uma interrupção ocorre. O ISR é reponsável em executar uma ação apropriada em resposta a interrupção.

O Windows NT suporta interrupções em uma base preemptiva. Quando uma interrupção de ordem superior ocorre, ISRs de interrupções de prioridade mais baixa podem ser preemptivas. Quando a ISR de uma interrupção de ordem superior está sendo executada, qualquer interrupção de ordem menor que ocorra deve esperar.

Para manter ISRs curtas, o Windows NT incorpora um mecanismo chamado de *chamada de procedimentos adiados* (DPC – *Deferred Procedure Calls*). DPC é uma rotina que é invocada após a ISR. DPCs tem menor prioridade do que ISRs, e não bloqueiam outras ISRs na execução. A Microsoft recomenda que a ISR execute somente o mais crítico processamento necessário fazendo isto o mais rápido, e que as DPCs executem os processamentos remanescentes não críticos. Este projeto assegura que o sistema tenha melhores características. A DPC executa em algum momento depois da ISR, dependendo de outras interrupções que estão pendentes.

Um obstáculo significante na utilização do Windows NT para sistemas em tempo real é que muitos *drivers* de dispositivos não são apropriadamente escritos quando fornece suporte de interrupção. Com estes *drivers* de dispositivos, a ISR pode levar muito tempo e efetivamente bloquear o sistema (p. ex. alguns *drivers* de disco e vídeo). A solução mais direta é identificar e desabilitar os dispositivos ofensivos (através do Gerenciador de Dispositivos do Painel de Controle do Windows). A dificuldade reside na escolha dos dispositivos. Algumas sugestões para desabilitação: proteção de tela (pode interromper a aplicação), mouse e teclado (se a aplicação em tempo real pode executar sem a interação do usuário), rede, porta serial,... etc.

- ***Hardware específico para suporte de interrupções***

Se a aplicação necessita tempo de reposta de interrupção em micro segundos ou existe requerimentos rígidos de tempo real que não podem ser garantidos pelo Windows NT, utiliza-se estruturas de *hardware* à parte, como processadores DSP ou processadores dedicados de proposta geral. Estas estruturas, geralmente sem a existência de sistema operacional, operam apenas o algoritmo de controle garantindo as condições de tempo real. Os dados para monitoração e atualização são enviados para o PC (onde executam os aplicativos Windows), sem a necessidade de operação em tempo real. A plataforma dSPACE apresenta esta concepção.

Em alguns sistemas, existe *hardware* dedicado de I/O para suportar operações de aquisição e liberar o processador principal para atendimento de tarefas de tempo real. Esta concepção é apresentada pelo sistema WinCon 3.0.

Como a evolução nos processadores ocorre com grande velocidade (em média a cada 18 meses dobra-se o desempenho) a necessidade de *hardware* integrado adicional pode ser reavaliada.

4.2.3.5 Tempo real no VIEnCoD

Toda abordagem relativa a operação de sistema de tempo real em Windows NT representa a principal preocupação do presente projeto. A sua concepção elimina a necessidade de *hardware* adicional integrado, sendo portanto, necessário a utilização de um recurso de *software* que trabalhe o contexto descrito anteriormente no sistema operacional Windows. A ferramenta adotada para este trabalho é o **RTX** da *VenturCom.* É um aplicativo que apresenta uma grande inserção no meio científico e industrial. Informações detalhadas sobre esta ferramenta podem ser encontradas em seu site[14]. *Ainda não foi feito nenhum trabalho de implementação pois tal aplicativo encontra-se em processo de aquisição pela instituição. Realizou-se apenas um estudo apurado para verificação do real atendimento das necessidades impostas.*

Além da preocupação da operação em sistema operacional Windows, o VIEnCoD apresenta outros pontos de destaque que tem influência direta na definição das condições de tempo real exigidas em simulações com HIL.

- ***Módulo CIVM***

O módulo CIVM apresenta uma única estrutura composta por rotinas de inicialização do sistema, leitura e escrita de dados e atualização gráfica da GUI. É o módulo responsável pelo gerenciamento da simulação com HIL, portanto a parte do VIEnCoD onde são exigidas condições de tempo real.

Primeiro procede-se o levantamento das constante de tempo do sistema (planta) que definirão o tempo de ciclo de malha de controle correspondendo ao tempo necessário para leitura de dados vindo dos sensores à atualização do sinal de controle na saída. Este levantamento determina o intervalo de amostragem Ts, que é avaliado junto às condições

[14] *VenturCom*, "*RTX 4.3 Features and Benefits*", http://www.vci.com/products/vci_products, Out. 1999.

impostas pelo RTX no gerenciamento do sistema operacional. Dependendo das condições impostas de tempo, como constantes de tempo muito pequenas (micro segundos), o Ts não é atingido exigindo *hardware* adicional.

A definição através do RTX dos níveis de prioridade estão diretamente relacionados com o T_L (Tempo de Latência . que é desejável que seja pequeno e constante) e o Tf (*Foreground Time*). Também é necessário a desabilitação de todos os drivers de dispositivos ofensivos comentados e tarefas desnecessárias ao processo. Isto ajuda na manutenção do tempo de latência constante evitando efeitos como *jitter* (produzido com esta variação) e garantindo um período de amostragem efetivo Te muito próximo ao Ts, além da liberação do Tf para atividade mais dedicada com a GUI do CIVM. O Tc (controlador de tempo real) deve esta compreendido dentro de Te, e corresponde ao tempo de ciclo de malha de controle. O Tc dever ser suficiente para a execução do ciclo.

4.2.3.6 Recurso Matlab Compiler / RTW

As ferramentas Matlab Compiler e RTW (*Real Time Workshop*) são destinadas a tornar o ambiente Matlab aberto a implementações em código C. Funções Matlab e diagramas Simulink são convertidos em código C cuja estrutura deve ser depurada para adaptação em ambiente externo.

Baseado nesta possibilidade o VIEnCoD em sua concepção inicial dedicou atenção para utilização destas ferramentas, visando alguma forma de integração a nível de código com a plataforma de instrumentação virtual LabWindows/CVI. Esta motivação foi baseada em alguns pontos :

- Do pouco conhecimento das potencialidades de programação visual do Matlab, toda a estrutura operacional e de interfaces (GUIs) do CACSD seria desenvolvida juntamente com a plataforma de instrumentação virtual no LabWindows/CVI. A padronização do ambiente em C sugeria a incorporação dos recursos Matlab desta forma, através do Matlab Compiler;
- A necessidade em se dotar o VIEnCoD da capacidade de implementação de qualquer estratégia de controle, resultado do CDSC, em simulação com HIL. Algoritmos de controle poderiam ser obtidos de blocos Simulink que

encapsulavam funções de transferência definindo os controladores. O RTW trabalharia o bloco de modo a defini-lo como função com entrada e saída de parâmetros (sinais).

As dificuldades encontradas que impediram a implementação destes pontos com as ferramentas foram objeto de trabalhos finais de graduação[15]. A principal delas recaiu no conflito conceitual com a proposta de concepção do ambiente, exigindo a existência de uma terceira plataforma (Visual C 5.0 ou superior e outros) para compilação. A proposta é a criação de uma ambiente CACSD através de ferramentas de desenvolvimento amigáveis e comuns ao universo do estudante, pesquisador e engenheiro de controle.

Em acréscimo, o arquivo C gerado apresenta uma estrutura de difícil depuração, não seguindo a mesma lógica de um programa C comum. Isto se agrava quando são convertidos recursos (*m-files*) que internamente procedem a chamada de outro *m-file*.

Estas dificuldades obrigaram a busca de uma solução que permitisse ao VIEnCoD a simulação com HIL de qualquer estratégia de controle obtida no CDSC (processo *offline*): a transferência dos parâmetros de uma equação a diferenças (lei de controle) ao processo *online* sob características de tempo real. Esta solução apresenta total suporte conceitual e técnico, conforme discutido anteriormente, atendendo a proposta de validação do controlador e fechamento do CDSC.

Apesar desta implementação possibilitar ao VIEnCoD suporte a um horizonte muito grande de aplicações, a utilização do Matlab Compiler e RTW é necessária para algumas situações que encontram nela obstáculo:

- Implementação de estratégias de controle adaptativas onde é necessário a utilização de recursos Matlab como algoritmos recursivos, presentes por exemplo, no *Identification Toolbox* (p. ex. Mínimos Quadrados Recursivos). Uma comunicação DDE com a ECUVI não atenderia os requisitos de tempo real.

---

[15] OLIVEIRA, G. L.; AMBRÓSIO, F. *VIEnCoD 2.0 - VXI-based Integrated Environment for Controllers Design*. Projeto Final do Curso de Engenharia de Computação. Pontifícia Universidade Católica do Paraná, 1998.

- Dar mais velocidade ao projeto através das MEX-files (estruturas em C que executam apenas no Matlab). Na atual implementação do VIEnCoD não existem no processo *offline* do CDSC situações que exijam esforço computacional acentuado, não apresentado portanto, preocupação neste sentido. Além disto, o processo *online* que possui a preocupação em desempenho, representa uma plataforma a parte do ambiente de análise e projeto do CDSC baseado em Matalab.

4.2.4 Estrutura global – suporte funcional ao CDSC

Todo o processo envolvido desde a abertura de um projeto à síntese e validação de um controlador (estratégia de controle) envolve uma constante troca de dados e atualização de arquivos de forma a suportar o gerenciamento sequencial que o ambiente proporciona. Uma parte deste fluxo de dados é estabelecido pela integração da plataforma *offline* Matlab/Simulink com o módulo de aquisição e controle ECUVI. As etapas de identificação e simulação com HIL são os maiores responsáveis por esta troca de dados.

Do conhecimento da atual estrutura de *hardware* (instrumentação) implementada no VIEnCoD, abordado no Anexo I desta dissertação, o CDSC explorado no capítulo 3 é repassado através de sua integração funcional com este sistema de instrumentação (*VXIbus*) onde são explicitados as trocas de dados efetuadas. Vale relembrar da abertura de *hardware* do ambiente, sendo utilizado esta estrutura apenas para auxílio descritivo.

4.2.4.1 Etapa de Identificação

Esta etapa engloba o processo de excitação da planta e leitura de sua resposta definindo uma *realização*. O projeto da interface gráfica do módulo PIVM permite a escolha dos sinais mais adequados ao processo de identificação estabelecendo o canal de comunicação com o gerador de formas de onda arbitrária (VX4790). De forma síncrona são habilitados os instrumentos VX4780 (condicionador de sinais) e o VX4244 (digitalizador) elementos responsáveis pela leitura da resposta do sistema sob condições adequadas de instrumentação. Vale lembrar que este processo corresponde ao *projeto de*

experimento, etapa inicial na metodologia proposta para o processo de identificação abordado no capítulo 3.

Ambos os vetores de excitação e resposta do sistema são armazenados na pasta do usuário (*data_io/tofrom_ECU/nome_uyfecu.dat*) compondo a biblioteca de realizações, subsídio para o processo de identificação da planta. As interfaces utilizadas para o suporte metodológico e teórico deste processo é visto no capítulo 5. Este processo é ilustrado na figura 43.

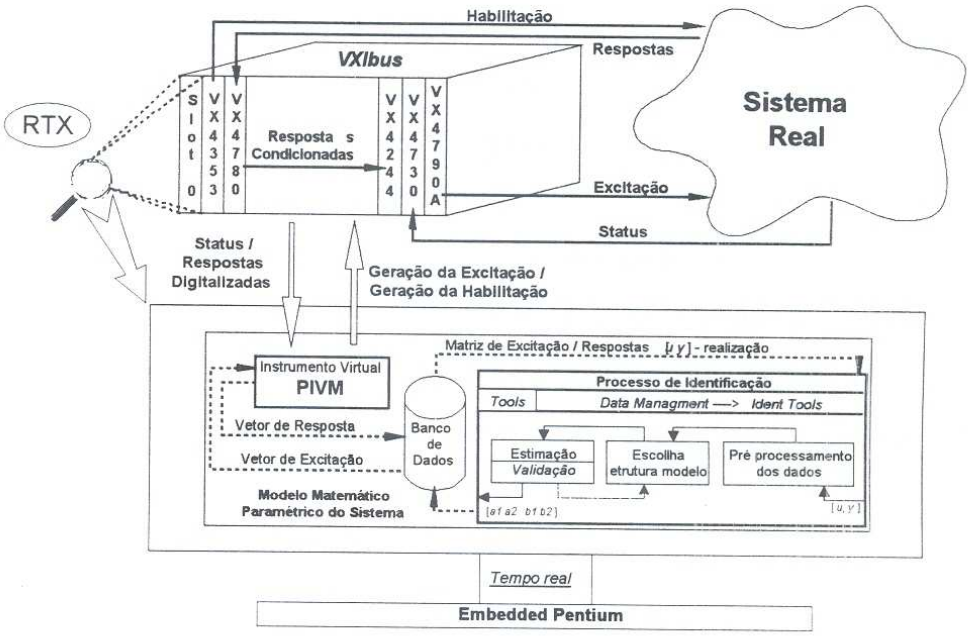


Figura 43 - Fase de Identificação

O modelo paramétrico da planta obtido neste processo é armazenado em (*models/plant/nome_id.mat*) que será utilizado para composição da malha de controle visando o processo de otimização do controlador.

4.2.4.2 Etapa de Otimização do controlador

Neste etapa primeiramente é necessária a definição do sistema em malha fechada na qual a planta e controlador estão inseridos. O VIEnCoD apresenta uma estrutura *single loop* sob a forma de diagrama onde sua configuração (definição de cada elemento – controlador, sensores, filtros) é gerenciada por uma GUI (*m-file*) dedicada.

A planta é representada pela estrutura (modelo paramétrico – *nome_id.mat*) obtido na etapa anterior e aceito por critérios de validação inerentes ao processo de identificação. Para o controlador define-se uma estrutura (modelo paramétrico genérico, estruturas PID) explicitando seus parâmetros (p. ex. [Kp Ti Td]) que serão objeto do algoritmo de otimização.

Configurado toda a malha, esta (o diagrama) é estruturada de forma a poder ser utilizada pelas ferramentas de otimização disponíveis. Após, é necessário a definição de critérios de desempenho estabelecidos em função de um sinal de excitação aplicado (p. ex. um degrau). Conforme visto no capítulo 3, a otimização visa a minimização de uma função objetivo. Conforme mostra a figura 44 a etapa de otimização é um processo *offline* não ocorrendo a troca de dados com a ECU (barramento VXI).

Os parâmetros do controlador otimizado são então armazenados e enviados à ECUVI para operação através do módulo CIVM para o processo de validação com simulação com HIL. Previamente a esta etapa, faz-se simulações com os modelos verificando o atendimento das especificações de projeto do sistema em malha fechada e fornecendo subsídios para a validação.

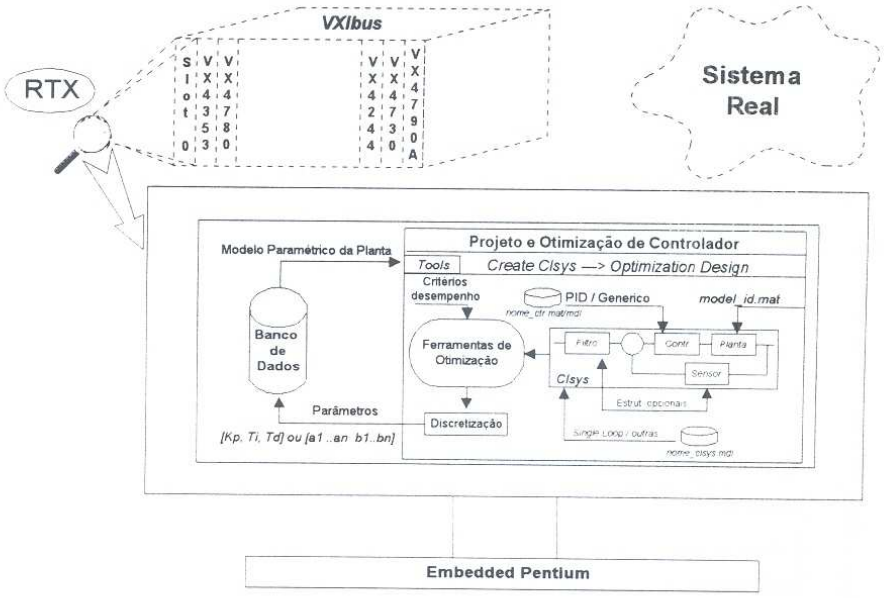


Figura 44 – Etapa de Otimização

4.2.4.3 Realização experimental para validação do controlador – simulação com HIL

Os parâmetros do controlador obtidos no processo de otimização são previamente manipulados de maneira a representarem os coeficientes da equação a diferença da lei de controle. Esta é suportada por uma estrutura adequada de programa que transforma a ECU, sob a gerência do módulo CIVM, na unidade *controlador* do sistema - figura 45.

Na simulação com HIL vale ressaltar a necessidade de critérios rígidos de tempo real onde são atendidas as constantes de tempo da planta e garantido o tempo de amostragem. Conforme abordado anteriormente neste capítulo, a imposição das condições de tempo real são estabelecidas pelo RTX juntamente com especificação adequada do sistema de aquisição.

O processo de leitura e escrita de dados é feito de forma semelhante ao especificado na etapa de identificação, somente que o VX 4790 é substituído por um conversor D/A (VX 4730) que disponibiliza o sinal de controle processado pelo algoritmo (lei de controle).

Os vetores de excitação e resposta do sistema são armazenados para posterior validação em processo *offline*, em confronto com os resultados do sistema simulado.

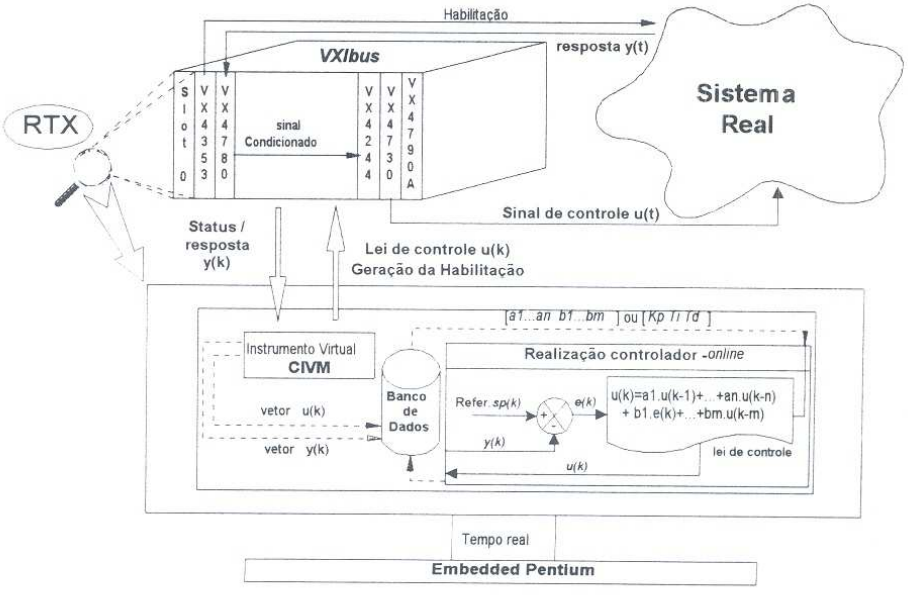


Figura 45 – Simulação com HIL

4.2.4.4 Implantação - controlador industrial

Esta última etapa, ilustrada na figura 46, visa a contextualização do objetivo final de todo o CDSC realizado pelo ambiente CACSD VIEnCoD : transferência e implementação da lei de controle obtida em controlador industrial.

Os parâmetros do controlador validado podem ser transferidos manualmente para o controlador industrial instalado junto com a planta, adequando à sua sintaxe. Tal tarefa, portanto, requer o conhecimento no manuseio do controlador (*software*) deixando o processo suscetível a erros.

A plataforma de instrumentação virtual adotada LabWindows/CVI, possui bibliotecas de protocolos de comunicação com um grande números de controladores industriais existentes. Desta forma, abre-se possibilidade para desenvolvimento de rotinas automáticas que efetuem a transferência dos parâmetros obtidos no CDSC de forma transparente e direta. Esta implementação é de grande valia para utilização do VIEnCoD na prestação de serviços onde se faz necessário o desenvolvimento em campo.

