DESENVOLVIMENTO E IMPLEMENTAÇÃO DE UMA FERRAMENTA COMPUTACIONAL APLICADA NO PROCESSO DE IDENTIFICAÇÃO DE SISTEMAS EM AMBIENTES FIELDBUS FOUNDATION.

Márcio José da Cunha

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Mestre em Engenharia Mecânica.

ORIENTADOR: Prof. Tit. Mário Pinotti Júnior

DEDALUS - Acervo - EESC

31100052128

São Carlos
2004

Quil. TESE-EESC
Con. 1752 1º
Tombo T 086/05
Sieno 144 3993

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

C972d Cunha, Márcio José da
Desenvolvimento e implementação de uma ferramenta
computacional aplicada no processo de identificação de
sistemas em ambientes Fieldbus Foundation / Márcio José
da Cunha. -- São Carlos, 2004.

Dissertação (Mestrado) -- Escola de Engenharia de São
Carlos-Universidade de São Paulo, 2004.
Área: Engenharia Mecânica.
Orientador: Prof. Tit. Mário Pinotti Júnior.

1. Identificação de sistemas. 2. Estimação
paramétrica. 3. Foundation fieldbus. 4. OPC. I. Título.

FOLHA DE JULGAMENTO

Candidato: Bacharel **MARCIO JOSÉ DA CUNHA**

Dissertação defendida e julgada em 06-10-2004 perante a Comissão Julgadora:

Prof. Titular **MARIO PINOTTI JUNIOR (Orientador)**
(Escola de Engenharia de São Carlos/USP)

aprovado

Prof. Dr. **GLAUCO AUGUSTO DE PAULA CAURIN**
(Escola de Engenharia de São Carlos/USP)

aprovado

Prof. Dr. **EDÍLSON REIS RODRIGUES KATO**
(Centro Universitário de Araraquara/UNIARA)

aprovado

Prof. Associado **JONAS DE CARVALHO**
Coordenador do Programa de Pós-Graduação
em Engenharia Mecânica

Profa. Titular **MARIA DO CARMO CALIJURI**
Presidente da Comissão de Pós-Graduação

Agradecimentos

Primeiramente, agradeço a Deus pela oportunidade.

Aos meus pais, Carlos e Antonia, meu irmão Marcelo e a minha querida cunhada Regiane pela ajuda, amor e carinho. A Ledmila pelo amor, afeto e ajuda. Também dedico esse trabalho a todos os meus familiares e amigos.

Ao amigo e Prof. Tit. Mário Pinotti Junior pela orientação, confiança, amizade e ajuda nos momentos difíceis.

Ao pessoal do Laboratório de Eletrônica Embarcada, em especial aos Engenheiros Carlos Eduardo Milhor, Dennis Brandão, Josemar dos Santos, Fabiano T. Costa, Álvaro M. B. Tibiriçá, Cristiano Tibiriçá, ao físico Jaime Marini Junior e ao Prof. Dr. Luis Carlos Passarini pela dedicação e amizade. Também os meus agradecimentos ao Prof. Dr. Glauco A. P. Caurin, pela ajuda e amizade.

Em especial aos amigos dos outros laboratórios, Ricardo A. G Coppola, João Paulo Godoy, Sandro Guimarães de Souza, Romeu Roni Cavalcante e a Maximiliano dos Anjos Azambuja (o AZA) pela ajuda e amizade.

A CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pelo auxílio financeiro, e à Escola de Engenharia de São Carlos da Universidade de São Paulo pelo apoio acadêmico e pela adequada estrutura de pesquisa.

Ao departamento de desenvolvimento eletrônico da Smar Equipamentos Industriais LTDA.