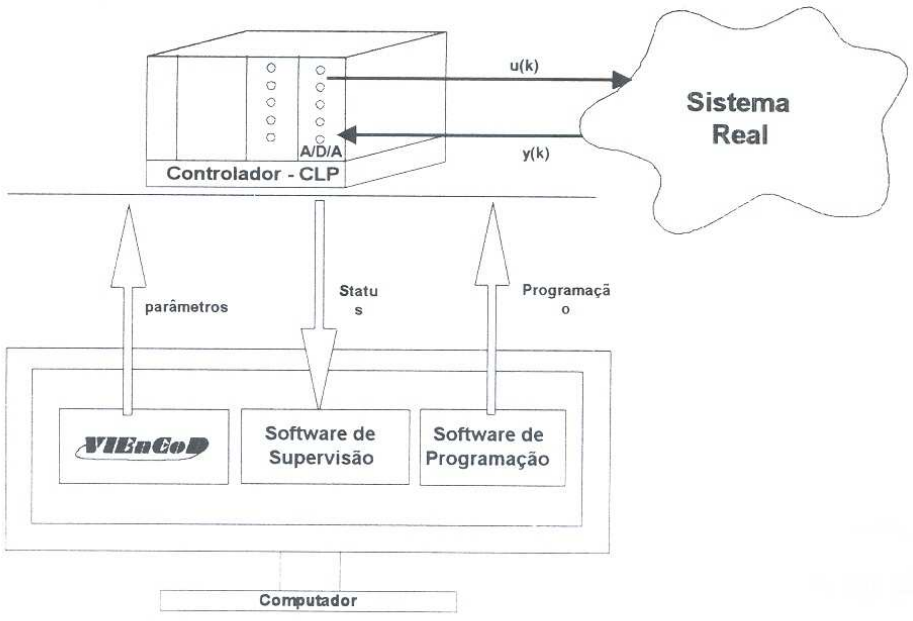


Figura 46- Fase de Implantação

4.3 Comentários

O projeto VIEnCoD é o somatório de dois contextos de desenvolvimento visando a integração sob metodologia específica na proposta de ambiente CACSD multifuncional: plataforma de Instrumentação Virtual e estrutura de software do *toolbox*. A maior complexidade ou pequenas falhas existentes foi determinada pelo tempo, sendo a validação das idéias apresentadas o objetivo principal deste trabalho.

A utilização do ambiente de Instrumentação Virtual LabWindows/CVI juntamente a aplicação do RTX na imposição de condições de tempo real ao PC, caracteriza o VIEnCoD como plataforma aberta em termos de *hardware*, que se traduz na ampla gama de plataformas de aquisição que o LabWindows suporta.

A grande potencialidade de programação e recursos gráficos e visuais do Matlab, aliado a forte tendência na construção de aplicativos nele baseados, definiram o VIEnCoD como *toolbox* com proposta CACSD. A estrutura lógica e seqüencial do CDSC foi traduzida no projeto de um ambiente modular e dotado GUIs que representam a principal estrutura de suporte ao ambiente. No projeto destas, técnicas especiais de implementação (árvore dinâmica, interfaces dinâmicas, *hints* adaptativos e outras) foram utilizadas de forma a suprir necessidades concomitantes de ensino (conteúdo teórico de controle e abordagem das ferramentas clássicas), pesquisa (utilização de técnicas avançadas) e serviços (conteúdo informacional e intuitivo elevados). A estrutura de arquivos e *m-files* representam esta modularidade, sendo cada *m-file* responsável por uma tarefa pertencente a uma etapa do CDSC. Da mesma forma, a organização dos arquivos baseados nos grupos de apoio ao CDSC. Isto permite fácil inserção de novas contribuições ao *toolbox*.

O desenvolvimento deste ambiente, principalmente dos recursos visuais, exigiu um esforço de implementação intrinsecamente baseado na análise de aplicativos já existentes em função da deficiência de fontes bibliográficas adequadas.

Para as simulações com HIL visando a validação das leis de controle obtida no processo *offline* do CDSC, são necessárias condições de tempo real. A análise realizada indica a necessidade de um dispositivo de gerenciamento do sistema operacional

Windows que opera no *hardware* utilizado pelo controlador, o PC. O dispositivo adotado é o RTX, sendo todo o estudo e especificação motivo do presente trabalho.

Capítulo 5

5 Descrição Funcional do VIEnCoD

Este capítulo visa a apresentação do projeto de software do *toolbox* VIEnCoD enfocando a disponibilização e gerenciamento de recursos, proporcionados pelas interfaces gráficas (GUIs) desenvolvidas. Desde a concepção e implementação houve a preocupação em dar suporte integral a cada etapa do CDSC sob o contexto educacional, de pesquisa, desenvolvimento e serviços. Os recursos são representados por :

- Todos os *m-files* existentes específicos a cada etapa do CDSC. Representam algoritmos desenvolvidos para a composição de um *toolbox*, por exemplo: *Control Toolbox, Identification Toolbox*,... etc;
- **Aplicativos implementados no ambiente de desenvolvimento de Instrumentação Virtual LabWindows/CVI e gerenciados através de *m-files* desenvolvidas;**
- M-files existentes responsáveis pelo suporte gráfico e operacional do Matlab;
- ***M-files* desenvolvidos para o gerenciamento transparente (para o usuário) de todos os recursos citados, implementação de novos e principalmente, o suporte de programação visual para a implementação das GUIs. Este conjunto de *m-files* desenvolvidos com objetivo de criação de um ambiente CACSD, compõem o novo *toolbox* baseado em Matlab, VIEnCoD.**

5.1 Módulos de Supervisão e Controle da ECU

A ECUVI (*Electronic Control Unit Virtual Instrumentation-based*) abordada no capítulo anterior, é responsável pela supervisão e controle da plataforma de *hardware*, responsável pela interface do ambiente de simulação e projeto de sistemas de controle com elementos físicos . Portanto, esta acumula as funções de sistema de aquisição de dados e elemento de implementação da lei de controle para validação do controlador através da simulação com HIL.

Esta funcionalidade exige um ambiente que atenda de forma amigável e eficaz tarefas relativas a três grupos ou módulos, determinados pelas etapas do CDSC. São eles:

1. Configuração do sistema de *hardware*: promove a interface do ambiente de simulação com o mundo físico;
2. Processo de Identificação: grupo responsável pelo suporte da primeira e segunda etapa no processo sugerido no capítulo 3 para identificação: projeto do experimento e aquisição e tratamento de dados, respectivamente;
3. Processo de validação do controlador : para supervisão e controle da simulação com HIL.

A seguir, cada um destes módulos são explorados de forma mais detalhada focalizando os suportes das diferentes necessidades de recursos através das implementações de GUIs.

5.1.1 Módulo de configuração da ECU - PCVM

O módulo PCVM (*Plant Configuration VIEnCoD Module*) suportado pela GUI ilustrada na figura 47, destina-se a promover a configuração adequada do sistema de aquisição em função das características previamente levantadas do projeto como um todo. Este objetivo é estabelecido através do suporte adequado para as tarefas :

- Levantamento e configuração das grandezas físicas e sinais relacionados à planta, objeto do projeto de controle;
- Levantamento e configuração de todas as variáveis de entrada e saída do sistema definindo a estrutura de controle MIMO ou SISO;
- Configuração da plataforma de *hardware*, em função das especificações e características dinâmica da planta (constantes de tempo) e sinais específicos dos sensores e atuadores bem como as restrições (valores máximos e mínimos, saturação,... etc) e perturbações (ruídos presentes, *offset*,... etc) impostas aos mesmos.

Toda a configuração do sistema, resultado de uma tarefa de levantamento e inspeção local à planta (características físicas, sensores e atuadores), pode ser armazenada em arquivo e carregada sempre quando da retomada de operações com a planta física.

Isto também possibilita a criação de uma biblioteca de especificações dos estudos de caso de projetos.

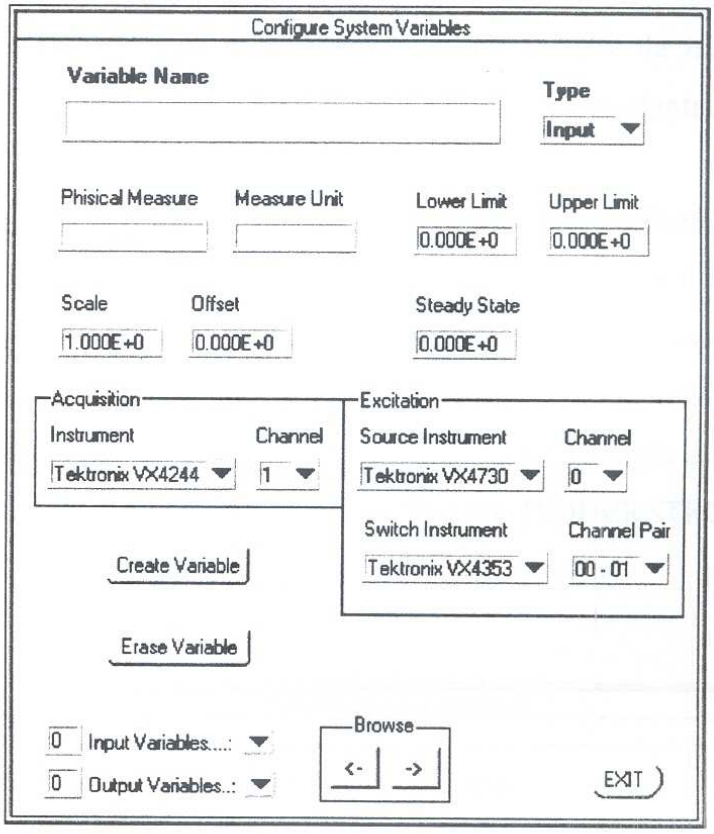


Figura 47 : Interface de configuração – módulo PCVM

Vale relembrar também a abertura de *hardware* que o ambiente permite (possibilitado pela plataforma de instrumentação virtual LabWindows/CVI) e sua configuração através da interface *adaptativa*, técnica explicada no capítulo anterior. Este termo se refere a mudança dinâmica do *layout* da GUI com nova definição dos elementos aparentes em função da plataforma de hardware adotada (VXI ou placa DAQ).

5.1.2 Módulo de aquisição e identificação - PIVM

O módulo PIVM (*Plant Identification VIEnCoD Module*) visa o suporte para que as tarefas de aquisição (excitação da planta e leitura de sua reposta) seja atendidas em todas as suas especificações. São atendidas as fases inicias da metodologia sugerida para Identificação de Sistemas.

Na primeira etapa, o *conhecimento a priori* da planta é necessário para que se tenha informações iniciais sobre a mesma, fornecendo subsídio para escolha do modelo de estrutura mais adequada para o processo da estimação. A aplicação de sinais comuns em controle como degrau, impulso, senóide,... etc e a avaliação da resposta do sistema, possibilitam a obtenção do conhecimento inicial da dinâmica da planta.

Também os métodos de estimação não paramétricas utilizam estes sinais para avaliação precedente à aplicação dos métodos de estimação paramétrica. Nesta, existem sinais apropriados que conduzem o processo de identificação (algoritmos de estimação) a um melhor resultado na obtenção de um modelo. A indisponibilidade destes sinais (PRBS, RBS) na biblioteca de função do LabWindows/CVI exigiu a geração através de algoritmos Matlab (geração de sinais aleatórios binários [SÖDERSTRÖM et al., 1989]) e acesso através de arquivo criado.

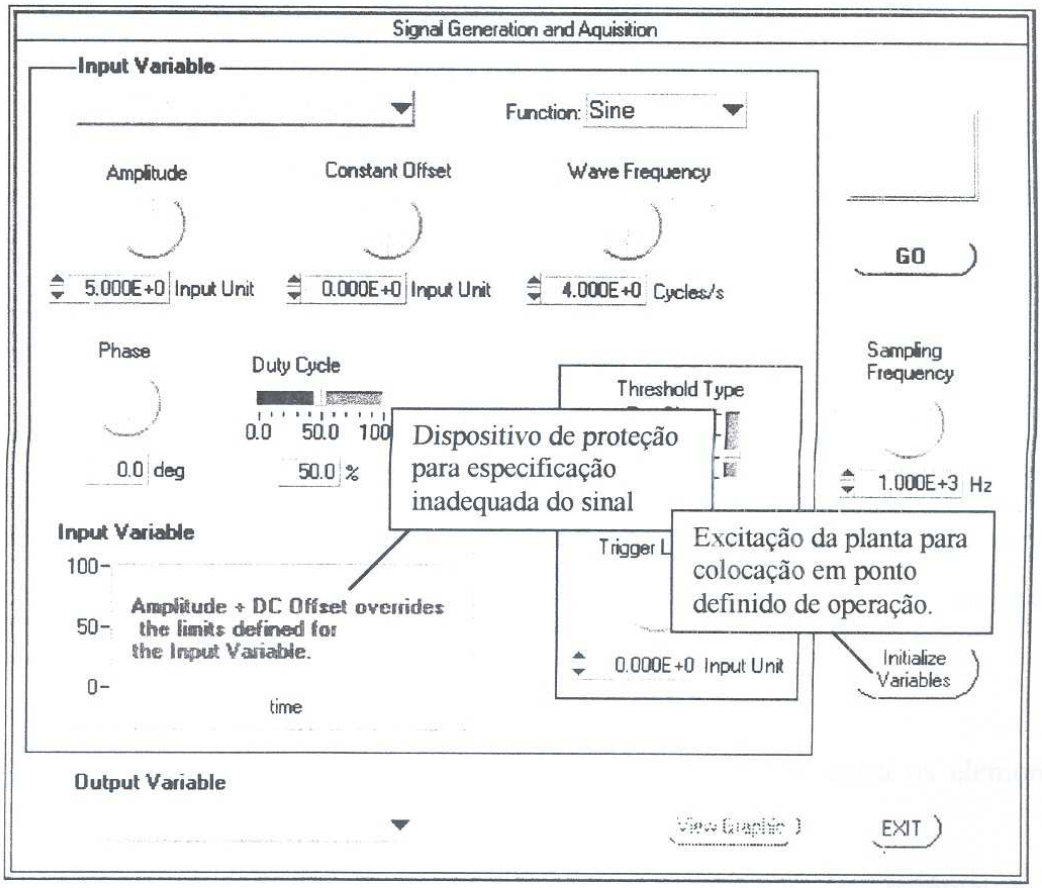


Figura 48 : Interface para processo de aquisição – módulo PIVM

Para qualquer sinal é necessário a manipulação de todos os seus parâmetros que variam conforme o tipo. O resultado desta manipulação em um sinal sugere uma visualização gráfica com atualização automática a qualquer mudança. Neste ponto as técnicas de *interface dinâmica e adaptativa* também se aplicam. Também deve existir dispositivos de proteção à realização de uma aquisição se os parâmetros definidos para o sinal (p. ex. amplitude) extrapolam as especificações de entrada do sistema (definido no módulo PCVM).

Em suma, a interface deve disponibilizar toda a variedade de sinais necessária, permitir fácil manipulação dos parâmetros, possuir recursos gráficos para a pré-visualização do sinal especificado e possuir também um dispositivo automático de verificação das limitações impostas ao sinal. Isto possibilita um processo de aquisição amigável, interativo, e menos suscetível a ocorrência de erros. Estes pontos foram observados no projeto da GUI ilustrada na figura 48.

5.1.3 Módulo de implementação do controlador - CIVM

O módulo CIVM (*Controller Implementation VIEnCoD Module*) propõem-se a dar suporte a implementação do controlador obtido no processo *offline* do CDSC, em simulação com HIL.

Na atual implementação do VIEnCoD uma forma polinomial do controlador otimizado é discretizado para obtenção da respectiva equação a diferenças. Os coeficientes desta equação são inseridos em uma estrutura genérica implementada em C, célula fundamental na composição do módulo CIVM. De forma análoga procede-se com o caso particular da estrutura PID.

A GUI de suporte deste módulo, ilustrada na figura 49 , concentra os elementos necessários para três tarefas principais :

1. *Acesso dos parâmetros de configuração do controlador* : os parâmetros do controlador bem como o intervalo de amostragem utilizado na discretização são automaticamente carregados juntamente com a interface. Para trabalhos isolados

com este módulo (sem vínculo com o CDSC), por exemplo, para verificação de desempenho de outros controladores, é permitida a configuração manual.

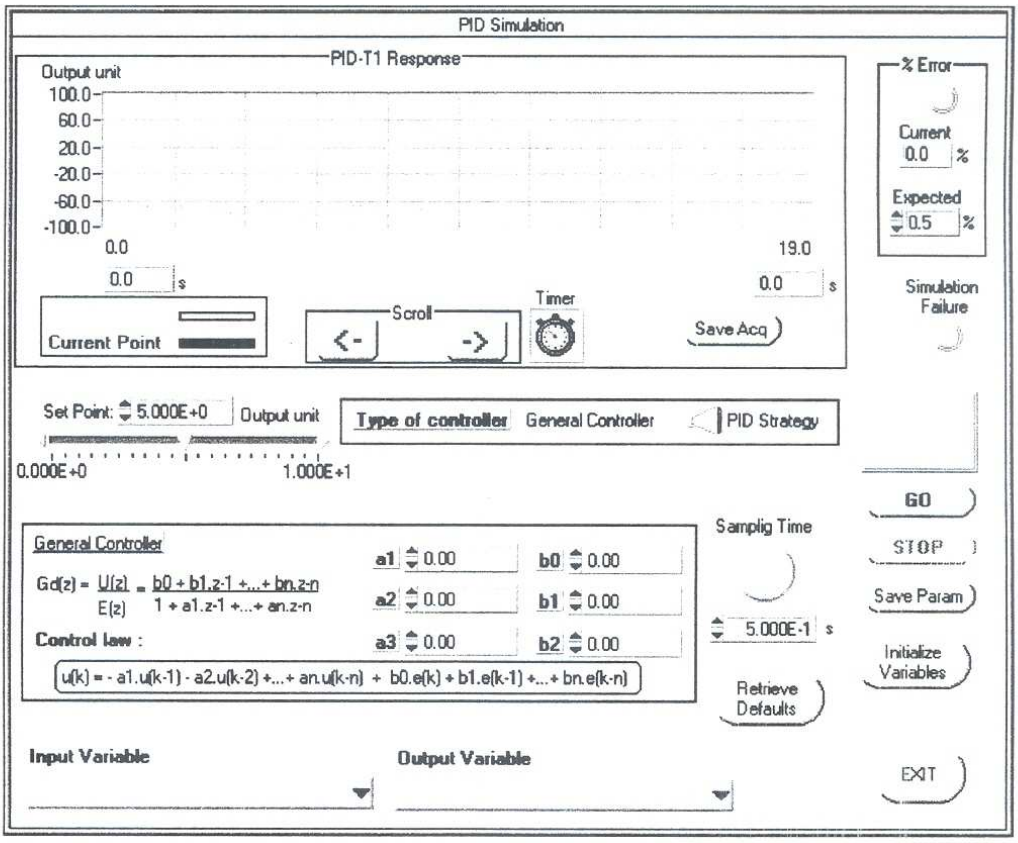


Figura 49 : Interface para processo de simulação com HIL – módulo CIVM

2. *Visualização gráfica e indicação online do comportamento do sistema em processo de execução em tempo real*: a atualização dos dados do sistema na interface são determinados pela estrutura de gerenciamento do sistema operacional pelo RTX dentro dos tempos e interrupções definidas garantindo a operação em tempo real.
3. *Armazenamento dos dados da simulação*: são os vetores do sinal de controle e sinal de resposta do sistema que serão avaliados graficamente e comparados com a resposta obtida em simulação com os modelos.

5.2 Módulos de Suporte ao CDSC - Gerenciador

O módulo gerenciador é a estrutura criada (*m-file*) para integrar de forma lógica e sequencial todos os recursos (*m-files* e módulos ECUVI) que compõe o *toolbox* VIEnCoD. Esta integração é embasada na metodologia do CDSC, onde existe a subdivisão e abordagem de tópicos de sistemas de controle em grupos que representam as diversas etapas no cumprimento de um projeto baseado neste Ciclo.

A estrutura de software deve ser capaz de disponibilizar ferramentas e recursos de forma a permitir o reconhecimento da caracterização dos mesmos dentro do grupo a que pertence (etapa do CDSC) e a visão do contexto em que está inserido (todo o CDSC). A implementação da técnica da *árvore dinâmica,* descrita no capítulo anterior, vem especificamente dar suporte a esta tarefa. O resultado é o projeto da GUI apresentado na figura 32 do referido capítulo.

A composição de grupos explicitada pela interface principal, através de menus e da árvore dinâmica, pode ser dividida basicamente em quatro grupos funcionais :

1. Modelagem e suporte para representação de sistemas;
2. Identificação;
3. Projeto de otimização de controlador;
4. Síntese do controlador – simulação com HIL.

A estruturação do ambiente para o atendimento de cada grupo são colocados a seguir.

5.2.1 Modelagem e suporte para representação de sistemas

A utilização de recursos computacionais para simulação do comportamento dinâmicos de plantas e sistemas requer a sua descrição computacional (modelos) através da adoção de formas de representação diferentes. O suporte a esta tarefa de modelagem é feita pelo Matlab através de estruturas polinomiais, funções e algoritmos. Formas de representação clássicas como Função de Transferência e Espaço de Estados merecem uma atenção maior pela existência de estrutura específica orientada a objetos – a classe LTI,

descrita em capítulo precedente. O encapsulamento destas estruturas em blocos e a interligação dos mesmos estabelecem uma forma de representação gráfica suportada pelo Simulink.

A escolha da forma de representação depende de fatores tais como :

- *A dinâmica da planta determina a melhor estrutura que a representa ou que exija menor esforço para tal.* Sistemas não lineares, por exemplo, encontram facilidades de modelagem através dos recursos (blocos) Simulink. Sistemas MIMO sugerem a representação em espaço de estados,... etc;
- Alguns recursos do Matlab exigem a manipulação de estruturas pré definidas. A ferramenta de análise dinâmica da planta (LTIView), por exemplo, permite apenas estruturas LTI;
- O processo de Identificação determina o uso de estruturas polinomiais específicas, onde são modeladas as influências dos diversos sinais presentes no sistema (excitação, ruído e saída). São diversas estruturas com diferentes composição de polinômios. A figura 50 ilustra o projeto de uma GUI que auxilia a tarefa de criação destes modelos.

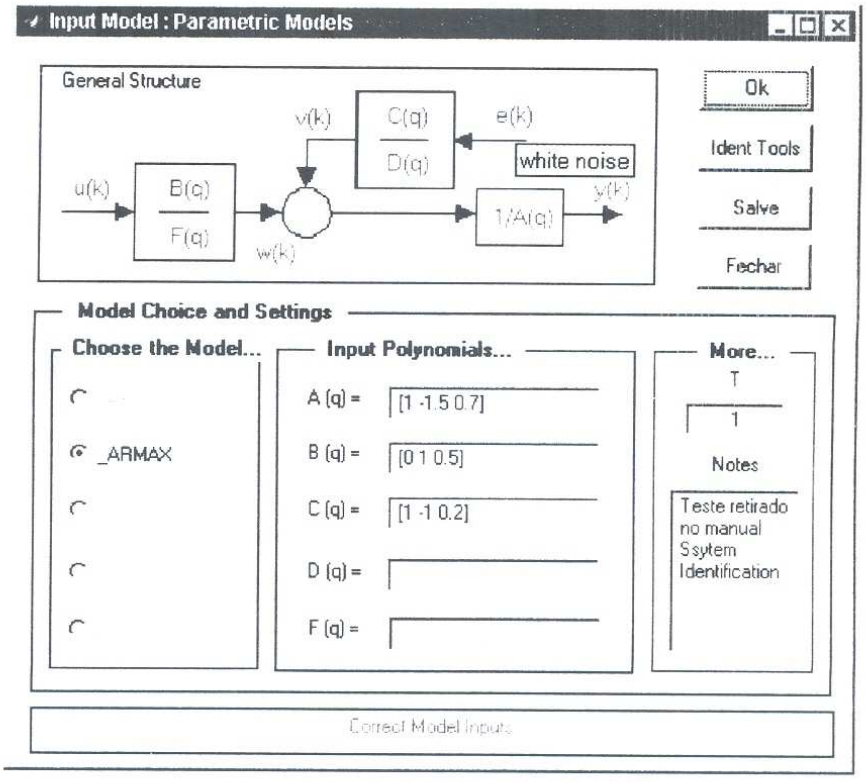


Figura 50 : Projeto de GUI para criação de estruturas paramétricas

- Preferência do usuário em função da facilidade de manipulação e compreensão da estrutura;
- Outros fatores.

Este cenário de possibilidades e necessidades delinearam a implementação de recursos que tornassem a criação e análise de modelos uma tarefa amigável e com adequado suporte. A figura 51 ilustra o esforço de implementação destas necessidades no projeto de uma GUI.

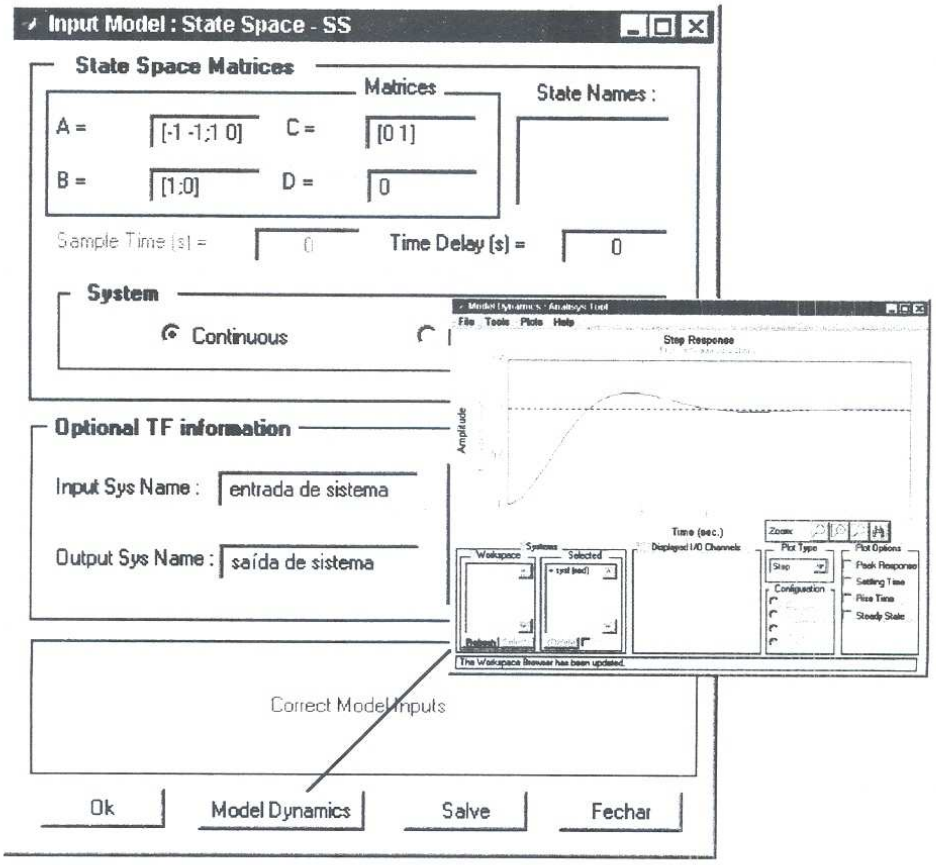


Figura 51 : Projeto de GUI para representação e análise de modelos

Durante o desenvolvimento do CDSC as necessidades de estruturas diferentes comutam sendo necessário conversões. Estas se apresentam de formas diferentes : conversão entre estruturas contínuas (formatos) e conversão entre modelos contínuos e discretos. A figura 52 mostra a GUI responsável por esta tarefa. Todas as possibilidades de conversão são possíveis. Para discretização (no tempo), diferentes métodos são colocados (segurador de ordem zero, aproximação bilinear,... etc). Quando de sua

aplicação é possível o confronto com o modelo contínuo (discretizado) na respostas dos mesmos ao degrau. No projeto desta GUI algumas implementações merecem destaque :

- Todos os modelos da biblioteca do projeto são automaticamente listados e identificados pelos respectivos formatos e dados (período de amostragem, no caso de modelo discreto);
- Simultaneamente à identificação do modelo, os campos de conversão devidos são habilitados e os que não se aplicam à estrutura são inabilitados (esmaecidos) automaticamente;
- A técnica da interface dinâmica também se aplica : da escolha de uma conversão, imediatamente a estrutura resultante é mostrada utilizando-se o formato da GUI da figura 52 onde onde os campos de informação do modelo são repassados e atualizados com informações (método de discretização);

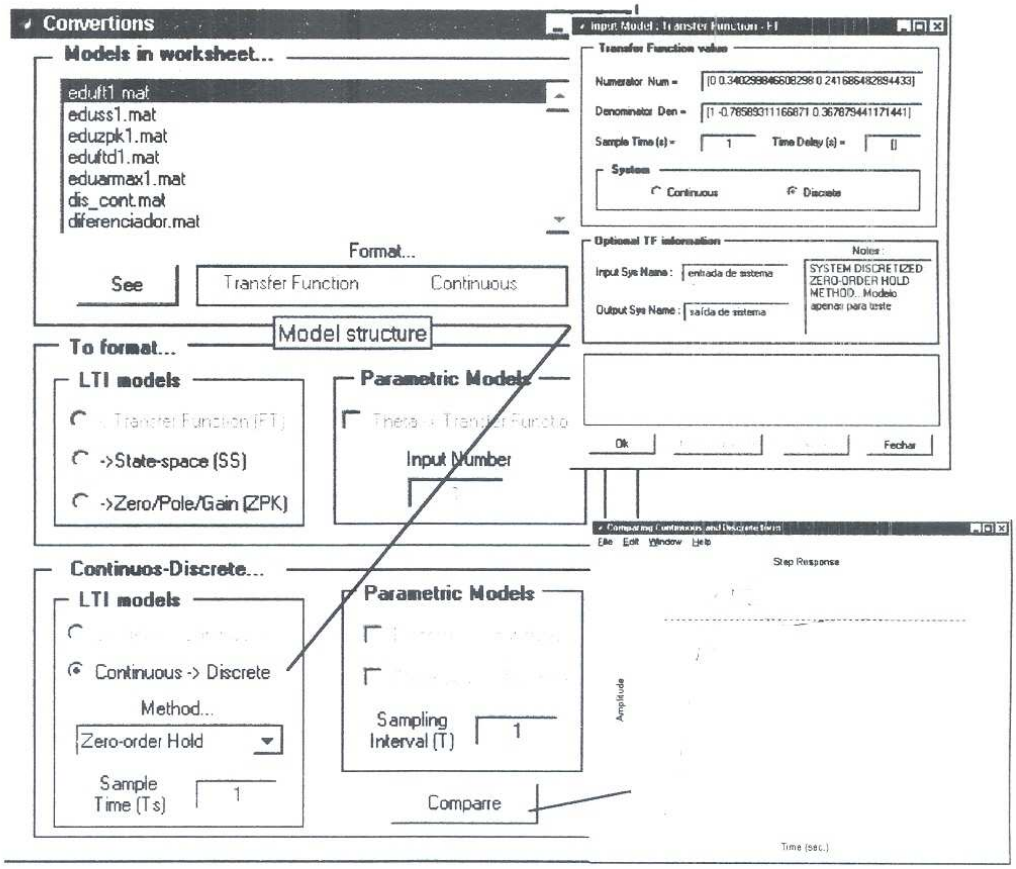


Figura 52 : Projeto de GUI para conversão de modelos - *discretização no tempo*

O objetivo central do CDSC é justamente a obtenção de um modelo matemático de um sistema físico (onde não é possível o levantamento de seus parâmetros físicos) através

do processo de Identificação. Entretanto, esta etapa pode ser dispensada quando já se dispõe do modelo e necessita-se o desenvolvimento de um controlador.

Neste sentido vale ressaltar o aspecto relativo ao ensino de controle, onde em certos casos é necessária a modelagem de um sistema que não existe. Desde que dados experimentais não são possíveis, a modelagem dever ser baseada na teoria e conhecimento a priori.

5.2.2 Identificação

O processo de identificação envolve uma quantidade de tarefas e procedimentos encadeados cuja utilização e encaminhamento dependem muito de conhecimento e experiência. A aplicação de algoritmos especiais à identificação (abordado no capítulo 3) ocorre desde a análise e tratamento dos dados (realização) à validação do modelo estimado.

Este contexto sugere a aplicação da metodologia descrita no capítulo 2 apoiada por recursos computacionais dedicados que nela se baseiam. Estes recursos são representados por dois grupos integrados :

1. *Grupo de suporte de base* : é representado por todos os algoritmos e ferramentas de análise abordados na literatura específica de Identificação [SÖDERSTRÖM et al., 1989] [LJUNG, 1995] [BOX et al., 1994]. A grande maioria destes recursos são encontrados no *toolbox System Identification.*
2. *Grupo de suporte de coordenação* : Ambiente composto por GUIs e suporte gráfico especialmente desenvolvidos para integração dos recursos de base (algoritmos) e disponibilização dos mesmos em ambiente de desenvolvimento amigável.

O desenvolvimento de um ambiente para suporte e coordenação requer um profundo conhecimento teórico de todo o processo de identificação de sistemas onde são necessárias ferramentas que automatizem certas etapas pertencentes a metodologia proposta. Duas principais:

1. Etapa de análise e tratamento dos dados da aquisição : seleção de dados para estimação e validação, filtragem, remoção de tendência lineares,... etc;
2. Etapa de escolha da estrutura de modelo para a estimação: existem inúmeras estruturas de modelos (paramétricos e não paramétricos) cuja escolha se baseia em informações preliminares obtidas experimentalmente ou do conhecimento da planta ou sistema. É um processo que pode conduzir a resultados insatisfatórios exigindo a análise de critérios de validação aplicados ao resultados em processo recursivo. Toda esta abordagem foi suficientemente desenvolvida no capítulo 3.

Tais etapas, em especial a segunda, representam as atividades fundamentais no processo de identificação, onde é requerido conhecimento e experiência para escolha de critérios e análise de cada resultado parcial.

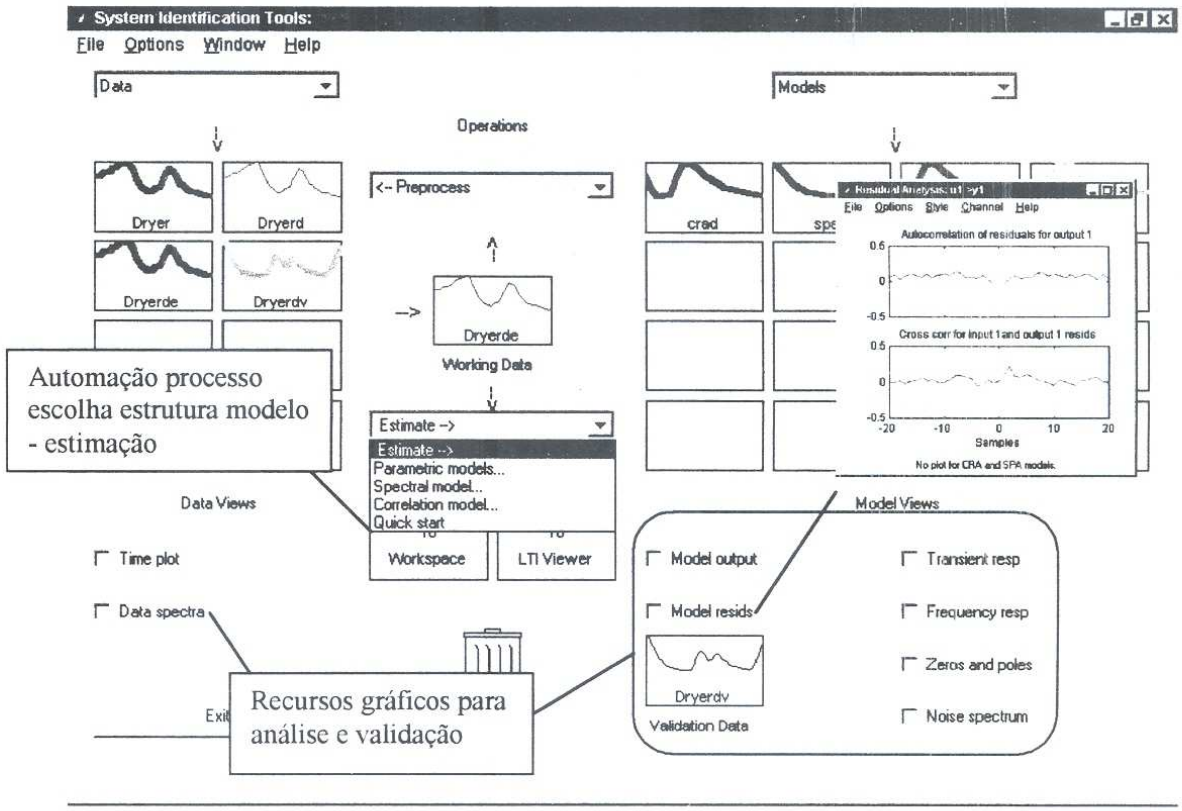


Figura 53 : Suporte do processo de Identificação

O *toolbox System Identification* possui, além dos algoritmos (*m-files - functions*) específicos, um aplicativo (*ident.m*) que proporciona suporte de coordenação integral (grupo 2). Este aplicativo é representado pela GUI ilustrada na 53.

No projeto desta interface todas as etapas do processo de identificação são contempladas, com caracterização da aplicação de metodologia. São disponibilizadas rotinas de automação na execução de etapas (descritas acima), recursos gráficos específicos à análise e interpretação de resultados (p. ex. análise de correlação) e dados relativos a critérios de validação.

Desta forma, foi desnecessário o desenvolvimento de recursos para esta atuação sendo feito apenas uma adequação à estrutura funcional proposta pelo VIEnCoD. Esta estrutura, relativa à etapa de identificação, é implementada através da formação do menu "*Identification*" (e da árvore dinâmica) da GUI principal.

A adequação do aplicativo principal *ident.m,* corresponde a dispositivos integrados para execução das seguintes tarefas:

- Leitura, visualização e manipulação dos dados (realizações) obtidos na etapa de aquisição pela ECUVI;
- Geração de sinais de excitação adicionais para o módulo PIVM, por exemplo,. PRBS;
- Simulação de uma realização para utilização e estudo das ferramentas de identificação. Aqui atende-se propósitos educacionais e familiarização dos conceitos e aplicabilidade da teoria de identificação.

Na simulação de uma realização é necessária a definição e configuração dos sinais de entrada (excitação), ruído presente e modelo da planta. Esta última requer uma estrutura paramétrica construída através das ferramentas de modelagem abordadas.

A integração dos dispositivos e o atendimento destas tarefas são feitas pela GUI ilustrada na figura 54. Nesta, algumas implementações merecem destaque:

- Aplicação da técnica da interface dinâmica onde na escolha de cada sinal atualizam-se os campos de configuração de parâmetros;
- Todos os campos são suportados por *hints* adaptativos que fornecem informações sobre o tipo de sinal, sintaxe de configuração do parâmetro,... etc;
- Orientação interativa no uso das ferramentas pela barra de mensagem (*message bar*).

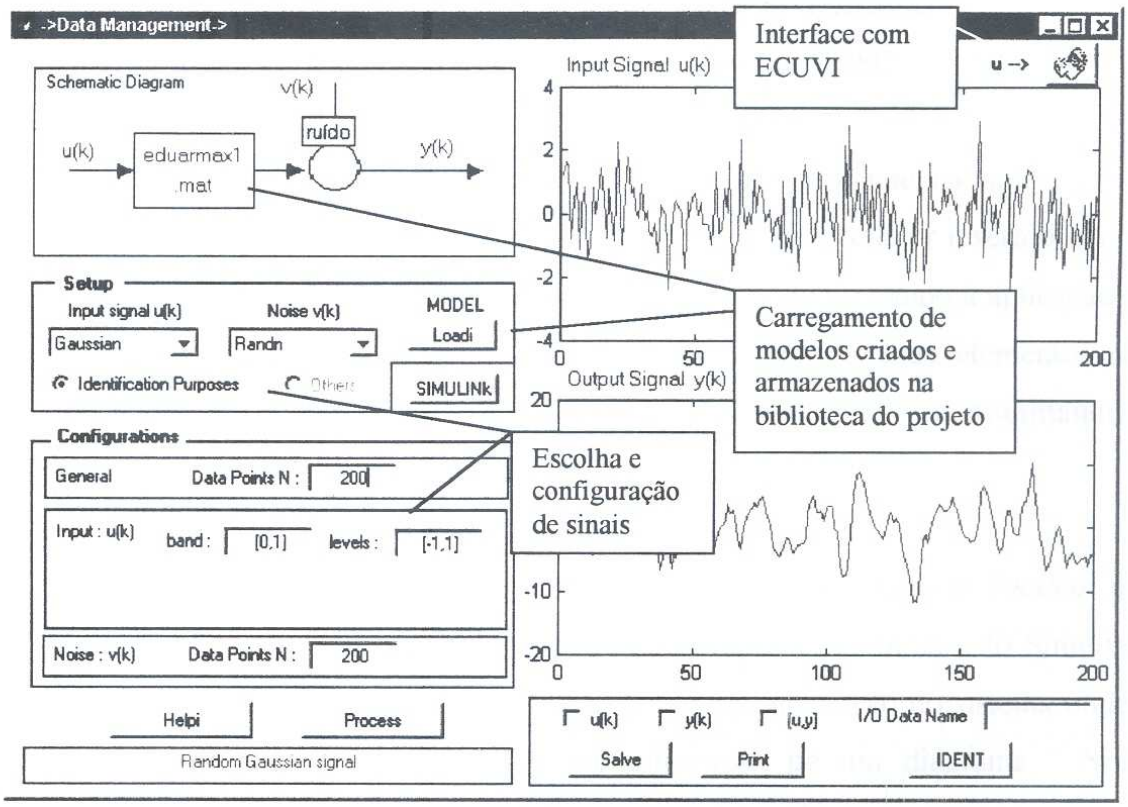


Figura 54 : Suporte funcional à etapa de Identificação

5.2.3 Projeto e otimização do controlador

Na sequência do CDSC atinge-se a etapa de projeto do controlador e otimização do controlador. O *toolbox* VIEnCoD prevê basicamente duas ferramentas para síntese do controlador que representam processos diferentes de projeto, sendo uma delas apenas estabelecida como secundária e de apoio.

Foram estudados os recursos Matlab com a preocupação em encontrar ferramentas que apresentassem um adequado grau de automação para projeto de controladores (adaptando-se ao perfil do ambiente) e que contemplassem as abordagens convencionais (proporcionando objetivos de ensino de controle). O *Control System Toolbox* , em sua última versão (versão 5.2), apresenta um aplicativo para suporte de projeto baseado no lugar das raízes – o *Root Locus Design* (*rltool.m*). O projeto de GUI utilizado se assemelha ao perfil adotado sendo viável sua integração ao ambiente, da mesma forma com o ocorrido na etapa de Identificação. As limitações impostas por esta técnica (p. ex.

apenas sistemas LTI) e a interação necessária com o usuário determina esta ferramenta como auxiliar na estrutura do *toolbox* e não de suporte direto ao CDSC.

A concepção fundamental da estrutura do VIEnCoD baseia-se na aplicação de algoritmos de otimização para sintonia dos parâmetros de um controlador inserido em um sistema em malha fechada, baseado em critérios de desempenho no tempo à aplicação de sinais de excitação específicos. Esta concepção permite a presença de elementos não lineares no sistema e a implementação futura de rotinas recursivas possibilitando a aplicação de estratégias adaptativas operando em tempo real.