SUMÁRIO

LISTA DE FIGURAS	I
LISTA DE TABELAS	III
LISTA DE ABREVIATURAS E SIGLAS	IV
LISTA DE SÍMBOLOS	VI
RESUMO	VII
ABSTRACT	VIII
1 INTRODUÇÃO	9
1.1 OBJETIVOS DO TRABALHO	11
1.2 ORGANIZAÇÃO DO TRABALHO	12
2 REVISÃO BIBLIOGRÁFICA	13
3 O PROTOCOLO <i>FOUNDATION FIELDBUS</i>	18
4 O PADRÃO DE COMUNICAÇÃO <i>OPC</i>	22
4.1 HISTÓRICO	22
4.2 HISTÓRICO DO PADRÃO <i>OPC</i>	24
4.3 FUNDAMENTOS DO PADRÃO <i>OPC</i>	25
4.4 FUNCIONALIDADE DO PADRÃO <i>OPC</i>	27
5 IDENTIFICAÇÃO DE SISTEMAS	34
5.1 ETAPAS DA IDENTIFICAÇÃO DE SISTEMAS	35
6 IDENTIFICAÇÃO DE SISTEMAS APLICADA EM ESTRUTURAS MATEMÁTICA DE VARIÁVEIS DE ESTADO	42
7 TESTES E RESULTADOS	45
7.1 APLICAÇÃO DO SINTONIZADOR NO PROCESSO DE IDENTIFICAÇÃO DE SISTEMAS	45
SIMULADOS	45
7.2 APLICAÇÃO DO SINTONIZADOR NO PROCESSO DE IDENTIFICAÇÃO DE SISTEMAS EM	48
AMBIENTE <i>FIELDBUS</i>	48
8 CONCLUSÕES	56
8.1 SUGESTÕES PARA TRABALHOS FUTUROS	57
APÊNDICE A – O APLICATIVO SINTONIZADOR	58
ANEXO A – FUNDAMENTOS TEÓRICOS	65
ANEXO B – TEORIA DE CONTROLE	68

LITERATURA REFERENCIADA	76
LITERATURA CONSULTADA	81

LISTA DE FIGURAS

Figura 01 - Estrutura de camada do protocolo Fieldbus Foundation.	19
Figura 02 - Acesso aos dados dos dispositivos através de Módulos.	23
Figura 03 - Acesso aos dados dos dispositivos através do padrão <i>OPC</i> .	23
Figura 04 – Objeto COM.	26
Figura 05 – Funcionalidade do objeto COM.	27
Figura 06 – Objeto DCOM.	27
Figura 07 – Forma de acesso de uma aplicação Cliente <i>OPC</i> .	28
Figura 08 – Objeto <i>OPC Groups</i> e objeto <i>OPC Items</i> .	29
Figura 09 – Estrutura funcional do padrão <i>OPC</i> .	30
Figura 10 – Interfaces do objeto <i>OPC Client</i> .	31
Figura 11 – Interfaces do objeto <i>OPC Server</i> .	31
Figura 12 – Interfaces do objeto <i>OPC Group</i> .	32
Figura 13 – Representação de um sistema dinâmico.	34
Figura 14 – Etapas da Ciclo da Identificação de Sistemas.	41
Figura 15 – Sistema Dinâmico Simulado.	45
Figura 16 – Resposta do Sistema Simulado a uma Entrada Degrau Unitária.	46
Figura 17 – Resposta do Sistema Identificado.	47
Figura 18 – Erro Quadrático Médio de Resposta do Sistema Identificado.	47
Figura 19 – Representação do Forno Elétrico.	48
Figura 20 – Representação do Túnel de Aquecimento.	49
Figura 21 – Conexão com Servidor <i>OPC</i> da Rede <i>Fieldbus</i> .	49
Figura 22 – Aquisição de Dados Experimentais na Rede <i>Fieldbus</i> .	50
Figura 23 – Resposta do Forno Elétrico a uma Entrada Degrau Unitária.	51
Figura 24 – Resposta do Sistema Identificado com dados Experimentais do Forno Elétrico.	52
Figura 25 – Erro Quadrático Médio da Resposta do Sistema Identificado com a Resposta do Forno Elétrico.	52
Figura 26 – Resposta do Túnel de Aquecimento a uma Entrada Degrau Unitária.	53
Figura 27 – Resposta do Sistema Identificado a partir de Dados Experimentais do Túnel de Aquecimento.	54
Figura 28 – Erro Quadrático Médio da Resposta do Sistema Identificado com a Resposta do Túnel de Aquecimento.	54