O suporte desta concepção baseia-se nos recursos do *Optimization Toolbox* e o *NCD blockset*, ambos direcionados à representação de digrama de blocos do Simulink. Isto moldou a filosofia de projeto onde é necessário a implementação do sistema onde a planta (identificada) está inserida, através da construção de um diagrama. Neste diagrama, todos os elementos físicos existentes ou previstos devem estar representados através de seu modelos encapsulados em blocos. Esta forma de representação permite melhor acesso ao fluxo de sinais presente no sistema.

Portanto, este processo envolve a definição e construção de diagramas e modelos que compõem o sistema determinando um conjunto de tarefas individuais. Este perfil sugere novamente o desenvolvimento de recursos que orientem o processo de forma intuitiva e amigável deixando transparente sintaxes e características operacionais do Matlab e Simulink; este com maior destaque onde exige-se o conhecimento de bibliotecas (blocos de sinais, conexão,... etc) além da manipulação e construção de diagramas.

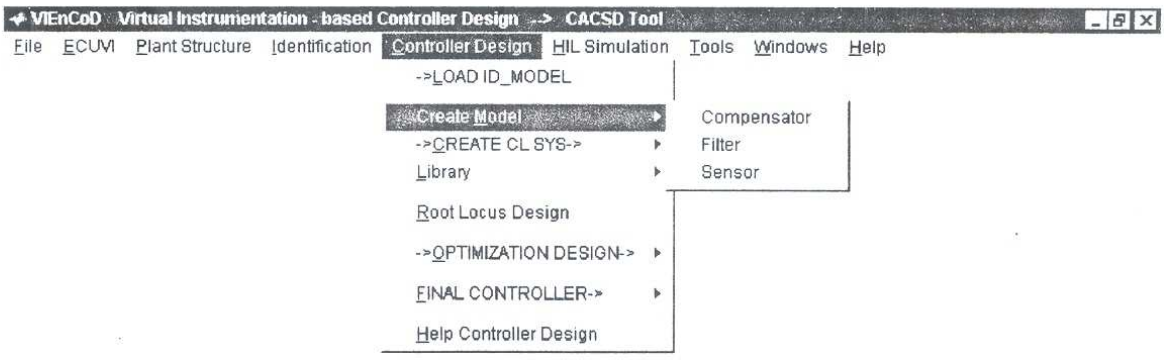


Figura 55 : Suporte funcional à etapa de Projeto e Otimização de controlador

Os recursos se apresentam na estrutura dos menus e aplicativos de suporte. A figura 55 ilustra a implementação desta etapa do CDSC com a contemplação dos pontos acima colocados. Vale aqui ressaltar que os menus com a indicação sob setas "→ ***nome submenu***→" representam o caminho principal (natural) do CDSC.

São previstas rotinas para criação dos modelos comumente presentes no sistema operando com a planta, como sensores e filtros e principalmente o controlador. Para estes elementos ou qualquer outro que se apresente, os mesmos aplicativos descritos para construção dos modelos de plantas são utilizados.

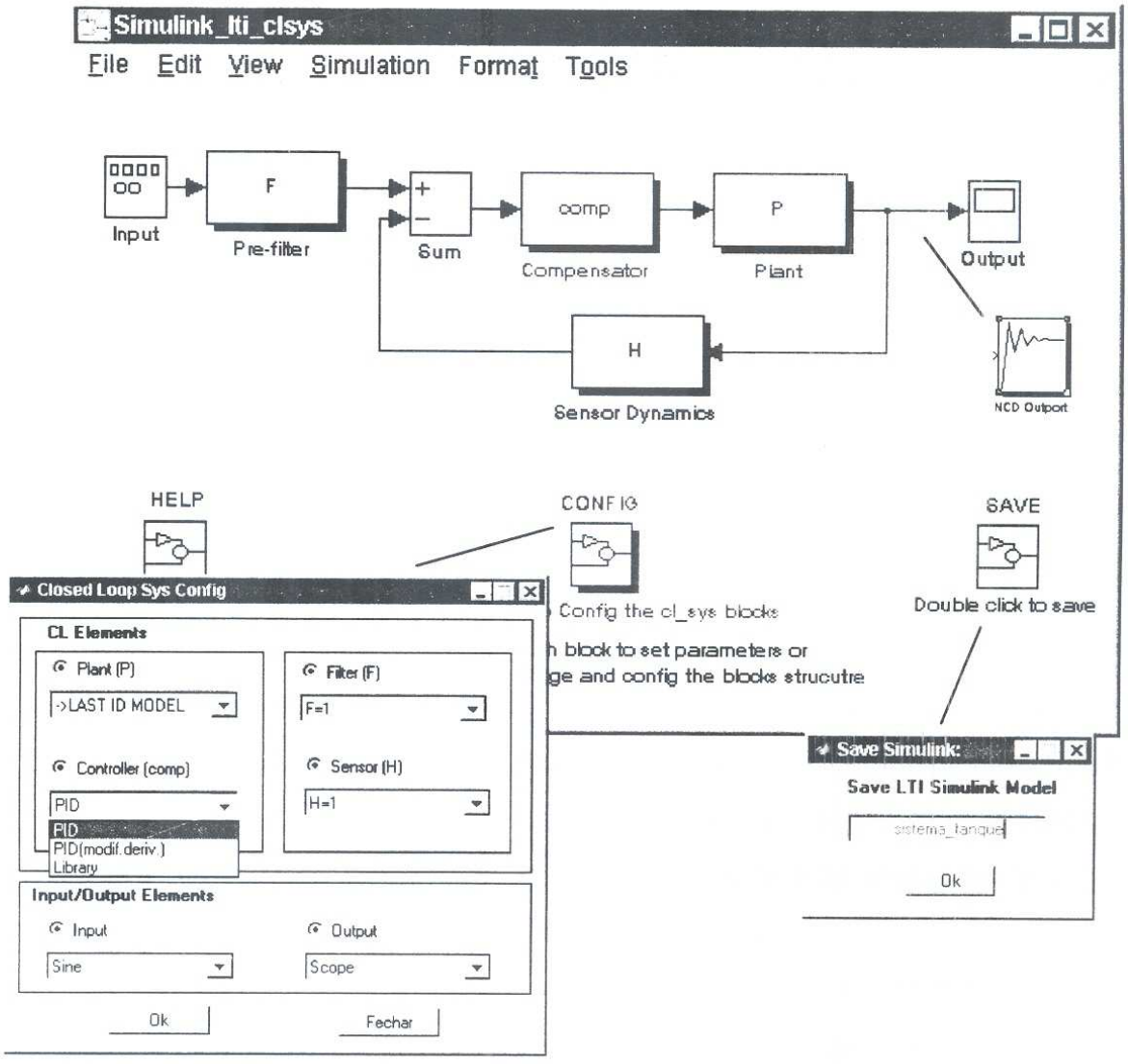


Figura 56 : Interface para configuração do sistema em MF

O ambiente suporta a criação do sistema em malha fechada de duas maneiras : através da existência de estrutura *single loop* previamente concebida ou estrutura aberta à definição do usuário. Ambas apresentam elementos (blocos) que encapsulam rotinas (*m-files*) de apoio à construção. A figura 56 mostra as GUIs correspondentes a estas rotina e a estrutura *single loop* que representa um arquivo mdl funcionando como interface para construção do sistema. As tarefas inerentes a esta interface são :

- Configurar os blocos componentes do sistema através do acesso às bibliotecas existentes (pasta do projeto): modelo da planta (planta identificada ou criada), modelo (estrutura) do controlador (estruturas pré concebidas, por exemplo, formas PID ou outras – criadas pelo usuário), modelos do filtro e sensor (criados pelo usuário), sinais de excitação e definição da saída do sistema (para arquivo, análise – *scope* ou bloco NCD – *blockset*);
- Rotinas de apoio à configuração acima, para armazenamento (biblioteca) e estrutura de *help*.

A atuação da interface de configuração dos elementos do diagrama é basicamente, após a seleção dos elementos desejados, promover a substituição ou reconfiguração dos blocos no diagrama, de forma automática sem manipulação operacional do Simulink.

Após a estruturação do sistema e armazenamento (nome.mdl), este é adaptado automaticamente para utilização das ferramentas de otimização. Vale ressaltar a possibilidade de simulação prévia do sistema. O bloco NCD (*Nonlinear Control Design*) é conectado na saída do sistema e encapsula um ambiente completo de suporte a otimização gerenciado por uma GUI, ilustrada na figura 57. Conforme a filosofia adotada, adotou-se este recurso existente que atende por completo as necessidades impostas. O aplicativo se baseia na aplicação de algoritmos de otimização abordados no capítulo 3, que trabalham em função de especificações que envolvem:

- Definição das características (parâmetros no domínio do tempo) do sinal de excitação do sistema (degrau – tempo de subida, sobressinal, tempo de acomodação,... etc);
- Declaração dos parâmetros a serem otimizados no sistema (do controlador cuja estrutura fora definida previamente);
- Configuração dos parâmetros de otimização.

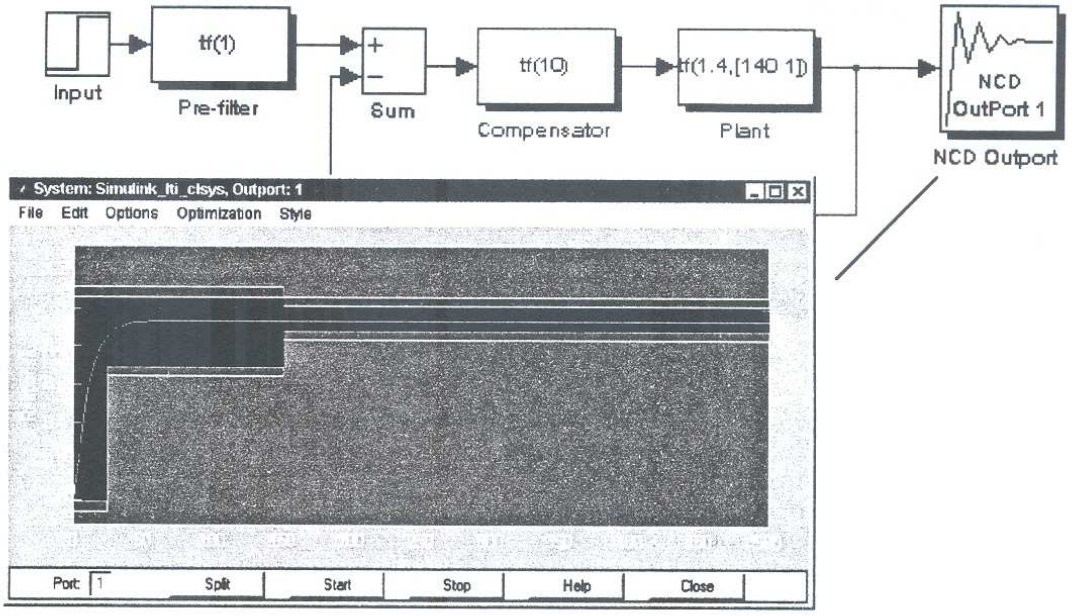


Figura 57 : Interface para processo de Otimização do controlador

Obtidos os parâmetros do controlador, é necessário a manipulação dos mesmos de forma a se adequar às condições impostas pela etapa de simulação com HIL. Todo o processo acima foi realizado com estruturas baseadas no domínio do tempo, incluindo o modelo do controlador. Portanto, os parâmetros obtidos são relativos ao seu modelo contínuo.

A metodologia aplicada é a realização do projeto do controlador no domínio do tempo (planta e demais elementos com seus modelos contínuos). O modelo contínuo resultante é discretizado e obtido a equação a diferenças que descreve seu comportamento (lei de controle). Este processo, referenciado em [BARCZAK, 1995] [OGATA, 1995] é suportado pelo ambiente que utiliza algoritmos de discretização e determina a configuração e implementação computacional na ECUVI através do módulo CIVM. Os parâmetros resultantes são carregados automaticamente na GUI do referido módulo, figura 49.

5.2.4 Implementação do controlador – simulação com HIL

Esta etapa corresponde a realização de procedimentos de análise e avaliação de desempenho do sistema com o controlador otimizado através de simulações. Estas se

apresentam de duas formas diferentes : simulação *offline* do sistema onde todos os elementos são modelos, e a simulação com HIL onde o controlador é implementado pela ECUVI no controle do sistema físico. A gerência da simulação com HIL é feita pelo módulo CIVM, descrito anteriormente.

A simulação *offline* é feita através do diagrama utilizado na otimização onde o bloco controlador agora é substituído por seu modelo discreto. O ambiente de simulação Simulink suporta a presença de elementos contínuos e discretos configurando os chamados sistemas híbridos [The MathWorks, 1997d]. Nestes, as saídas dos elementos discretos são mantidos constantes entre os tempos de amostragem.

Este sistema se aproxima do sistema formado na simulação com HIL, fornecendo portanto, dados para confronto com os resultados obtidos nesta simulação para o processo de validação. Uma interface gráfica é sugerida e ilustrada na figura 58. Ela se baseia na visualização gráfica de ambas as repostas para avaliação de desvios acentuados na região transitória ou em regime permanente. Critérios de validação numéricos (p. ex. MSF – *Mean Square Fit*) podem ser aplicados auxiliando este processo.

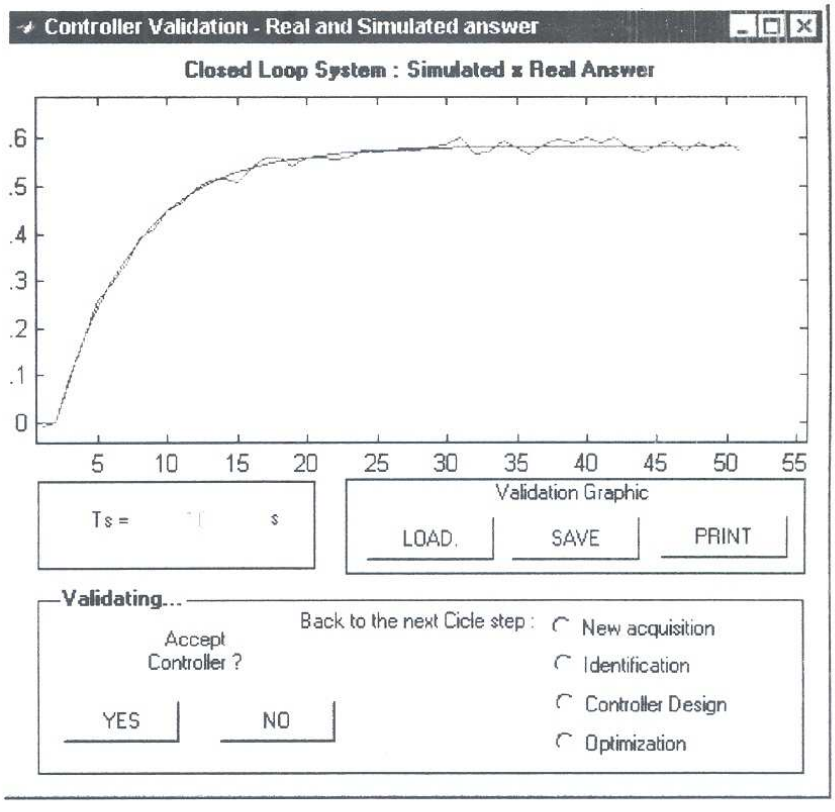


Figura 58 : Interface para processo de validação

O controlador sendo considerado aceitável é transferido para um controlador industrial manualmente ou via implementação feita pelo ambiente de instrumentação virtual. O aval final é estabelecido na interface, armazenando (biblioteca do projeto) e imprimindo resultados. O ciclo é fechado ou retorna-se às etapas posteriores na varredura de erros ou desvios ocorridos.

5.3 Comentários

A concepção estrutural e funcional do *toolbox* VIEnCoD se baseia-se na gerência de recursos existentes do Matlab e Simulink através de *m-files* desenvolvidos para integração e disponibilização dos mesmos, deixando transparente as sintaxes envolvidas.

A estrutura de software (m-files) bem como o projeto das diversas GUIs, foi realizado modularmente em função das diferentes etapas do CDSC.

Em todas as etapas houve a participação de aplicativos existentes já dotados de GUIs cujas ferramentas atendiam todas as necessidades técnicas de cálculos e análise para uma determinada tarefa. As implementações realizadas neste sentido foram focadas em adaptações destas estruturas (ou tentativa, conforme visto no capítulo 4) e inserção das mesmas no ambiente CACSD do VIEnCoD.

Este mesmo esforço ocorreu na integração com os módulos da ECUVI, gerenciados pelo ambiente visual totalmente desenvolvido no Matlab. Em todas as GUIs desenvolvidas houve a preocupação em tornar explícito o objetivo e funcionalidade de cada tarefa dentro do contexto da metodologia proposta. Isto foi conseguido com : estruturação da interface principal na modularização dos menus e implementação da técnica da *árvore dinâmica*; projetos de *layout* de GUIs para as tarefas específicas com adoção de técnicas especiais que determinaram, dentre outros pontos, a manipulação dos conceitos de controle de forma intuitiva e de orientação.

Capítulo 6

6 Utilização do VIEnCoD – Estudo de Caso

O VIEnCoD, conforme visto no presente trabalho, reúne um conjunto de recursos que visam dar suporte a todo CDSC. Cada fase, desde a Modelagem e Identificação à Síntese do Controlador, encontra suporte operacional específico de forma a atender aos seus objetivos e com a preocupação da integração funcional com todo o CDSC.

Um projeto de desenvolvimento de um controlador para uma planta (estudo de caso) é avaliado neste capítulo, de forma a verificar o atendimento das necessidades e objetivos de cada fase do CDSC bem como do ambiente como um todo. Os parâmetros físicos da planta são conhecidos permitindo uma avaliação mais profunda sobre os resultados dos recursos de análise e projeto. Os conceitos de simulação com HIL, RCP e condições de tempo real são verificados pela utilização da ECUVI através de seus módulos.

6.1 Descrição do Estudo de Caso – Sistema de Controle de Nível de Líquido

Um sistema de controle de nível de líquido é utilizado para estudo de caso através do protótipo ilustrado na figura 59. Este sistema é comumente encontrado nos processos industriais formando uma célula básica no treinamento dos engenheiros de controle, além de sua abordagem nos conteúdos curriculares do cursos de engenharia e pós-graduação.

O projeto do experimento prevê a utilização da configuração de *hardware* da plataforma de instrumentação *VXIbus*, cuja descrição se encontra no Anexo I deste trabalho.

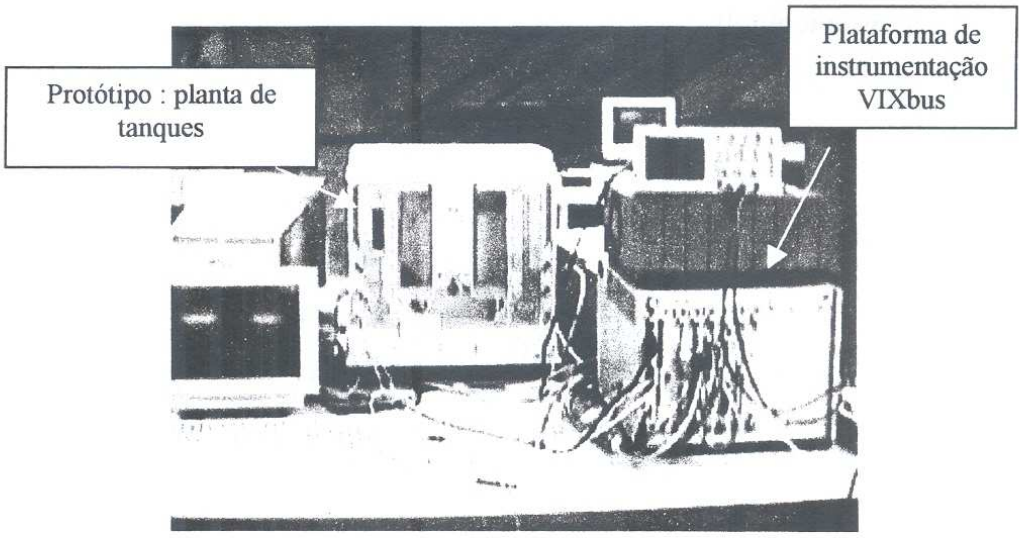


Figura 59 : Estudo de caso : planta e *VXIbus*

6.1.1 Modelo matemático e características dinâmicas

A figura 60 ilustra o sistema de um tanque de líquido com um vazão de entrada controlada por um bomba e a vazão de saída controlada por uma válvula. O conhecimento das variáveis e parâmetros do sistema e aplicação das leis físicas permite a obtenção de um modelo matemático. No caso, tal modelo relaciona a altura do líquido no tanque (**H**) com a vazão de entrada ($\mathbf{Q}_i$).

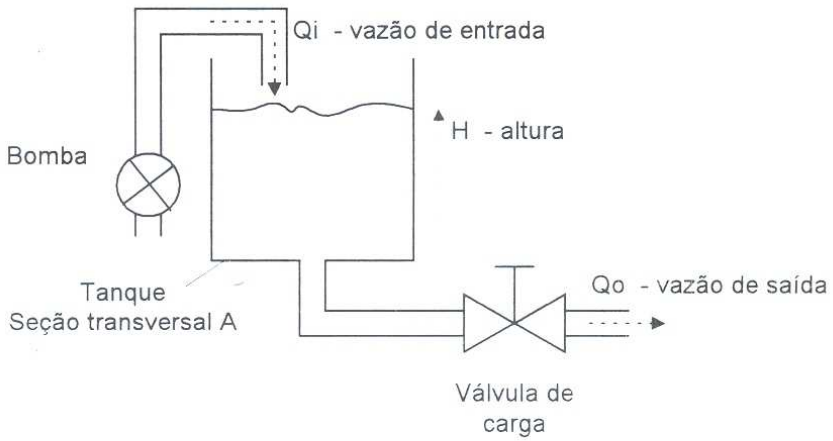


Figura 60 : Representação tanque

O modelo do sistema é determinado pelo fluxo de entrada no tanque ($\mathbf{Q}_i$) pelo fluxo de saída pela válvula ($\mathbf{Q}_o$). Tem-se, portanto:

$Q_i - Q_o$ = *Taxa de variação no volume de fluido no tanque*

$$Q_i - Q_b = \frac{dV}{dt} = A\frac{dH}{dt} \tag{6.1}$$

Onde:

A = área da seção transversal do tanque

V = volume de fluido

$\mathbf{Q_i}$ = taxa de fluxo de entrada devido a bomba

$\mathbf{Q_o}$ = taxa de fluxo de saída pela válvula

Se a válvula comporta-se, por hipótese, como um orifício em *canto vivo*, o fluxo através da válvula será relacionado com o nível do fluido no tanque (**H**) através da expressão:

$$Q_o = C_d.a.\sqrt{2g.H} \tag{6.2}$$

Onde :

a = área da seção transversal do orifício. Na prática isto representa as dimensões da válvula e o canal de fluxo no qual está montada. Dado que esta dimensão muda ao longo do comprimento do canal, **a** deve representar o valor médio.

$\mathbf{C_d}$ = coeficiente de descarga da válvula. Este coeficiente leva em conta todas as características do fluido, perdas e irregularidade no sistema tal que os dois lados da equação são balanceados.

g = constante gravitacional (9,8 m/s^2).

Esta última relação assume $\mathbf{C_d}$ constante e, portanto, que $\mathbf{Q_o}$ é proporcional a raiz quadrada do nível **H** para todas as condições de operação possíveis. Em uma válvula prática (como a utilizada no protótipo) a taxa de fluxo $\mathbf{Q_o}$ será uma função não linear geral do nível **H**, $Q_o = f(H)$. Combinando esta relação com a anterior tem-se o modelo matemático que descreve o comportamento do sistema :

$$A\frac{dH}{dt} + f(H) = Q_i \tag{6.3}$$

Esta é uma equação diferencial de primeira ordem relacionado a taxa de fluxo de entrada, Q_i, com o nível de fluido no tanque, **H**. De forma a tornar esta relação aplicável a

proposta de sistemas de controle, deve-se linearizar a equação considerando pequenas variações sob um ponto de operação desejado de nível do fluido no tanque, ou seja :

$$H = H' + h$$

Onde : H' = nível de operação normal - é uma constante

h = pequena variação em torno do nível

Portanto, para pequenas variações de **h** sobre **H'**, pode-se aproximar a função f(**H**) pela linha tangente a **H'**, conforme mostra a figura 61.

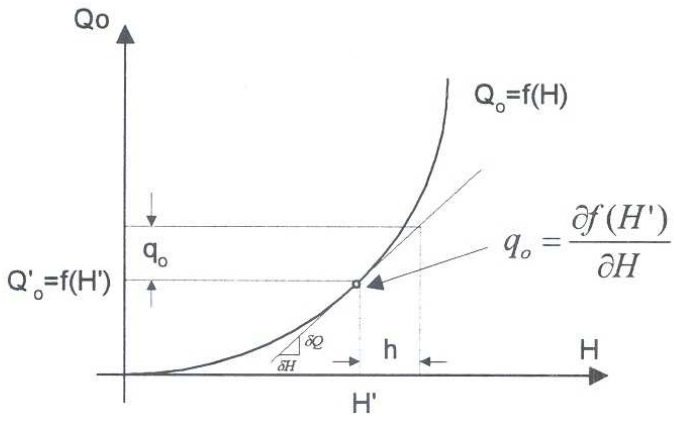


Figura 61 : Relação não linear de Qo e H

Dado que o fluxo de entrada $\mathbf{Q}_i$ consiste de um componente constante $\mathbf{Q}_i'$ mais uma pequena componente variável $\mathbf{q}_i$, então se $\mathbf{Q}'_o$ é o estado constante de fluxo de saída correspondente a **H'** e $\mathbf{q}_o$ é a pequena variação no fluxo de saída correspondente a **h**, a equação 6.3 pode ser escrita como :

$$A\frac{dh}{dt} + Q_o' + q_o = Q_i' + q_i \qquad (6.4)$$

Com referência a figura 61 tem-se que :

$$A\frac{dh}{dt} + f(H') + h.D = Q_i' + q_i \qquad (6.5)$$

Onde o coeficiente é a inclinação das características da válvula ao nível **H'**. Assim :

$$D = \frac{\delta f(H')}{\delta H} \tag{6.6}$$

Quando o nível é constante, com $\mathbf{q}_i$=0 e **h**=0, então a equação 6.5 fornece a relação em regime permanente para fluxo e nível :

$$f(H') = Q'_i \tag{6.7}$$

Subtraindo a equação 6.7 da equação 6.5 e desenvolvendo obtém uma equação diferencial linear de primeira ordem para um sistema de tanque simples :

$$T.\frac{dh}{dt} + h = k.q_i \tag{6.8}$$

A sua transformada de Laplace fornece a função de transferência do sistema :

$$h(s) = \frac{k}{Ts+1} q_i(s) \tag{6.9}$$

Onde : **T** = constante de tempo do sistema dado por $T = A/D$

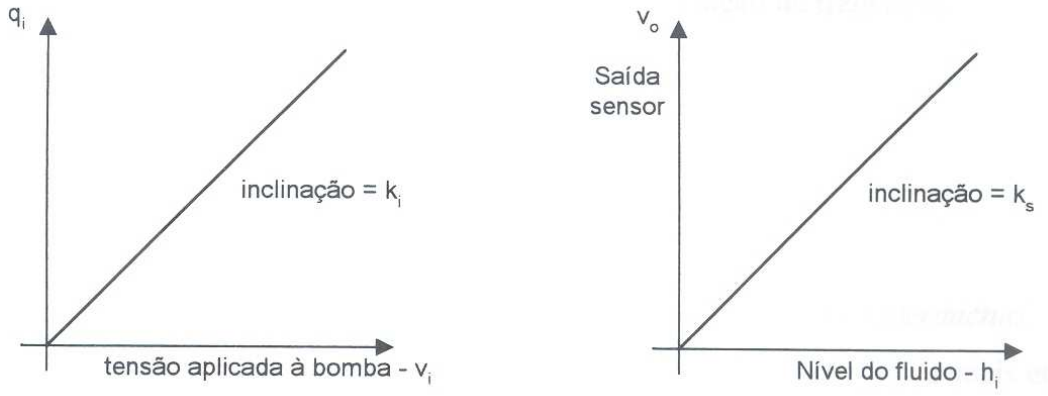
$\mathbf{k} = D^{-1}$

D = inclinação da curva avaliada em H'

A = área da seção transversal do tanque

6.1.1.1 Influência do sensor e atuador

Para análise da planta em um sistema de controle deve-se levar em conta a presença de sensores e atuadores que representam funções de transferência adicionais na malha de controle. No protótipo utilizado, a taxa de fluxo de entrada $\mathbf{q}_i$ é controlada pelo ajuste de tensão aplicada a um amplificador do motor da bomba, $\mathbf{v}_i$. Da mesma forma, o nível de líquido é obtido através de um transdutor de pressão que fornece uma tensão de saída, $\mathbf{v}_o$, que é proporcional ao nível de líquido **h**. A característica da bomba na relação $q_i=f(v_i)$ é tido com linear e ilustrado na figura 62a. Da mesma forma, a característica do transdutor na relação $v_o=f(h)$ conforme mostra a figura 62b.



(a) Característica da bomba (b) Característica sensor

Figura 62 : Relações lineares de sensor e atuador

As funções de transferência do sensor e bomba são avaliados apenas como ganhos constantes $\mathbf{k_s}$ e $\mathbf{k_i}$ respectivamente, levando as seguintes relações:

$$q_i = k_i \,.\, v_i \qquad\qquad v_0 = k_s \,.\, h$$

A função de transferência do sistema, equação 6.9, fica :

$$v_o(s) = \frac{G}{Ts+1} . v_i(s) \tag{6.10}$$

Onde : **G** = ganho do sistema = $k_i \,.\, k_s \,.\, k$

6.1.2 Identificação

Previamente ao processo de aplicação de sinais específicos à Identificação realizou-se o levantamento da resposta da planta à aplicação de um degrau. O valores obtidos são colocados a seguir:

- A aplicação de 5 V à bomba corresponde a aproximadamente metade da taxa de fluxo de vazão de entrada Qi (1,7 l/min) sendo que o nível do tanque atingiu 63% do valor final (170 mm) em 140 s. Isto para válvula de saída na metade de sua abertura. O valor de tensão indicado pelo sensor de nível no valor final (em

regime) foi de 7 V. Baseado na equação 6.10, a função de transferência da planta (sistema de 1ª ordem) fica :

$$FT(s) = \frac{1.4}{140s + 1}$$

Este conhecimento a priori agrega facilidades no *projeto do experimento,* etapa inicial na metodologia descrita no capítulo 3 deste trabalho, bem como nas demais etapas. Por exemplo, o conhecimento da constante de tempo da planta permite uma melhor configuração das características dos sinais de excitação (p. ex. definindo o período de amostragem no processo de aquisição - Ts, aproximadamente a 10ª parte da constante de tempo da planta em malha fechada [SÖDERSTRÖM et al., 1989]).

O conhecimento obtido com a resposta ao degrau do sistema também permite um direcionamento mais pontual nos recursos de análise e algoritmos de estimação disponíveis. Neste caso, a ferramenta de identificação adotada pelo VIEnCoD permite uma busca automática da melhor estrutura de modelo e estimador associado (vide capítulo 3 deste trabalho) baseado em procedimentos iterativos na escolha da ordem e parâmetros.

As condições para o experimento são feitas através da configuração dos sinais e período de amostragem pela GUI do módulo PIVM.

A aquisição (vetor de excitação e resposta - [u(k),y(k)]) que forneceu melhores resultados ao processo de identificação foi baseada em um sinal PRBS de excitação com limites superiores e inferiores de 3V e 7V respectivamente, aplicado à planta operando em regime à 5V. Os recursos do LabWindows/CVI não contemplam este sinal sendo portanto, gerado diretamente do ambiente Matlab e carregado na ECUVI. O intervalo de amostragem utilizado foi de 10 segundos. Houve a preocupação em tornar os tempos dos estados do PRBS (limite superior e inferior) longos o suficiente de forma a excitar a planta em sua faixa de freqüência de destaque, ou seja, em baixas frequências. A figura 63 mostra a resposta do sistema ao PRBS e também o pré tratamento dos dados (retirada da média, tendências lineares, separação dados para estimação e validação,... etc) em preparação a aplicação dos algoritmos de estimação.

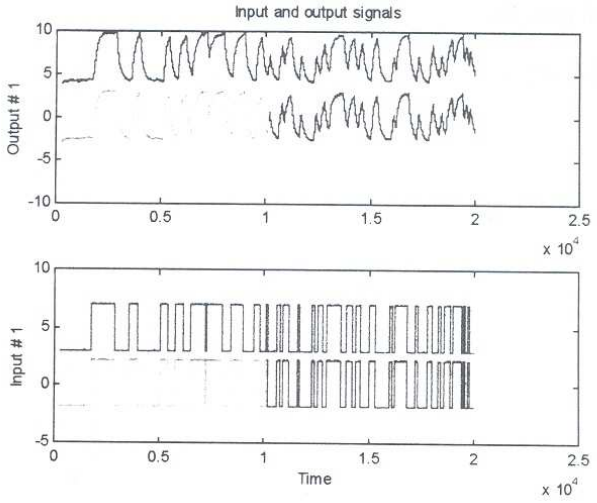


Figura 63 : Resultado da aquisição : aplicação de PRBS

Como já existe um conhecimento a priori da planta, modelo de 1ª ordem, utilizou-se um modelo ARX ([na=1 nb=1 nk=1]) e ARMAX ([na=1 nb=1 nc=1 nk=1])de ordem 1. Os resultados foram bastante satisfatórios, igualando-se aos métodos determinados automaticamente pelo aplicativo de identificação. Os resultados da aplicação dos algoritmos de estimação forneceram algumas informações:

- Pela análise da autocorrelação confirma-se as características de ruído branco do sinal gerado PRBS. A correlação cruzada mostra que não houve nenhum pico fora do intervalo de confiança indicando que toda a dinâmica da planta foi abordada na identificação. Vide figura 64.

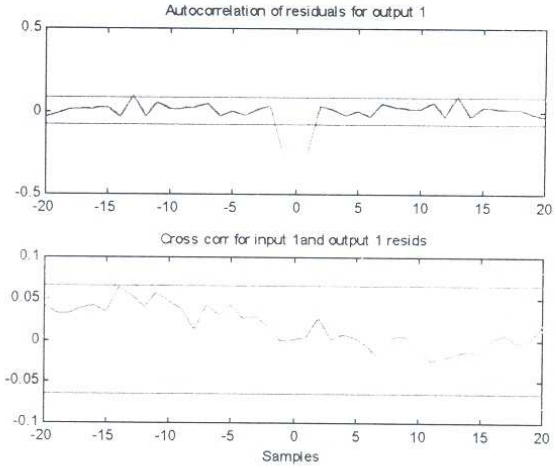


Figura 64 : Análise dos correlações

- O confronto com os dados reais (dados para validação) intensifica ainda mais a qualidade do modelo estimado. A aplicação do critério MSD (*Mean Square Fit*) confirma este resultado. Vide figura 65.

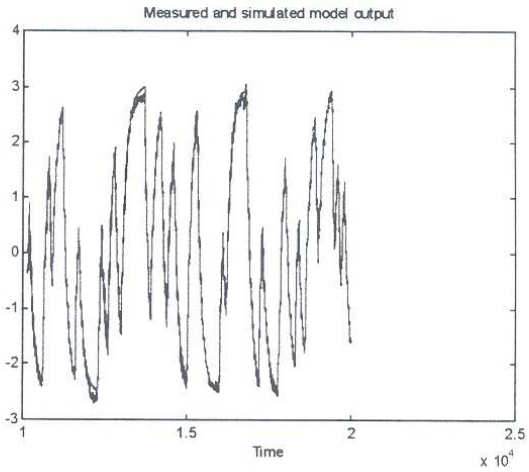


Figura 65 : Saída real x simulada

Em suma, o modelo obtido foi :

$$FT(s) = \frac{1.37}{138s + 1}$$

Que se aproxima muito do modelo levantado com a aplicação do degrau. O processo de validação do modelo é muito importante pois a qualidade da representação do sistema físico pelo modelo determina maior ou menor confiabilidade na próxima etapa do CDSC: projeto do controlador e otimização.

6.1.3 Projeto e Síntese do Controlador

Para o projeto do controle através da utilização de recursos de otimização, utilizou-se a estrutura *single loop* ilustrada na figura 66, onde estão presentes o elemento não linear de saturação do sinal de controle e o bloco NCD (*Nonlinear Control Design* - Interface para Otimização). O modelo da planta é obviamente, o modelo obtido no processo de identificação. A preocupação central neste trabalho é o fechamento de todo o CDSC proposto.

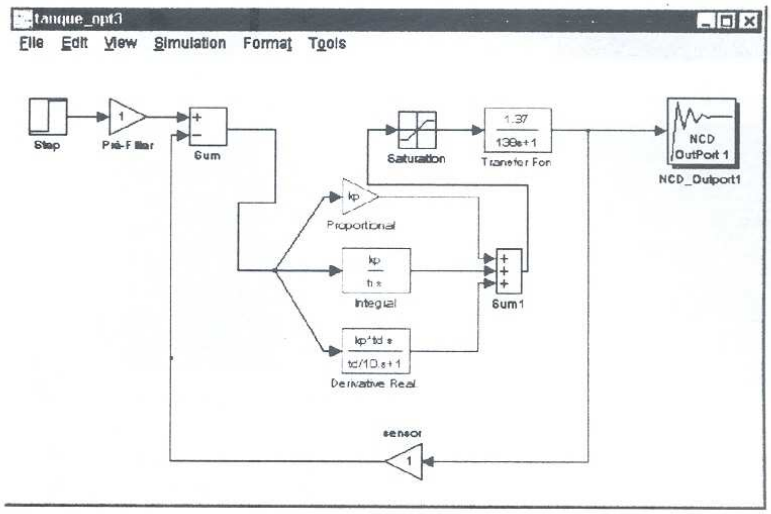


Figura 66 : Controlador PID sob otimização

A definição deste elemento não linear foi feito em 0 a 10V. Na implementação da lei de controle (controlador) pela ECUVI, está previsto a limitação da saída do sinal de controle à planta em 10 V.

O modelo de controlador utilizado é o PID. O algoritmo utilizado é :

$$Gc(s) = K_p\left(1+\frac{1}{T_i s}+\frac{T_d s}{1+Ns}\right)$$

Os parâmetros [Kp Ti Td] serão trabalhados de forma a estabelecer uma melhora na resposta do sistema realimentado, por exemplo, ao degrau. Para início do algoritmo de otimização relativo ao NCD é necessário o fornecimento de valores iniciais. Tais valores podem ser obtidos pelo método de Ziegler Nichols ou pela simples verificação da dinâmica da planta para valores do PID. A velocidade de convergência dos valores depende das exigências (restrições) feitas ao sinal e dos valores destes parâmetros. De forma a tornar mais evidente os resultados obtidos por esta ferramenta foram escolhidos valores que determinam pouca melhora na dinâmica do sistema: Kp=2 , Ti = 10000, Td = 1. Sabe-se que a planta em malha fechada sem a presença de controlador apresenta um tempo de subida de aproximadamente 135 segundos e atinge 63% do valor final em 58 segundos.

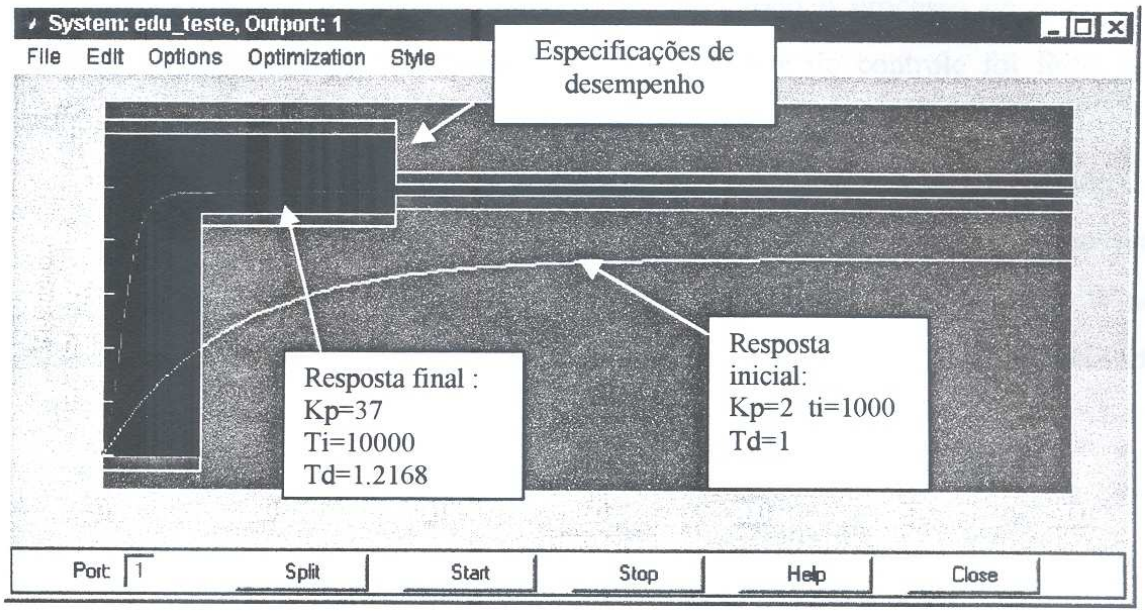


Figura 67 : Especificação de resposta para otimização do controle

A resposta inicial e as especificações desejadas são definidas através da interface gráfica da figura 67. Conforme verificado também nesta figura, a resposta resultante se ajustou da melhor forma sob as restrições impostas de resposta do sinal. Os parâmetros determinados (que determinam a resposta desejada) são então enviados à ECUVI para o processo de validação do controlador com HIL.

Para os valores em questão e resposta definida resultou nos parâmetros PID [Kp=37, Ti=10000, Td= 1.22]. A ação integral permanece sem efeito e a ação proporcional sofreu um grande acréscimo. Por este resultado percebe-se que o problema de controle está sendo tratado como um problema de otimização, onde os conceitos de análise da dinâmica do sistema com as ações de controle são vistos como uma "caixa preta".

6.1.4 Simulação

O algoritmo PID digital correspondente ao utilizado na simulação com processo *offline* é implementado na ECUVI através de sua equação a diferenças, conforme processo descrito no capitulo 3.