Figura 29 – Tela inicial do aplicativo Sintonizador.	58
Figura 30 – Tela de conexão com servidores <i>OPC</i> .	59
Figura 31 – Tela de Identificação de Sistemas <i>Off-Line</i> .	60
Figura 32 – Tela de Identificação de Sistemas <i>On-Line</i> .	60
Figura 33 – Tela do Módulo de Aquisição de Dados.	61
Figura 34 – Tela com as Especificações para Aquisição de Dados.	62
Figura 35 – Tela de Configurações.	62
Figura 36 – Tela de configuração das matrizes do bloco Estimador de Estados.	63
Figura 37 – Tela da Configuração dos Requisitos de Desempenho do Sistema em Malha Fechada.	64
Figura 38 – Resposta do Sistema em Malha Fechada.	64
Figura 39 – Diagrama de blocos para sistemas de controle discreto via Espaço de Estado.	70
Figura 40 – Diagrama de blocos para sistemas de controle contínuos via Espaço de Estados.	70
Figura 41 – Realimentação de Estados.	71
Figura 42 – Observação de Estados.	73
Figura 43 – Realimentação de estados estimados.	74
Figura 44 – Realimentação de estados estimados com entrada referência.	75

LISTA DE TABELAS

Tabela 1- Comparativo entre Métodos de Modelagens de Sistemas.

11

LISTA DE ABREVIATURAS E SIGLAS

ARMAX	- <i>Auto-Regressive Moving eXogenous Model</i> (Modelo linear, auto-regressivo, de média móvel e com entrada externa)
ARX	- <i>Auto-Regressive eXogenous Model</i> (Modelo linear, auto-regressivo e com entrada externa)
CAR	- <i>Continuous-time Auto Regressive</i> (Modelos auto-regressivos contínuos)
COM	- <i>Component Object Model</i> (Modelos de Componentes Objetos)
CORBA	- <i>Common Object Request Broker Architecture</i> (Padrão de gerenciamento de objetos)
DCOM	- <i>Distributed Component Object Model</i> (Modelos de componentes objetos distribuídos)
DCS	- <i>Distributed Control System</i> (Sistemas de controle distribuído)
DDE	- <i>Dynamic Data Exchange</i> (Troca dinâmica de dados)
DLL	- <i>Data Link Layer</i> (Camada de enlace de dados)
DLL	- <i>Dynamic Link Library</i> (Bibliotecas de vinculação dinâmica)
ERP	- <i>Enterprise Resource Planning</i> (Sistemas de gestão empresarial)
FBAP	- <i>Function Bloch Application Process</i> (Processo de aplicação de blocos funcionais)
FCS	- <i>Fieldbus Control System</i> (Sistema de controle <i>fieldbus</i>)
GUID	- <i>Globally Unique Identifier</i> (Identificadores globalmente exclusivos)
HMI	- <i>Human Machine Interface</i> (Interface homem-máquina)
IEC	- <i>International Eletrotechnic Commision</i>
IEEE	- <i>The Institute of Electrical and Eletronics Engineers</i>
ISA	- <i>International Society for Measurement and Control</i>
ISO/OSI	- <i>International Standard Organization / Open System Interconnection</i>
ISP	- <i>Interoperable System Protocol</i>
LAS	- <i>Link Active Scheduler</i> (Escalonador ativo de conexões)
LS	- <i>Least Square</i> (Método dos mínimos quadrados não recursivo)