Através da GUI do módulo CIVM é possível todo o processo de configuração, monitoração e controle da simulação com HIL. O teste do controle foi feito com as seguintes especificações :

- Período de amostragem (Ts) de 1.5 s : como ainda não está implementado o RTX no gerenciamento do Windows, não é possível a garantia de tempo real no sistema. Portanto, o Ts sofre desvios aleatórios ditado por eventos do Windows NT. Como o período de amostragem é alto, percentualmente não traz efeitos nocivos ao sistema.
- Definição do sinal de referência : sinal quadrado de 160 s de período.

A resposta do sistema visto na figura 68 mostra um desempenho satisfatório do sistema para o controlador obtido. Ocorre neste momento a validação do controlador.

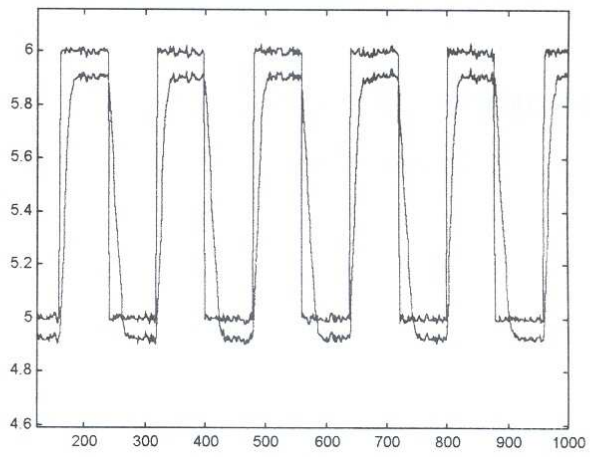


Figura 68 : Resposta sistema com controlador otimizado

6.2 Comentários

A parte que merece atenção na implementação do CDSC é a determinação de condições de tempo real para que se garantam taxas de amostragem constantes e também propicie diminuição em seus valores. Esta última permitiria o controle de plantas com constantes de tempo menores. A solução encontrada e que forma a base da concepção do VIEnCoD é o controle do sistema operacional do Windows através do RTX, que

conforme já mencionado no capítulo 4, encontra-se em processo de aquisição e que será alvo dos próximos trabalhos de implementação.

A funcionalidade dos módulos de Instrumentação Virtual (ECUVI) foram neste estudo de caso, aplicadas a toda a parte de configuração e operação da plataforma de instrumentação *VXIbus*. A utilização desta plataforma foi motivada por suas características ótimas de interface e instrumentação eletrônica, abordadas no Anexo I. Foram muitos os problemas de implementação encontrados de forma a propiciar o funcionamento adequado de cada placa dentro do contexto do ambiente. O maior problema na adaptação às necessidades do VIEnCoD foi com relação ao módulo de simulação com HIL, o CIVM, onde um tempo excessivo era gasto na operação da placa digitalizadora da plataforma VXI. Este ponto impossibilita aplicações a plantas com pequenas constantes de tempo. A solução está sendo atualmente trabalhada, tanto na solução voltada ao barramento VXI quanto a solução com placa multifunção para PC.

Para o estudo de caso utilizado - planta com constante de tempo elevada, os resultados formam satisfatórios apesar das restrições existentes.

Neste capítulo observou-se o fato de que o VIEnCoD permite que ferramentas avançadas de controle possam ser utilizadas de maneira amigável e prática atingindo simultaneamente os objetivos de ensino (contextualização de conceitos), pesquisa (implementação rápida de uma estratégia de controle) e serviços (ambiente amigável sem a necessidade de conhecimento a nível de pesquisador).

Capítulo 7

7 Conclusão

Para uma abordagem aplicada de sistemas de controle no contexto de ensino e pesquisa, surge a necessidade de uma ferramenta CACSD específica de suporte, suprindo a deficiência dos pacotes comerciais existentes, que atendesse a diversos objetivos, tais como :

- Ambiente amigável para suporte ao ensino e pesquisa de controle;
- Possibilidade de expansão e adaptação às necessidades do pesquisador como simulação e testes de novas estratégias de controle;
- Integração com elementos físicos dando suporte ao CDSC;
- Ferramenta de suporte a prestação de serviços na área de controle com característica de adaptação a dispositivos industriais de forma amigável;

As características acumuladas pela plataforma VIEnCoD em seu projeto e atual estado de implementação aliado à tendência sobre a temática de integração dos diversos elementos em diferentes níveis hierárquicos (item 2.3, 2.4 do capítulo 2) motivam a continuidade do presente trabalho nesta linha - integração a um sistema de gestão empresarial integrada ou ERP atuando como uma ferramenta de otimização de máquinas e processos ao nível de chão de fábrica estendendo os objetivos do sistema no controle e otimização do sistema produtivo.

A característica multifuncional direcionou a concepção do VIEnCoD. Sua implementação foi embasada sob o Ciclo de Desenvolvimento de Sistemas de Controle - CDSC, metodologia que orienta o tratamento do projeto de um sistema de controle desde a modelagem e identificação da planta à síntese do controlador.

Nesta metodologia são abordados muitos conceitos da teoria clássica de controle e recursos de controle avançado. Estes recursos são representados por algoritmos e funções que são encontrados nos pacotes computacionais sem a necessidade de desenvolvimento. O que ocorre, entretanto, é que estes recursos são disponibilizados de forma setorizada em

função da área de atuação em controle, compondo os chamados *toolboxes*. Esta característica vem de encontro a necessidade de integração funcional de recursos para suporte de todo o CDSC.

A linguagem de programação de alto nível aliado as fortes recursos de programação visual do Matlab sugeriram o desenvolvimento de uma estrutura de *software* baseada em seu ambiente operacional, de forma a integrar e gerenciar a grande quantidade de aplicativos voltados a controle disponíveis. A composição e aspectos funcionais desta estrutura baseou-se no CDSC proposto. São diversos *m-files* destinados ao atendimento de cada etapa determinando características modulares e definindo o VIEnCoD como um *toolbox* CACSD. A característica modular na composição do *toolbox* e na sua característica operacional determinam a possibilidade de fácil integração de novas contribuições. Especial destaque se deu à implementação de técnicas especiais no projeto das GUIs tornando-as intuitivas no seu uso com repasse de conhecimento. Este contexto pode ser avaliado no capítulo 6.

A insuficiente disponibilidade de referências bibliográficas sobre programação (principalmente aos recursos visuais) no Matlab tornaram o desenvolvimento do ambiente essencialmente baseado no estudo de aplicativos já existentes. Entretanto, a criação de aplicativos específicos baseados em Matlab é uma tendência atual, justificando o esforço empregado.

A etapa inicial e final do CDSC prevê a interface com elementos físicos. O processo de Identificação necessita de dados relativos a excitação da planta e sua reposta. A última etapa prevê a validação do controlador através de simulação com HIL onde elementos físicos (planta) são inseridos na malha de controle juntamente com modelos (controlador). Ambos os casos necessitam de uma estrutura de *hardware* (sistema de aquisição e processamento) que promova a interface com o ambiente Matlab.

A concepção de Instrumentação Virtual permite que o processo de configuração, supervisão e controle do sistema de aquisição escolhido (ampla gama de possibilidades) seja feito através de aplicativos desenvolvidos (sob a forma de GUIs) executando em Windows. Esta automação da instrumentação, encapsulada pelos aplicativos sob forma

modular, são incorporados pela estrutura do *toolbox* VIEnCoD. Foram definidos, portanto, os módulos PCVM, PIVM e CIVM que compõe a ECUVI para suporte as fases do CDSC.

A abertura de *hardware* do ambiente permite a definição do tipo utilizado no sistema, por exemplo, segundo sua funcionalidade: plataforma *VXIbus* voltado a pesquisa e serviços (especificação rígida de instrumentação industrial; sistema modular) e plataforma baseada em placas para PC multi-funções para suprimento de laboratórios em aulas práticas (sistema de baixo custo).

Apesar da grande facilidade de operação e criação de aplicativos baseados em LabWindows/CVI, existiu um trabalho intenso no processo de configuração e operação do sistema *VXIbus* - plataforma de hardware (instrumentação) adotada no estudo de caso - de forma a adequá-lo às condições impostas pelas características de instrumentação necessárias à identificação e simulação com HIL. Existem ainda propriedades que devem ser trabalhadas como o aspecto de imposição de condições de tempo real adequados, em função de características de configuração do sistema VXI.

A simulação com HIL visa a validação do controlador obtido na fase de projeto e otimização em processo *offline*. As condições impostas ao projeto e simulação basearam-se em critérios de desempenho bem definidos de forma a trabalhar a dinâmica da planta colocando-a em operação ótima. Estes critérios estão diretamente relacionados com o tempo de ciclo de malha e período de amostragem. A estabilidade do sistema depende do atendimento de suas constantes de tempo. Para tal é necessário que o ambiente CACSD forneça condições de tempo real para as simulações com HIL garantindo a reprodução dos resultados obtidos em processo de simulação *offline* para propósito de validação.

Na implementação adotada pelo VIEnCoD existem elementos que atuam no estabelecimento das condições de tempo real. O controlador executa no mesmo processador onde executa todo o processo *offline* do ambiente sob o sistema operacional do Windows, não havendo portanto, qualquer condição de tempo real. De forma a evitar a utilização de plataforma de *hardware* adicionais e dedicadas ao controlador, é proposto a utilização da ferramenta RTX para atuação do sistema operacional do Windows dando

condições ao controlador operar em tempo real. Este procedimento sustenta a base da concepção do VIEnCoD permitindo a abertura em termos de *hardware*, definido pelo sistema de aquisição.

O estudo aprofundado direcionado pelas necessidades operacionais expostas e a especificação da ferramenta RTX representam o esforço do presente trabalho. A implementação e testes não foram ainda realizados em virtude da indisponibilidade do recurso, em processo de aquisição. Este é o principal ponto de destaque para continuidade das atividades de implementação do projeto. Esta característica pode ser suprida pelo trabalho de gerenciamento de prioridades do Windows aliado a elevada constante de tempo da planta (estudo de caso) que forneceu resultados satisfatórios.

O procedimento de simulação com HIL permite a validação do controlador. Uma vez validado é possível a sua transferência para o controlador industrial presente no sistema. O LabWindows/CVI apresenta diversos drivers destes equipamentos e sistemas (CLPs, sistemas Fieldbus,... etc) permitindo a interface automática. Este é um importante ponto a ser considerado quando da adequação futura do VIEnCoD para operação recursiva automática com o sistema. Estratégias de controle adaptativas poderiam ser integradas ao controlador industrial diretamente através do VIEnCoD que operaria com controlador *self-tuning*. Outro ponto também é a atuação mais efetiva dentro de um sistemas ERP, onde temos por exemplo, sistemas Fieldbus dotados de interfaces com sistema SAP.

O capítulo 6 dedicou-se a utilização efetiva de todo o ambiente colocando em observação o atingimento dos objetivos propostos, verificação de deficiências e afirmação de necessidades de continuidade de trabalhos em pontos específicos.

7.1 Perspectivas futuras

Em função da característica multifuncional que o VIEnCoD apresenta, existe um leque muito grande de contribuições em diversos pontos de atuação. Conforme já colocado acima existem ainda implementações necessárias para melhoria e adequação integral da concepção proposta. Divide-se então em dois grupos :

1. **Ajustes e melhorias :**
   - Implementação do RTX e verificação dos resultados inerentes às condições de tempo real imposta ao sistema;
   - Implementações de software que não foram atingidas em virtude do tempo tais como: melhor controle e atuação sobre as ferramentas de identificação e otimização incorporadas por suas interfaces gráficas de forma a permitir inserção e leitura automática de dados, outros problemas de ordem menor;
   - Implementação de sistema de hardware baseado em placa DAQ e verificação das características para comparação dos resultados com a plataforma *VXIbus*.

2. **Futuras contribuições :**
   - Implementação de um ambiente de desenvolvimento de controladores FUZZY;
   - Desenvolvimento de interfaces, via Instrumentação Virtual, à barramentos de comunição de campo (Fieldbus, Profibus) e interfaces a sistemas supervisórios elevando a hierarquicamente a atuação dentro do contexto ERP.
   - Adaptação do ambiente para implementação de estratégias de controle adaptativas que requerem operação recursiva em processo *online*. Um caminho poderia ser um trabalho bem detalhado sobre o Matlab Compiler e RTW;
   - Implementação da ferramenta *Run Time Server* para geração de aplicativo *stand alone*.

Anexo I - Estrutura de *hardware* utilizada - *VXIbus*

A atual implentação de *hardware* do VIEnCoD se baseia na plataforma de instrumentação modular *VXIbus*. Previamente a descrição desta estrutura, vale ressaltar da abertura de *hardware* suportado pelo ambiente permitindo a utilização de outros sistemas de aquisição como placas DAQ, que está em processo de implementação.

O *VXIbus* (*VMEbus eXtension for Instrumentation*) é um sistema de instrumentação modular controlada por computador que possui as seguintes características :

- Ampla gama de produtos (placas), são mais de 1000 placas ampliando as possibilidades na configuração da plataforma;
- Abertura em termos de fornecedores pela padronização nas especificações VXI a nível de *hardware*;
- *Software* com padronização *VXIplug&play* simplificando a configuração, operação e integração do sistema;
- Características otimizadas tais como : menor dimensão física, melhor fidelidade do sinal, redução dos níveis de ruído e outras;
- Estrutura modular e robusta determinando taxas MTBF (*Mean-Time Between Failure*) e MTTR (*Mean-Time To Repair*) satisfatórias.

Estas características agregam vantagens ao sistema tais como : redução nos tempos de integração do sistema, na reconfiguração do sistema (adaptação ou mudanças) e no *down time* do sistema (reposição de instrumentos).

Estes aspectos levaram a utilização da plataforma na implementação atual do VIEnCoD. Esta, baseada nas necessidades impostas pelo ambiente, foi estruturada sob a seguinte configuração :

- Pentium embutido com 80M RAM – Controlador do sistema : menor capacidade de memória implica em dificuldades de processamento pelo carregamento gráfico imposto pelos aplicativos do *toolbox* VIEnCoD;
- Sistema operacional Windows 95 ou NT (*Server* ou *Workstation*) : o RTX deve estar presente para estabelecimento das condições de tempo real;

- Um Gerador de Formas de Onda Arbitrárias Tektronix VX4790A : disponibilizar uma variedade de sinais de excitação; elemento responsável pela injeção de sinal no sistema (de excitação e controle);
- Um Condicionador de Sinais Tektronix VX4780 : tratamento dos sinais de leitura dos sensores (filtragem, atenuações,... etc) evitando a utilização de algoritmos matemáticos que realizam esta tarefa e introduzindo aproximações indesejadas na realização;
- Um Digitalizador de 16 canais Tektronix VX4244 : interface de conversão A/D. Elemento de leitura dos sinais de resposta do sistema. Deve apresentar resolução e velocidade de conversão elevados com a preocupação das características de tempo real;

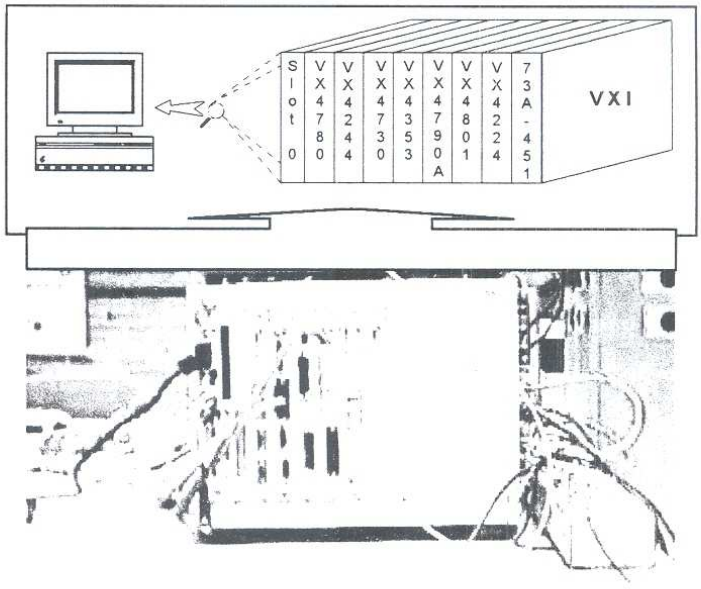


Figura 69 : Plataforma de *hardware* implementada - *VXIbus*

- Uma Placa D/A de 12 canais Tektronix VX4730 : interface para injeção de sinais de controle ao sistema;
- Uma Placa de Comutação de Relés de 32 canais Tektronix VX4353: para comutar as fontes de sinais de excitação e controle do sistema- VX4730 e VX4790.

A figura 69 ilustra a composição da plataforma utilizada. Maiores detalhes quanto a arquitetura e características operacionais podem ser encontradas no Apêndice II deste trabalho.

Glossário

CACSD – *Computer Aided Control System Design.* Ambiente de *software* para estudo, análise e projeto de sistemas de controle possuindo ou não plataforma de aquisição integrado.

Callbacks – rotinas vinculadas a um elemento gráfico de uma GUI que as chama.

CDSC – Abreviatura do Ciclo de Desenvolvimento de Sistemas de Controle.

Ciclo de Desenvolvimento de Sistemas de Controle – metodologia que orienta a abordagem de áreas em sistemas de controle desde a modelagem e identificação da planta à síntese e implementação do controlador.

CIVM – *Controller Implementation VIEnCoD Module.* Módulo da ECUVI de suporte à realização da implementação de controladores com simulação HIL.

CLP – Controlador Lógico Programável. Dispositivo dotado de microcontrolador e interface analógica e digital destinado ao controle programado de máquinas e processos industriais.

Compensador – Neste trabalho apresenta o mesmo significado que controlador - elemento destinado a determinação de comportamento dinâmico específico a um sistema realimentado.

CVI – *C for Virtual Instrumentation.* Ambiente de *software* para desenvolvimento de aplicativos baseados em instrumentação virtual.

DAQ – *Data Acquisition Board.* Elemento de *hardware* aqui referenciado como placa para instrumentação baseado em PC.

DDE – *Dynamic Data Exchange.* Processo de troca de dados que ocorre entre aplicativos Windows.

ECUVI – *Electronic Control Unit Virtual Instrumentation-based.* Denominação de toda estrutura de suporte para simulação com HIL que é composta por elementos de *hardware* (plataforma de aquisição e controlador) e *software* (supervisão e controle da instrumentação e implementação do controlador sob o conceito de instrumentação virtual).

GUI – *Graphical User Interface.* Painéis padrão Windows desenvolvidos para funções específicas.

HIL – *Hardware-in-the-Loop*. Processo de simulação onde, partindo-se de um sistema todo representado através de modelos, gradativamente ocorre a substituição destes por elementos físicos reais.

Hint – Pequena caixa de texto com informações sobre um elemento da interface gráfica.

LTI – *Linear Time Invariant*.

MA – Malha aberta. Sistema de controle sem realimentação.

MF – Malha fechada. Sistema de controle onde existe realimentação, ou seja, o sinal de saída do sistema retorna a entrada do sistema visando a melhora de desempenho do sistema.

m-file – Denominação aos arquivos Matlab que representam funções.

MIMO – *Multi-Input Multi-Output*. Referência a sistemas de controle onde existe a participação de mais de uma entrada e uma saída.

PCVM – *Plant Configuration VIEnCoD Module*. Módulo da ECUVI responsável pela configuração do sistema de controle (planta e plataforma de instrumentação).

PID – Controle Proporcional Integral e Derivativo.

PIVM – *Plant Identification VIEnCoD Module*. Módulo da ECUVI responsável pelo gerenciamento do processo de aquisição de dados para obtenção das *realizações*.

Planta – No presente trabalho pode ser entendido como qualquer objeto físico a ser controlado.

PRBS – *Pseudorandom Binary Sequence*. Sinal implementado através de algoritmo que simula as características de um ruído branco.

Processo – No presente trabalho pode ser entendido como qualquer operação a ser controlada.

RCP – *Rapid Control Prototyping*. Metodologia que permite a aplicação e teste de novas estratégias de controle em *hardware* sob tempo real rapidamente e facilmente, para validação do controle antes de sua aplicação em dispositivo industrial.

Realização – Termo relacionado à implementação da lei de controle para simulação com HIL com propósito de validação.

RTW – *Real-Time Workshop*. Aplicativo Matlab que gera código C de diagramas Simulink.

RTX – *Real Time Extension for Windows*. Aplicativo para gerenciamento do Windows para operação em tempo real.

S/H – *Sample / Holder.* Amostrador e Segurador; elementos participante do processo de discretização de um sinal analógico.

SAP – *Software* para gestão empresarial integrada.

Scheduling – Programação Finita da Produção.

Scripts – tipos de arquivo Matlab (m-file) sem necessidade de entrada de parâmetros.

Single loop – sistema formado apenas por um laço de realimentação.

SISO – *Single-Input Single-Output.* Referência a sistemas de controle onde existe apenas a participação de uma entrada e uma saída.

Sistema – Combinação de componentes que atuam conjuntamente e realizam um objetivo.

Sistema físico – Referência a elemento(s) reais e não a modelos. Também denominado de sistema real.

Sistemas em Tempo real – Denominação que faz a sistemas onde existe a preocupação quanto ao tempo (desempenho) de processamento (p. ex. sistemas com constantes de tempo pequenos exige processamento rápido) e garantia nos tempos aplicados (p. ex. período de amostragem) aos eventos, implicando determinismo ao sistema.

Tempo de ciclo de malha de controle – tempo decorrido entre a leitura do sinal vindo do sensor e a posterior injeção do sinal de controle.

VI – *Virtual Instrumentation* ou *Virtual Instrument.* Instrumentação Virtual ou Instrumento Virtual é a denominação que se faz a dispositivos de instrumentação onde a configuração, controle, monitoração e visualização são feitos via *software* através do PC, que se comunica com o elemento de *hardware*.

Referências Bibliográficas

[ALSÈNE, 1999] ALSÈNE, A. The Computer Integration of the Enterprise. *IEEE Transactions on Engineering Management*, v. 46, n. 1, p. 26-35, February 1999.

[AMARAL et al., 1992] AMARAL, W. C.; ARRUDA, L. V.; BUENO, S. S. et al. CAD Package for Industrial Process Identification. *J. Proc. Cont.*, v. 2, n. 3, p. 155-161, 1992.

[ARGONDIZZA et al., 1997] ARGONDIZZA, A.; BONA, B.; CARABELLI, S. Automatic Control Teaching with Matlab and MatDSP. *Proceedings, 3rd European Control Conference*, Belgium, p. 01-06, 1997.

[ARMSTRONG, 1997] ARMSTRONG, B. A Controls Laboratory Program with Accent on System Identification. *Proceedings of the 1997 American Control Conference*, Oct. 1997.

[ARRUDA et al., 1994] ARRUDA, L. V. A Knowledge-Based Environment for Intelligent Design and Supervision of Control Systems. *IEEE Conference on Systems Man and Cybernetics*, Texas, USA, v. 3, p. 2680-2685, 1994.

[ARRUDA, 1992] ARRUDA, L. V. *Etude d'Algorithmes d'Estimation Robuste et Developpement d'un Systeme a Base de Connaissance pour l'Identification.* Sophia Antipolis, França, 1992. Tese de Doutorado - Universidade de Nice.

[ÅSTRÖM et al., 1998] ÅSTRÖM, K. J.; JOHANSSON, M.; GÄFVERT, M. Interactive Tools for Education in Automatic Control. *IEEE Control Systems*, v. 18, n. 3, p. 33-40, Jun. 1998.

[BARCZAK, 1995] BARCZAK, C. L. *Controle Digital de Sistemas Dinâmicos.* São Paulo : Edgard Blücher, 1995.

[BOOM et al., 1994] BOOM, A. van den; Grubel, G.; Varga, A. Developments in the Field of Control Libraries and their Impact on Control Education. *Proceedings of the 3rd Symposium IFAC Advances in Control Education*, Japan , p. 287-290, Aug. 1994.

[BOX et al., 1994] BOX, G. E.; Jenkins, G. M.; Reinsel, G. C. *Time Series Analysis.* 3rd ed. New Jersey : Prentice-Hall, 1994.

[BRANCH et al., 1996] Branch, M. A.; Grace A. *Matlab Optimization Toolbox User's Guide*. USA: The MathWorks, Dec. 1996.

[BUSETTI, 1998] BUSETTI DE PAULA, M. A. *Prozeβkopplung mittels einer VMEbus-basierten Hardware-Plattaform für eine verteilte Simulationsumgebung mechatronischer Systeme.* Paderborn, Alemanha, 1998. Tese de Doutorado - Universidade de Paderborn.

[CHAFOUK, 1997] CHAFOUK, H. Automatica Industrial. *Seminário, Curso de Pós Graduação em Engenharia Elétrica e Informática Industrial, Centro Federal de Educação Tecnológica do Paraná*, Curitiba, 15 à 19 de Setembro de 1997.

[COELHO, 1995] COELHO, A. R. Laboratory Experiments for Education in Process Control. *Proceedings of the Workshop on Control Education and Technology Transfer Issues*, Curitiba, PR, p. 133-138, September 1995.

[COGHI, 1996] COGHI, M. A. Otimização de Processos pela Perfeita Análise e Sintonia de Malhas de Controle. *Anais do 1º Simpósio de Automátia Aplicada*, São-Paulo, SP, p. 13-22, Set. 1996.

[DOYLE et al., 1996] DOYLE III,F. J.; VENKATASUBRAMANIAN, V.; KENDI, T. A. Purdue Control Modules: A Flexible Set of *Software* Modules for na Undergraduate Process Dynamics and Control Laboratory. *Computer Aplications in Engineering Education*, v. 4(3), p. 179-190, 1996.

[DOYLE et al., 1998] DOYLE III,F. J.; GATZKE, E. P.; PARKER, R. S. Practical Case Studies for Undergraduate Process Dynamics and Control Using Process Control Modules. *Computer Aplications in Engineering Education*, v. 6, p. 181-191, 1998.

[FLAUS et al., 1994] FLAUS, J.; CHERUY, A.; LEBOURGEOIS, F. ODASYS: A *Software* Tool to Help Knowledge and Experience Interaction Between University and Process Industry. *Proceedings of the 3rd Symposium IFAC Advances in Control Education*, Japan, p. 239-241, August 1994.

[HANSELMANN, 1996] HANSELMANN, H. *Hardware*-in-the-Loop Simulation Testing and its Integration into a CACSD Toolset. *Proceedings of the IEEE International Symposium on Computer-Aided Control System Design*, Michigan, USA, September 1996.

[HORÁCEK, 1994] HORÁCEK, P. Modular Control Laboratory Architecture. *Proceedings of the 3rd Symposium IFAC Advances in Control Education*, Japan, p. 271-274, August 1994.

[LANIER et al., 1997] LANIER, B.; ROSS, L.; KAKAD, Y. P. Fuzzy logic for Process Control. *Proceedings of the 13th ISPE/IEE International Conference on CAD/CAM Robotics & Factories of the Future*, Colombia, v. 1, p. 35-39, December 1997.

[LJUNG, 1995] LJUNG, L. *Matlab System Identification Toolbox User's Guide*. USA: The MathWorks, May 1995.

[LOURES et al., 1995] LOURES, E. R.; Silveira, M. R.; Busetti, M. A. LEPEC - Laboratório de Ensino e Pesquisa em Energia e Controle : Uma Proposta de Integração. *XXIII Congresso Brasileiro de Ensino de Engenharia*, Recife, v. 1, p. 61-68, Out. 1995.

[MACIEL, 1997] MACIEL, P. V. Z. *Modelagem de um Dispositivo Virtual de Manufatura para Controle de Equipamentos por ANALYTICE.* Dissertação (Mestrado), Pós Graduação em Engenharia Elétrica e Informática Industrial, Centro Federal de Educação Tecnológica do Paraná, 1997.

[MAGNUS, 1993] MAGNUS, R. C. Computer-Aided Control System Design. *IEEE Control Systems*, v. 13, p. 14-16, 1993.

[MARCHAND, 1999] MARCHAND, P. *Graphics and GUIs with MATLAB.* 2ª ed. CRC Press, Apr. 1999.

[MATKO et al., 1994] MATKO, D.; SCHUMANN, R.; ZUPANCIC, B. Design of a Realistic CACSD Course. *Proceedings of the 3rd Symposium IFAC Advances in Control Education*, Japan, p. 263-266, August 1994.

[MCDONALD, 1998] MCDONALD, R. Real-Time Control and Reliability with Windows NT. *AutomationView Newsletter*, v. 3, n. 4, p. 1-3, Winter 1998/99.

[MOLLENKAMP, 1988] MOLLENKAMP, R. A. *Controle Automático de Processos.* 1ª ed. São Paulo : EBRAS, 1988.

[MUNTEANU et al., 1997] MUNTEANU, D.; MICHAU, F.; GENTIL, S. Autodidact: a Simulation-based Learning Environment in Automatic Control. *Proceedings of the 3rd European Control Conference*, Belgium, 1997.

[OGATA, 1993] OGATA, K. *Engenharia de Controle Moderno.* 2ª ed. Rio de Janeiro : Prentice-Hall do Brasil, 1993.

[OGATA, 1995] OGATA, K. *Discrete-Time Control Systems.* 2ª ed. New Jersey : Prentice-Hall, 1995.

[RAVN et al., 1995] RAVN, O.; TORP, S.; NORGAARD, P. M. et al. A CACE Tool for Analysis and Design of Adaptive Control Systems. *Proceedings of the Symposium on Adaptive Systems in Control and Signal Processing*, Budapest, Hungary, 1995.

[RINGWOOD et al., 1994] RINGWOOD, C.; McCORKELL, C.; WHELAN, J. A Final Year Undergraduate Digital Control Laboratory Assignment. *Proceedings of the 3rd*

Symposium IFAC Advances in Control Education, Japan, p. 207-210, August 1994.

[RUTZ et al., 1995] RUTZ, R.; RICHERT, J. CAMeL - An Open CACSD Environoment. *IEEE Control Systems*, April 1995.

[SEIDEL, 1994] SEIDEL, M. A Hypertext Based Tutoring System for a CACSD Tool. *Proceedings of the 3rd Symposium IFAC Advances in Control Education*, Japan, p. 189-192, August 1994.

[SILVEIRA et al., 1998] SILVEIRA, M. R.; ARRUDA, L. V.; LOURES, E. R. Integração de Novas Tecnologias de Equipamentos e Métodos de Ensino Baseados em Computador. *Anais do IV Congresso Ibero-Americano de Informática na Educação*, Brasília, Nov. 1998.

[SILVEIRA, 1998] SILVEIRA, M. R. *Desenvolvimento de um Sistema Multimídia para Autoria de Cursos de Controle*. Dissertação (Mestrado), Pós Graduação em Engenharia Elétrica e Informática Industrial, Centro Federal de Educação Tecnológica do Paraná, 1998.

[SIPPER et al., 1997] SIPPER, D.; BULFIN, L. R. *Production Planning, Control and Integration*". USA : McGraw-Hill, 1997.

[SOARES et al., 1995] SOARES, L. R.; CASANOVA, V. H. Laboratory Workstation and Experiments for Education on Discretization, Identification and Computer Control of Dynamic Process. *Proceedings of the Workshop on Control Education and Technology Transfer Issues*, Curitiba, PR, p. 139-143, September 1995.

[SÖDERSTRÖM et al., 1989] SÖDERSTRÖM, T.; STÖICA, P. *System Identification.* New Jersey : Prentice-Hall, 1989.

[The Mathworks, 1997a] The MathWorks. *Building GUIs with Matlab*. USA: The MathWorks, 1997.

[The MathWorks, 1997c] The MathWorks. *Using Matlab*. USA: The MathWorks, 1997.

[The MathWorks, 1997d] The MathWorks. *Using Simulink*. USA: The MathWorks, 1997.

[VIZJAK et al., 1994] VIZJAK, A.; ZUPANCIC, B.; JOVAN, V. et al. A Process Laboratory for Analysis, Education and Research of Control Engineering Approaches and Methods - a Conept. *Proceedings of 3rd Symposium IFAC Advances in Control Education*, Japan , p. 267-270, August 1994.

[VOLLMANN et al., 1997] VOLLMANN, T. C.; Berry, W. L.; Whybark, D. C. *Manufacturing Planning & Control Systems*. 4th ed. McGraw Hill, 1997.

[WITTENMARK et al., 1998] WITTENMARK, B.; HAGLUNG, H.; JOHANSSON, M. Dynamic Pictures and Interactive Learning. *IEEE Control Systems* , v. 18, n. 3, p. 26-32, June 1998.

[WOLF, 1998] WOLF, M. Visual-MOOMo - A Graphical Object-Oriented Modelling Environment for the Design of Mechatronic Systems. *Proceedings of the 6th UK Mechatronics Forum International Conference*, Mount Billingens, Sweden, Sep. 1998.

[YEDDANAPUDI et al., 1997] YEDDANAPUDI, M.; Potvin, A. *Matlab Nonlinear Control Design Blockset User's Guide*. USA: The MathWorks, 1997.

[ZANELLA, 1996] ZANELLA, M. C. *Concepção de um Ambiente de Simulação de Sistemas Mecatrônicos com Hardware-in-the-loop*. Dissertação (Mestrado), Pós Graduação em Engenharia Elétrica e Informática Industrial, Centro Federal de Educação Tecnológica do Paraná, 1996.

Bibliografia Recomendada

[ATHERTON et al., 1994] ATHERTON, D. P.; SORENSEN, O. B.; GOUCEN, A. Teaching Control Engineering Using Implementation of Matlab. *Proceedings of the 3rd Symposium IFAC Advances in Control Education*, Japan, p. 291-294, Aug. 1994.

[BARKER et al., 1993] BARKER, H. A.; CHEN, M.; GRANT, P. W. et al. Open Architecture for Computer-Aided Control Engineering. *IEEE Control Systems*, v. 13, p. 17-26, Apr. 1993.

[BUSETTI et al., 1999a] BUSETTI, M. A.; LOURES, E. R. VIEnCoD - Proposal of an Environment CACSD integrated to an ERP system as a tool in the Support to the Control System Development Cycle Based on *VXIbus* / LabWindows and MATLAB Platform. *Proceedings of the Dynamic Problems in Mechanics and Mechatronics*, Günzburg, Germany, July 1999.

[BUSETTI et al., 1999b] BUSETTI, M. A.; LOURES, E. R. VIEnCoD - Proposta de Integração de Plataforma de Instrumentação Virtual e Matlab na Implementação de um Ambiente CACSD. *Anais do 4º Simpósio Brasileiro de Automação Inteligente*, São Paulo, SP, p. 603-608, Setembro 1999.

[DSPACE, 1997] DSPACE. *Solution for Control*. Catálogo de soluções da plataforma, Alemanha, 1997.

[GILLET et al., 1997] GILLET, D.; SALZMANN, C.; BONVIN, D. et al. Telepresence : Na Opportunity to Develop Real-World Experimentation in Education. *Proceedings of the 3rd European Control Conference*, Belgium, 1997.

[JOBLING et al., 1994] JOBLING, C. P.; GRANT, P. W.; BARKER, H. A. et al. Object-oriented Programing in Control System Design: a Survey. *Automatica*, v. 30, n. 8, p. 1221-1261, 1994.

[JOSHUA et al., 1997] JOSHUA, P.; SUN, Y.; TILBURY, D. et al. Control Tutorials for Matlab On the World Wide Web, *Proceedings of the American Control Conference*, USA, Oct. 1997.

[LAWSON, 1992] LAWSON, H. W. *Parallel Processing in Industrial Real-Time Applications*. New Jersey : Prentice-Hall, 1992.

[LJUNG et al., 1989] LJUNG, L.; NAGY, A. J. An Intelligent Toolfor System Identification. *Proceedings of the IEEE Workshop on Computer-Aided Control System Design*, Florida, USA, p. 58-63, December 1989.

[LOURES et al., 1998a] LOURES, E. R.; Busetti, M. A.; Arruda, L. V.; Erzinger, A.; Jansson, J. VIEnCoD - Proposta de um ambiente CACSD no Suporte ao Ciclo de Desenvolvimento de Sistemas de Controle baseado em plataforma *VXIbus* / LabWindows e MATLAB. *Anais do 12º Congresso Brasileiro de Automática*, Uberlândia, MG, v. 6, p. 1903-1910xx, Set. 1998.

[LOURES et al., 1998b] LOURES, E. R.; BUSETTI, M. A. VIEnCoD - Proposal of an environment CACSD in the Support to the Control System Development Cycle based on *VXIbus* / LabWindows and MATLAB Platform. *Proceedings of the Midwest Symposium on Circuits and Systems*, Indiana, USA, p. 296-299, August 1998.

[LOURES et al., 1999] LOURES, E. R.; BUSETTI, M. A. VIEnCoD - Proposal of an Environment CACSD integrated to an ERP system as a tool in the Support to the Control System Development Cycle Based on *VXIbus* / LabWindows and MATLAB Platform. *7th IEEE International Conference on Emerging Technologies and Factory Automation*, Barcelona, Spain, p. 761-766, October 1999.

[MARTIN et al., 1996] MARTIN, H.; MEIER-NOE, U. Classification in the Object-Oriented Modelling Language Objective-DSS Exemplified by Vehicle Suspensions. *Proceedings of the IEEE International Symposium on Computer-Aided Control System Design*, Michigan, USA, September 1996.

[NATIONAL, 1996a] NATIONAL INSTRUMENTS. *LabWindows/CVI Advanced.* Hands-on Course, Version 1.0, Part Number 321057A-01, 1996.

[NATIONAL, 1996b] NATIONAL INSTRUMENTS. *LabWindows/CVI Basics*, Hands-on Course, Version 2.0, Part Number 320803D-01, 1996.

[NATIONAL, 1996c] NATIONAL INSTRUMENTS. *LabWindows/CVI – Visual Programming for Instrumentation*, Manuals, Part Numbers 32068xC-01, 1996.

[NATIONAL, 1997] NATIONAL INSTRUMENTS. *LabWindows/CVI VXI*, Hands-on Course, Version 3.0, Part Number 320968C-01, 1997.

[NATIONAL, 1998] NATIONAL INSTRUMENTS. *Instrumentation - Measurement and Automation.* Catalogue, 1998.

[TECQUIPMENT] TECQUIPMENT. *CE105/CE105MV Coupled Tanks Apparatus.* Canada: TecQuipment, 1993.

[The Mathworks, 1996a] The MathWorks. *Control System Toolbox User's Guide*. USA: The MathWorks, 1996.

[The Mathworks, 1996b] The MathWorks. *Using Matlab Graphics*. USA: The MathWorks, 1996.

[The MathWorks, 1997b] The MathWorks. *Real-Time Workshop User's Guide*. USA: The MathWorks, 1997.

[The Mathworks, 1997e] The MathWorks. *Matlab Compiler User's Guide*. USA: The MathWorks, 1997.

Apêndice I - Ambiente Operacional – LabWindows/CVI

A principal característica do LabWindows/CVI são os recursos que ele oferece para suporte ao desenvolvimento de aplicativos voltados a instrumentação. Os painéis das funções estão dispostos organizadamente em uma árvore hierárquica, subdividida por assunto. Através do menu *Library* da janela *Project* pode-se acessar os seguintes grupos de bibliotecas:

• *User Interface* : Permitem a elaboração de interfaces gráficas com a ajuda de uma coleção de objetos como barras de menu, painéis, controles e gráficos. A interface do usuário pode ser construída pela edição de códigos na composição do programa ou com o *User Interface Editor*.

• *Analisys* : Inclui funções para manipulação de vetores uni-dimensionais e bi-dimensionais, operações complexas, operações com matrizes, e funções estatísticas.

• *Advanced Analisys* : Inclui funções para geração de sinais, processamento de sinais e ajuste de curvas.

• *Data Acquisition* : Serve para controlar as placas de aquisição de dados da *National Instruments*. Isto inclui as placas compatíveis com IBM PC AT quem usam o barramento AT, e as EISA-A2000 para computadores EISA.

• *Easy I/O for DAQ* : Contém funções que tornam mais fácil a escrita de programas DAQ simples comparativamente a utilização da biblioteca *Data Acquisition*.

• *VXI* : Servem para controlar os equipamentos VXI a partir de um controlador VXI embutido (*embedded*) ou de um PC equipado com controlador VXI e comunicação via MXI. A biblioteca VXI inclui funções de Comandante e Servidor para o Protocolo Serial de Palavra (*word serial protocol*), acesso de baixo nível ao *VXIbus*, acesso a recursos locais, sinais do VXI, interrupções, *triggers*, gerenciadores de interrupção do sistema, e configuração do sistema.

• *GPIB/GPIB 488.2* : Serve para criar uma interface com instrumentos GPIB usando o protocolo NI-488/488.2.

• RS-232 : Serve para controlar as portas RS-232 em um PC ou estação SPARC.

• *VISA* : Propicia aos desenvolvedores de *software* para VXI e GPIB, particularmente aos desenvolvedores de *drivers* de instrumentos, uma biblioteca de interface única para controlar instrumentos VXI, GPIB e RS-232.

• *TCP* : Usa a biblioteca *Transmission Control Protocol*, para controlar cartões de rede TCP no PC e nas estações SPARC.

• *DDE* : Usa a biblioteca *Dynamic Data Exchange* para criar uma interface com outras aplicações Windows usando o padrão DDE.

• *Formatting and I/O* : Serve para entrar e sair dados para arquivos e manipular o formato de dados em um programa.

• *Utility* : Contém funções que não constam em nenhuma outra biblioteca LabWindows/CVI. Servem para manipulação de arquivos e outras tarefas de miscelânea.

• *ANSI C* : Contém as funções ANSI C disponíveis no LabWindows/CVI.

O ambiente de programação do LabWindows conta com um recurso chamado *SmartCode*, que proporciona recursos de edição e *debug* similares aos encontrados nos modernos ambientes ditos "visuais". Alguns destes recursos são:

• Geração de Código - ao utilizar um painel de funções para gerar código, o usuário pode automaticamente recuperar uma lista de variáveis que o mesmo declarou em seu programa ou os controles definidos pelo usuário no GUI;

• Sintaxe Colorizada;

- Menus Sensíveis ao Contexto, a partir dos quais o usuário pode:
  - recuperar um painel de função para ver ou mudar valores de argumentos;
  - encontrar automaticamente objetos de interface do usuário referenciados no código fonte;
  - abrir arquivos adicionais referenciados no código fonte;
  - demais funções encontradas na maioria dos ambientes de desenvolvimento existentes no mercado.