<i>MIMO</i>	- <i>Multiple Input Multiple Output</i> (Sistemas com múltiplas entradas e múltiplas saídas)
<i>MISO</i>	- <i>Multiple Input Single Output</i> (Sistemas com múltiplas entradas e uma saída)
<i>na</i>	- Número de coeficientes de saída da estrutura <i>ARX</i>
<i>NARMAX</i>	- <i>Non Linear Auto-Regressive Moving eXogenous Model</i> (Modelo não-linear, auto-regressivo, de média móvel e entrada externa)
<i>NARX</i>	- <i>Non Linear Auto Regressive eXogenous Model</i> (Modelo não-linear, auto-regressivo e entrada externa)
<i>nb</i>	- Número dos coeficientes de entrada da estrutura <i>ARX</i>
<i>OLE</i>	- <i>Object Linking Embedding</i> (Ligação e incorporação de objetos)
<i>OPC</i>	- <i>Ole for Process Control</i> (Padrão <i>OLE</i> aplicados em Controle de Processos)
<i>RELS</i>	- <i>Recursive Extensible Least Square</i> (Mínimos quadrados recursivo estendido)
<i>RLS</i>	- <i>Recursive Least Square</i> (Método dos mínimos quadrados recursivo)
<i>SCADA</i>	- <i>Supervisory Control & Data Acquisition Systems</i> (Sistemas supervisórios de controle e aquisição de dados)
<i>SISO</i>	- <i>Single Input Single Output</i> (Sistemas com uma entrada e uma saída)
<i>TI</i>	- Tecnologia de Informação
<i>UTC</i>	- <i>Universal Time Coordinated</i> (Coordenada universal de Tempo)
<i>XML</i>	- <i>Extensible Markup Language</i> (Linguagem de marcação extensível)

LISTA DE SÍMBOLOS

λ - pólos do sistema dinâmico

RESUMO

CUNHA, M. J. (2004). *Desenvolvimento e Implementação de uma Ferramenta Computacional Aplicada no Processo de Identificação de Sistemas em ambientes Fieldbus Foundation*. São Carlos. 81p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo.

Técnicas experimentais de Identificação de Sistemas de controle têm despertado interesse na indústria, devido a sua facilidade em se ajustar modelos matemáticos, facilitando a formulação e a resolução de problemas de controle de processos. É proposto o uso de uma técnica experimental de Identificação de Sistemas, utilizando a estrutura matemática linear ARX. Os parâmetros da estrutura matemática ARX são estimados por meio do algoritmo dos Mínimos Quadrados Recursivo (RLS). A comunicação e a aquisição de dados de redes *fieldbus* é feita através do padrão de comunicação OPC.

Palavras chave: identificação de sistemas, estimação paramétrica, fieldbus foundation, OPC.

ABSTRACT

CUNHA, M. J. (2004). *Development and Implementation of a Computational Tool Applied to the System Identification Process in a Fieldbus Foundation Environment*. M.Sc Dissertation - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2004.

Experimental system identification techniques are considered interesting by industrial sector due to the simple approach to adjust mathematical models, making it easy the formulation and the solution of process control problems. In this work a computational tool is proposed, the Sintonizador, based on the experimental technic of System Identification, using the linear mathematical structure *ARX* (*Auto-Regressive with eXogenous inputs*). The *ARX* structure parameters are estimated by *RLS* (*Recursive Least Square*) algorithm. The data communication and data aquisition of the *fieldbus* network has been done through of the *OPC* communication standard.

Keywords: system identification, parametric estimation, fieldbus foundation, OPC.

1 INTRODUÇÃO

A Identificação de Sistemas têm despertado interesse na indústria, devido a sua facilidade em ajustar modelos matemáticos, facilitando a formulação e a resolução de problemas de controle de processos. A Identificação de Sistemas é classificada como parte da modelagem matemática.

A modelagem matemática, segundo IWASE et al. (2002), é um tipo de modelo matemático, onde este é utilizado para descrever a dinâmica de um sistema, importante para o desenvolvimento de sistemas de controle. Os modelos matemáticos, segundo SANTOS (2000), são obtidos através de dois modos: o primeiro modo, o analítico, conhecido também como modelagem teórica, é constituído por um modelo matemático formado por uma ou mais equações diferenciais e (ou) equações algébricas. O segundo modo, o experimental baseia-se em um modelo matemático obtido através de experimentos com informações de entrada e saída de um determinado processo dinâmico real, e tal modelo é representado por meio de equações diferenças.

Um modelo analítico é composto por descrições físicas do sistema, isto é, o modelo é descrito com base nas leis teóricas (leis da mecânica, leis da física, leis da termodinâmica, etc) e leis empíricas necessárias para descrever o comportamento dinâmico do sistema, de modo que tal abordagem permite derivar modelos que descrevem a dinâmica interna do sistema, além da relação entrada-saída (RODRIGUES, 2000). Esse tipo de modelagem também é conhecido como modelagem caixa-branca (BOSSLEY, 1997), pois descreve o comportamento dinâmico de um fenômeno ou de um sistema por aproximações analíticas, através de equações constitutivas. Sua principal vantagem está no fato do modelo possuir uma interpretação física clara.

Em contra partida, como ressalta SANTOS (2000), este tipo de modelo possui algumas desvantagens em relação à complexidade de sua estrutura, ou seja, esse modelo pode envolver dezenas ou até de centenas de equações constitutivas, e geralmente esses modelos são contínuos no tempo, ao passo que as variáveis de saída dos sistemas e ações de controle são discretas no tempo. Para PAIVA (1999), esse tipo de modelagem, por ser a mais

tradicional, apresenta diversas dificuldades em relação ao tratamento de sistemas mais complexos, não-lineares, estocásticos ou variantes no tempo. Assim, a construção de modelos analíticos para sistemas dessa natureza é geralmente uma tarefa árdua, devido ao seu custo tanto para desenvolvimento quanto computacional.

Segundo RODRIGUES (2000), a modelagem pelas leis físicas de um sistema pode ser um processo complexo; quando o sistema torna-se grande, complexo ou quando aumenta a complexidade nas interações das variáveis do sistema, dificultando assim a sua descrição matemática. Conseqüentemente, conforme a complexidade do sistema aumenta, a precisão de um modelo utilizado para descrever o sistema com rigor fica restrita, diminuindo sua precisão no decorrer do processo.

Paralelamente a essas dificuldades citadas, PAIVA (1999) diz que a indústria tem demonstrado interesse em adquirir autonomia nos processos de produção. Tal autonomia está relacionada ao desenvolvimento de métodos automáticos de aquisição e de conhecimento do processo e a sua incorporação nos sistemas de produção, exigindo assim a utilização de técnicas mais sofisticadas.

No modo experimental de modelagem, também conhecido como modelagem experimental ou como Identificação de Sistemas é necessária a realização de experimentos e um modelo matemático é obtido através das informações de entrada e saída coletadas, em contraste com a modelagem analítica, onde esta é feita baseada em relações matemáticas e o conhecimento teórico do sistema ou do fenômeno a ser identificado (LJUNG, 1999).

Através da Identificação de Sistemas, não se faz necessário o conhecimento físico e ou teórico do sistema a ser identificado, pois o modelo matemático é obtido através de dados experimentais de entrada e saída do sistema a ser identificado, por isso o modelo pode ser classificado como um modelo do tipo caixa-preta. Modelos do tipo como caixa preta são ignorados os estados interiores (SJÖBERG *et al.*, 1994), (LJUNG, 1999), (SIMANI *et al.*, 2000).

Para SANTOS (2000), uma das grandes vantagens desta técnica é a facilidade em se ajustar o modelo matemático, facilitando a formulação e a resolução de problemas de controle de processos. Porém os parâmetros processados na Identificação de Sistemas não possuem significados físicos, caracterizando-se assim uma desvantagem.

Abaixo, na Tabela 1, são apresentadas algumas diferenças entre a modelagem teórica e a modelagem experimental, segundo SANTOS(2000):

Tabela 1- Comparativo entre Métodos de Modelagens de Sistemas (SANTOS, 2000).