Apêndice II - Arquitetura do Sistema *VXIbus*

O *VXIbus* (*VMEbus eXtension for Instrumentation*) surgiu da necessidade de reduzir o tamanho dos Equipamentos de Testes Automáticos entre outros fatores, levando a criação das normas para instrumentação modular baseada nas especificações do *VMEbus*. Por ser uma extensão do *VMEbus* também é uma arquitetura de sistema aberto, permitindo módulos de instrumentos de vários fabricantes; conexão em um mesmo *bakpanel* sob operação em sistema integrado. O *VXIbus* não define um sistema de hierarquia ou topologia específico, nem o tipo de microprocessador específico, sistema operante ou o tipo de interface para o computador. O que *VXIbus* define é a plataforma sobre qual as resoluções podem ser feitas assegurando compatibilidade entre diferentes fabricantes. A seguir e feita a abordagem de alguns termos e da arquitetura de *hardware* do sistema.

1 Módulos VXIbus

Os módulos *VXIbus* são uma parte física básica, no qual o sistema *VXIbus* é construído. Estes módulos vem em 4 tamanhos básicos, chamados de A, B, C e D. Os módulos A e B são totalmente compatíveis com o padrão *VMEbus*. Já os módulos C e D foram acrescentados à especificação *VMEbus* orientada em direção a instrumentação que pode melhor ser descrita como um *super-dispositivo eletro-mecânico* e *um sub-dispositivo* lógico.

Os módulos A e B são similares aos tamanhos das placas do *VMEbus* de dimensões 3,9 x 6,3 polegadas e 9,2 x 6,3 polegadas respectivamente. Os módulos C e D são baseados no padrão dos cartões Eurocard e tem 9 e 14 polegadas de altura respectivamente e 13 polegadas de fundo. O Eurocard C tem a mesma altura da placa B do VME, enquanto que o módulo D é três vezes mais alto.

2 *Sistemas e Subsistemas VXIbus*

Um sistema *VXIbus* pode ter até 256 endereços lógicos, com endereços lógicos distintos, incluindo um ou mais subsistemas *VXIbus*. Um subsistema *VXIbus* é alojado em um bastidor *VXIbus*, o qual é equivalente ao *rack* de cartões VME. O bastidor *VXIbus* pode ter no máximo 13 módulos *VXIbus*, sendo que o Slot 0 é o módulo central de controle e temporização e os Slots 1 até 12 são os módulos de instrumentos. Um subsistema *VXIbus* consiste de um módulo central atribuído para o Slot 0 com 12 módulos adicionais de instrumento, que são incluídos através de adaptadores e conectores P1, P2 e P3.

É importante entender a distinção entre um sistema *VXIbus* e um subsistema *VXIbus*, onde os módulos em um subsistema, compartilham dos mesmos sinais de sincronização no *backplane* (conectores P2 e P3), podendo ser sincronizado por qualquer um deles. Os módulos em um sistema, comunicam-se entre si, mas se eles são de diferentes subsistemas, não compartilham do mesmo sinal de sincronização do *backplane*.

Todos os módulos *VXIbus* são conectados ao *backplane* através do ***conector P1***, e opcionalmente através do ***conector P2 e P3***. O conector P1 provém 8 e 16 bits de capacidade de transferência de dados, o qual são baseados todos os protocolos de alto nível para *VXIbus*, os conectores P2 e P3 incrementam o sistema *VXIbus* adicionando:

- Transferências de dados de 32 bits em alta velocidade;
- Alimentação/Potência DC adicional;
- Sinais de sincronização para o sistema;
- Barramento analógico dedicado;
- Capacidade de auto-configuração;
- Recursos do sistema;
- Barramento local (comunicação módulo a módulo em alta velocidade).

O **conector P1** contém todos os sinais necessários para completar os 24 bits de endereço e 16 bits de dados junto com o controle, suporte a interrupção e tensões requeridas para implementar o sistema de comunicação. O **conector P2** é usado para a

expansão de endereço e dados para 32 bits sobre módulos do tipo B. Já o **conector P3** em módulos do tipo D expande ainda mais as características:

- *Clock* de 100 MHz;
- Capacidade de sincronização;
- 4 linhas de *trigger* ECL;
- 36 linhas de barramento local;
- Conjunto de linhas que permitem a comunicação assíncrona inter-módulos.

Os sinais no *VXIbus* são agrupados em 7 barramentos:

1. *Clock bus* : fornece dois sinais de *clock* e um de sincronismo: CLK10, CLK100, SYNC100.
2. *Star bus* : é composto de 24 pares de linhas de sinais bidirecionais. Dois pares (STARX+/STARX- e STARY+/STARY-) os quais conectam cada módulo ao módulo Slot 0. O nó Slot 0, por meio de uma matriz de chaveamento, pode-se conectar cruzando um par de linhas STAR com outro par qualquer. Sendo o retardo máximo entre a comunicação do Slot 0 com qualquer um dos módulos de 5ns, o tempo máximo de propagação entre módulos é de 10ns. Esta capacidade de conexão cruzada permite ter elevadas velocidades de sinalização entre os módulos do *VXIbus*.
3. *Trigger bus* : têm 8 linhas de *trigger* TTL e 6 linhas ECL. Todas as linhas TTL e 2 linhas ECL estão localizadas no conector P2, as 4 linhas restantes estão localizadas no conector P3. As linhas de *trigger* são de uso geral, sendo usadas individualmente ou em conjunto, para a transferência paralela de dados.
4. *Local bus* : fornece uma comunicação privada entre um módulo e o módulo adjacente, exceto para os dois Slots da ponta o Slot 0 e o Slot 12, cada Slot têm dois barramentos locais separados: um para os módulos a sua direita e outro para os módulos a sua esquerda. O local bus têm 5 classes de sinais: TTL, ECL, Analógico Baixo, Analógico Médio, Analógico Alto. O local bus é um versátil método de intercomunicação entre os módulos do *VXIbus*. Pelo uso do local bus, evita-se o uso de cabeamento de interligação entre os módulos pelo painel frontal. Os módulos fazem uso do *local bus* dependendo de sua necessidade específica.

5. *Analog Sumbus* : o sinal Analog Sumbus é uma somatória nodal dos sinais, em todo o barramento *VXIbus*. Um dos usos deste sinal, é a geração de forma de onda complexa, por exemplo, as saídas de geradores de forma de onda arbitrária combinadas aditivamente pelo barramento *Sumbus*, gerando uma forma de onda que é a superposição de cada uma delas.
6. *Module Indentification Bus* (MODID) : a linha MODID permite que o módulo Slot 0, determine onde cada módulo está alocado logicamente no *backplane* do *VXIbus*, saindo do módulo Slot 0, 11 linhas de controle uma para cada módulo.
7. *Power Distribution Bus* : o barramento de alimentação distribuída fornece até 1000W de potência, com as seguintes saídas: +5V, -5.2V, +12V, -12V, +24V e -24V.

3 Dispositivo VXIbus

Um dispositivo *VXIbus*, sendo o elemento mais inferior da hierarquia tem um conjunto de registradores de configuração, todos totalmente acessíveis por P1 sobre o qual permite o sistema identificar o dispositivo, classes, modelo e fabricante, espaço de endereço (A16, A24, A32), e memória requeridos. Dispositivos com somente este mínimo nível de capacidade são chamados ***Register Based Devices***. Um exemplo, é o sistema de somente uma CPU que pode implementar instrumentos usando somente *Register Based Devices* com protocolos de dispositivo específico sobre como a CPU comunica-se com cada um destes instrumentos.

VXIbus também define ***Memory Devices***, que podem ser identificados como RAM, ROM, ou outro tipo de memória e ser configurado em blocos contínuos de memória baseado na velocidade ou tipo de memória. Os protocolos dos dispositivos *VXIbus* definem as porções do espaço de endereço para que não haja conflito entre os módulos. Vários dispositivos podem estar sobre um módulo simples, e um simples dispositivo pode consistir de múltiplos módulos.

Se um alto nível de capacidade de comunicação entre módulos é desejado no sistema, *VXIbus* define uma classe de dispositivos chamados ***Message Based Devices***. Eles são requeridos para ter registradores de comunicação que são acessíveis por outros

módulos no sistema. Cada dispositivo no sistema pode então usar os protocolos específicos de comunicação assim como o ***Word Serial Protocol*** do *VXIbus* para comunicar-se com outros dispositivos. Um bom exemplo deste tipo de sistema, onde cada instrumento é um *Message Based Device* e é, portanto, capaz de receber instruções de um *host* ou uma interface comum de *host*, é um sistema de várias CPU's. Portanto, com a adoção de protocolos de comunicação, assim como *Word Serial*, diferentes fabricantes podem construir alguns destes instrumentos e garantir a compatibilidade com o resto do sistema. Alto nível de protocolos de comunicações classificados como ***Protocolo de Memória Compartilhada*** podem ser definidos usando estas plataformas.

4 Operação dos Dispositivos

Os dispositivos são os mais baixos componentes lógicos no sistema *VXIbus*. Normalmente um dispositivo consistirá de um módulo *VXIbus*. Porém, são permitidos dispositivos com múltiplas placas e placas com múltiplos dispositivos. Para cada dispositivo no sistema, existe uma única lógica de endereço associada. Exemplos de dispositivos são computadores, multímetros, multiplexadores, osciladores, interfaces operadoras, e contadores.

A primeira diferença entre os módulos VXI e VME é que o módulo VXI tem 4 registradores de 16 bits para configuração. Cada dispositivo VXI, têm sua "assinatura" sob um endereçamento lógico de 8 bits. Este endereço lógico pode ser estabelecido manualmente (via chave DIP) sob os módulos ou pode ser automaticamente configurado pelo sistema quando de sua inicialização. Os 8 bits lógicos de endereçamento especificam, onde o registrador do dispositivo encontra-se alocado no espaço de endereçamento de 16Kbits. Todo registrador de configuração dos dispositivos, permite ao sistema rapidamente identificar cada módulo, tipo, modelo, fabricante, espaço de endereçamento e memória requerida. Os registradores de configuração permitem ao "*resource manager*" configurar automaticamente o dispositivo quando ligar o equipamento.

4.1 Dispositivos Baseados em Mensagens

Suportam os protocolos de configurações e comunicações do *VXIbus*. Esta categoria inclui somente dispositivos com elementos *Commander* e/ou *command based servant*. Exemplos *de Message Based Devices* são alguns dispositivos com inteligência local que requerem um certo nível de capacidade de comunicação como DMM's, Analisadores de Espectro, Controladores de Espectro, dispositivos de interface 488-*VXIbus*, etc.

4.2 Dispositivos Baseados em Registros

Suportam mapas de registros *VXIbus*, mas não protocolos de comunicação *VXIbus*. Tipicamente *Register Based Devices* são simples, rentáveis, como cartões de I/O digitais, simples cartões de interface serial e em geral alguns cartões que requerem pequena ou nenhuma inteligência local.

4.3 Dispositivos de Memória

Possuem registros de configuração e contém certos atributos de um dispositivo de memória como tipo e tempo de acesso, etc. mas não tem outros registros ou protocolos *VXIbus* definidos. Cartões de memória RAM e ROM são desta categoria.

4.4 Dispositivos Extendidos

São úteis para dispositivos *VXIbus* que têm registros de configuração podendo assim ser identificados pelo sistema. Esta categoria de dispositivos permitirá por definição das futuras classes de dispositivos suportar altos níveis de compatibilidade.

4.5 Dispositivos Híbridos

São dispositivos *VMEbus* compatíveis que baseiam-se em torno de dispositivos *VXIbus* e têm a capacidade de comunicar-se com eles ou fazer uso deles, mas não obedece os requerimentos dos dispositivos *VXIbus*. Um exemplo são placas *VMEbus* que com o apropriado *software* faz uso de dispositivos *VXIbus*.

5 *Configuração do sistema VXI*

A configuração do sistema VXI é composta de:

- Controlador rodando um programa aplicativo;
- Um ou mais subsistemas VXI contendo módulos de instrumentos VXI;
- Interface entre o controlador e o *VXIbus*.

O controlador de um sistema VXI, pode ser utilizado como um *desktop*, com interface GPIB ou MXI ou como um sistema incorporado ao sistema comumente chamado de VXI *Embbeded*.

5.1 GPIB

A forma mais comum de interfaceamento entre o computador e os instrumentos de diferentes fabricantes, chamado de (*General Purpose Interface Bus*), oficialmente conhecida como IEEE-488.2. Esta interface é disponível para vários tipos de computadores bem como os interfaces *drivers* para diferentes linguagens de programação. Muitos *softwares* para procedimentos de testes automáticos, são projetados para suportar a interface GPIB. Com capacidade máxima de interfaceamento de 15 dispositivos, distância de 20 metros, e taxa de transferência de 1Mbyte/segundo com 8 bits de transferência.

5.2 MXIbus

O *Multi-system eXtension Bus*, foi projetado pela *National Instrument* para interfacear o *VXIbus*. Interface paralela com alta velocidade de transferência de dados entre o *VXIbus* e o computador de controle. O *MXIbus* têm capacidade máxima de driver de instrumento para 8 dispositivos, distância máxima de 20 metros e taxa de transferência de 20Mbytes/segundo com 32 bits de transferência.

5.3 VXI Embbeded

Controlador modular para sistemas *VXIbus* baseado em plataformas 486 ou Pentium operando entre 100 e 133 MHz de *clock*, com acesso direto ao *VXIbus*, taxa de transferência de 40Mbytes/segundo a 32 bits. A vantagem de se utilizar um controlador embutido, está mostrado nas figuras a e b. Todas estas interfaces, ocupam o Slot 0, no

bastidor VXI, também elas provêm os sinais básicos para o controle do sistema: CLK10, CLK100 e SYNC100.

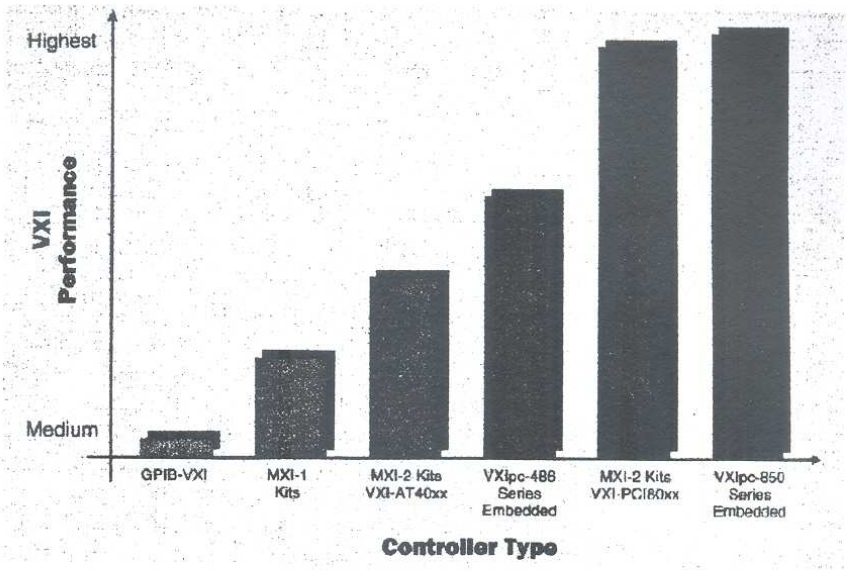


Figura a - Performance / Tipo de Controlador

6 *Abordagem de software para VXI*

O *software* é a consideração mais importante aplicada ao sistema VXI. Tornando regra básica ao desenvolvimento de sistemas automáticos de aquisição de dados e ao controle de instrumentação, sua qualidade e flexibilidade, determinam a performance final do sistema. Sendo assim, foi criado o *VXIplug&play System Alliance*, tendo como meta fundamental oferecer uma arquitetura específica que facilite a tarefa de configuração, programação e integração de sistemas de testes baseados em *VXIbus*. A figura b ilustra a estrutura integral de sistemas VXI.

No núcleo desta nova solução encontra-se a Arquitetura de *Software* de Instrumento Virtual - VISA. A VISA oferece compatibilidade de protocolo para os programas em uso e Ambiente de Desenvolvimento de Aplicação (ADE), operando em diversas interfaces utilizadas atualmente. De igual importância ao *VXIplug&play* é o conceito de estrutura de sistemas. Para lidar com várias plataformas de sistemas operacionais utilizadas atualmente, esta aliança segmentou o domínio do *VXIplug&play* em três estruturas de sistemas:

- WIN - para Windows convencional da Microsoft;

- GWIN - para ambientes gráficos baseados em Windows;
- DOS - para ambientes de programação baseados em DOS da Microsoft.

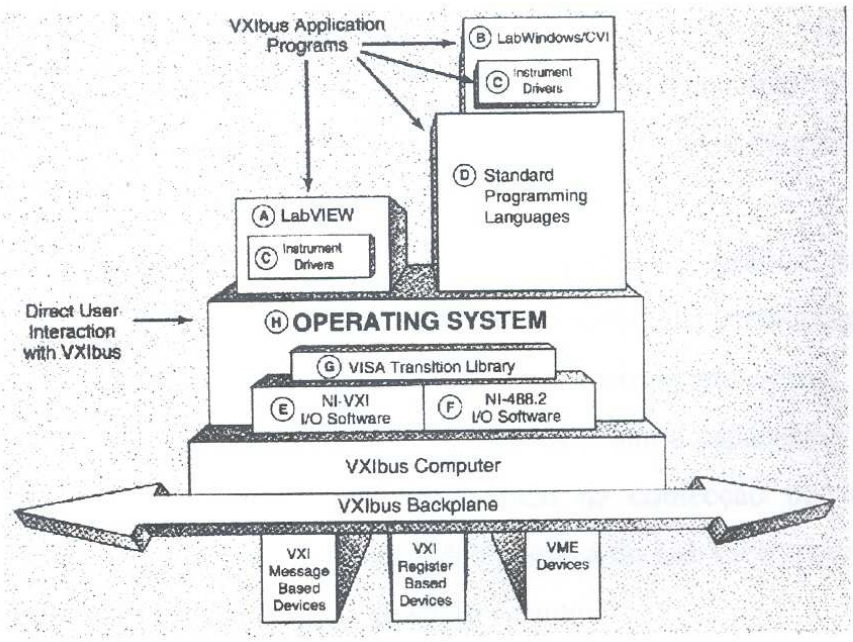


Figura b : Estrutura VXI

Cada estrutura define os requisitos básicos para programas aplicativos, ADE, computadores, instrumentos, *mainframes*, etc. Os esforços estão sendo canalizados na estrutura WIN permitindo aos usuários opções em ferramentas de programação, conhecidos como WIN *Frameworks*. O WIN *Framework* está no centro do VXIplug&play, provendo as características necessárias para suportar uma larga variedade de *software* aplicativos, bem como propicia um excelente ponto de partida para migração de outros futuros *Frameworks* tal como WIN32 (Windows 95 - NT) e HP-UX. Uma das principais características do WIN *Frameworks* é seu *Run-Time linking*, o qual acessa os recursos dos instrumentos e componentes de *software*.

O *VXIplug&play* carrega uma biblioteca de *softwares* para controle do instrumento chamado "*Instrument Driver*", O ID é um pacote de rotinas de *softwares*, que controlam um instrumento programável, cada rotina corresponde a uma operação programada, tal como configuração, leitura, escrita e *trigger* do instrumento. O ID simplifica os controles do instrumento e reduz o desenvolvimento de programas de testes, eliminado a necessidade de conhecer o protocolo de cada instrumento. Incluindo códigos fontes, DLL's (*Dynamically Linked Library*) e um *software* do painel frontal do instrumento

executável, podendo operá-lo sem qualquer programação do mesmo. O Painel Frontal (SFP) é análogo ao painel físico do instrumento, sendo a interface do usuário com o instrumento físico.

As linguagens indentificadas para operar o VXIplug&play, compatíveis com o WIN Framework, que podem chamar um DLL do Microsoft Windows, e interfacear com o ID ou VISA:

- **LabView** : Revolucionária linguagem de programação gráfica para controle, aquisição, análise e apresentação de dados, oferecendo uma metodologia inovadora que permite ao usuário montar graficamente o seu instrumento virtual. Não fazendo uso de linguagem de programação em C, o LabView tem a capacidade de utilizar os DLL's dos drivers de instrumentos, facilitando na confecção do instrumento e reduzindo os custos de desenvolvimento. A biblioteca do LabView, consta de mais de 500 drivers de instrumentos de mais de 45 fabricantes.
- **LabWindows/CVI** : LabWindows/CVI (C para Instrumentação Virtual), é funcionalmente equivalente ao LabView. A diferença reside no fato de os códigos gerados no LabWindows/CVI serem em C e no LabView serem diagramas de blocos gráficos. Ver documento em anexo.
- **Borland C++ / Microsoft Visual C++** : Linguagens Avançadas em C, podendo usar os formatos DLL e ANSI C dos drivers de instrumentos, dando total suporte ao desenvolvimento de aplicativos que provêm interfaceamento gráfico com o usuário.
- **Microsoft Visual Basic** : Provendo acesso ao DLL no Framework, o Visual Basic, não têm suporte direto ao controle de instrumentos. Pode-se criar um Painel Frontal, mas necessita escrever-se os códigos dos drivers de instrumentos por meio de outra linguagem.
- **Hewlett-Packard HP-VEE** : O HP-VEE, não usa a linguagem de programação C, mas pode acessar o DLL dos drivers de instrumentos, permitindo o desenvolvimento de interface gráfica com o usuário. Tendo uma biblioteca de componentes, para interfaceamento gráfico, suporta uma extensiva lista de instrumentos.