Modelagem Teórica	Modelagem Experimental
Requer poucos experimentos, pois os experimentos são utilizados apenas para estimar os parâmetros desconhecidos do modelo;	Requer experimentos extensos, por causa da necessidade em se confiar completamente nos experimentos para o desenvolvimento do modelo;
Informações internas sobre o estado interno do sistema;	Informações sobre o sistema que podem ser influenciadas por alguma ação de controle, chamadas de parte controláveis; parte das informações do sistema que pode ser medidas, parte observável;
Estabelece os fundamentos do funcionamento interno de um sistema, havendo a necessidade do conhecimento total do sistema;	Não é necessário o conhecimento do sistema, pois os dados da saída são obtidos através das mudanças das entradas, não havendo a necessidade de conhecimento dos fundamentos internos do sistema;
Esse tipo de modelagem não é utilizada para modelar os comportamentos desconhecidos ou alguns sistemas complexos;	Utilizado para modelagem de comportamentos desconhecidos ou sistemas complexos;
Geralmente criam-se modelos de sistemas lineares e não-lineares;	É necessário um método específico para gerar modelos não-lineares.

1.1 OBJETIVOS DO TRABALHO

Segundo BRANDÃO (2000), a razão pela qual não existe controle moderno em redes *Fieldbus Foundation*, é a inexistência de malhas distribuídas e de blocos funcionais disponíveis no mercado que possam realizar os processamentos adequados.

Tendo em vista a construção de um aplicativo computacional aberto aplicado a estratégias de controle adaptativo em ambientes *fieldbus* e visando o avanço tecnológico de ferramentas de apoio e de novas tecnologias em controle e instrumentação, neste trabalho é proposta a construção do aplicativo Sintonizador.

O primeiro objetivo desse trabalho, é a criação de um aplicativo computacional que possibilite a comunicação em ambientes *fieldbus* através do padrão de comunicação *OPC*, onde se é possível à manipulação (aquisição, edição) dos dados disponíveis. O protocolo *Foundation Fieldbus* foi utilizado para os testes realizados nesse trabalho.

O segundo objetivo, é implementar, com base nos dados coletados pelo aplicativo, a técnica experimental de Identificação de Sistemas. Deve-se considerar que as contribuições

deste trabalho servem de base para o desenvolvimento de técnicas de controle de processos cuja operação é baseada em um modelo fiel da planta, como por exemplo, técnicas de controle moderno e de controle avançado.

Este trabalho apresenta aspectos importantes da arquitetura *fieldbus* e suas implicações no projeto e implementação da técnica experimental de identificação de sistemas que podem ser aplicadas a estratégias de controle. Tais aspectos podem ser estendidos a aplicações correlatas.

1.2 ORGANIZAÇÃO DO TRABALHO

Esta dissertação está organizada da seguinte forma: no Capítulo 2, é feita a Revisão Bibliográfica das pesquisas recentes que estão relacionadas com o trabalho proposto. Capítulo 3 faz-se uma breve descrição sobre o protocolo para redes de chão-de-fábrica *Fieldbus Foundation*, das suas características técnicas e sua aplicabilidade. No Capítulo 4 é descrito o padrão de comunicação *OPC*, descrevendo o seu histórico, fundamentos, aplicabilidade e suas especificações existentes.

O Capítulo 5 faz-se uma breve descrição sobre a modelagem de sistemas teórica e a modelagem de sistemas experimental e um comparativo sobre suas principais diferenças. Também é descrita a técnica de Identificação de Sistemas utilizada neste trabalho.

No Capítulo 6 é composto pela descrição da aplicabilidade do sistema identificado pela Identificação de Sistemas para estruturas matemática de Variáveis de Estado. Os testes e os resultados são descritos no Capítulo 7. Por fim, no Capítulo 8, as conclusões e discussões sobre o trabalho realizado, onde são feitas as sugestões de novos trabalhos.

2 REVISÃO BIBLIOGRÁFICA

A evolução tecnológica dos sistemas industriais de instrumentação e de controle de processos obteve um crescimento considerável durante os últimos 20 anos, forçando mudanças em equipamentos e softwares configuradores e de supervisão, onde a integração entre ambos e a disponibilidade de dados dos dispositivos passaram a ser primordiais para se obter maior confiabilidade dos processos industriais.

No ambiente industrial com a introdução de microprocessadores nos dispositivos de campo, os transmissores inteligentes, possibilitaram que o desenvolvimento da comunicação digital ficasse inune a ruídos, tornando o pré-processamento e as medidas mais confiáveis e precisas.

Com isso, estratégias de controle distribuídas foram destinadas a processar dados dos sensores, tornando essas mais seguras e velozes, comparadas às estratégias de controle centralizadas (ZEILMANN, 2002). Estratégias de controle distribuídas possuem um algoritmo de controle dividido em módulos menores e são executadas dentro dos próprios dispositivos de campo.

Esses dispositivos de campo comunicam-se entre si, possibilitando a troca de dados de entradas e de saídas, aumentando a confiabilidade e a precisão de parâmetros essenciais no ambiente industrial.

Dentre as diversas arquiteturas propostas para a comunicação em barramento de campo, o protocolo *Fieldbus Foundation* foi concebido para garantir a interoperabilidade entre dispositivos de campo de diferentes fabricantes para a distribuição da instrumentação em plantas industriais.

O *Foundation Fieldbus* é uma rede de comunicação digital, bidirecional, multi-ponto, serial, sendo utilizada para interconexão de instrumentos de campo, por exemplo, sensores, atuadores e controladores. Disponibiliza também um conjunto de blocos funcionais padronizados, onde são implementados os algoritmos responsáveis pela manipulação de dados e aplicações de controle. Tais blocos podem funcionar isoladamente, ou conectado a outro bloco funcional.

Existem softwares utilizados para realizar a configuração e supervisão de processos executados em plantas industriais. Os sistemas supervisórios permitem a monitoração, tomadas de decisões e a configuração de estratégias de controle das variáveis disponíveis nas plantas industriais.

Atualmente, diversos softwares configuradores e de supervisão adotaram alguns padrões de comunicação de interface *Windows*, como o padrão *OPC* (do inglês *OLE for Process Control*) e o padrão *DDE* (do inglês *Dynamic Data Exchange*). JANKE (2000) afirma que o *OPC* tornou-se um padrão em ambientes industriais, pois considerado mais robusto e eficiente do que outros meios de comunicação entre aplicativos, por exemplo, padrão de *DDE*. Outro ponto importante, é que o *OPC* possibilita uma grande economia no tempo e em gastos com o desenvolvimento de novos aplicativos e na manutenção dos existentes. Assim, o padrão *OPC* provê a facilidade na transferência de dados entre os softwares e as informações do contidas em redes *fieldbus*, tornando tais informações acessíveis a redes corporativas dentro ou fora da indústria.

SANZ e ALONSO (2001), dizem que o processo de incorporação da Tecnologia de Informação (TI) em ambientes industriais gerou profundas modificações no desenvolvimento de sistemas aplicados em ambientes industriais e nos diferentes níveis de informação de processos industriais, tais como, gerência de campo, gerência de processos e na gerência de negócios. O *OPC* interage diretamente nos níveis da informação dos processos industriais, na Gerência de Campo, na Gerência de Processos e na Gerência de Negócios.

Na Gerência de Campo, os dados de controle e configuração, fornecidos pelos equipamentos inteligentes de campo, estão disponíveis para usuários ou mesmo para outras aplicações. Dentre as pesquisas aplicadas na gerência de campo, pode-se destacar BLEVINS et al (1999), SHIMANUKI (1999) e alguns fabricantes de dispositivos de chão-de-fábrica BERGE (2000). HADELLIS et al. (2002), propôs a integração ao acesso das diferentes bases de dados de diferentes protocolos através do *OPC*.

Na Gerência de Processos, os dados são fornecidos por sistemas de controle do tipo supervisórios (*SCADA*), que permite o controle descentralizado dos processos, que permitem a gerência efetiva e integrada, de todo o processo industrial. LING e YU (2001) propõem a construção de um software do tipo supervisórios (*SCADA*) baseado em *Ethernet Industrial*. A idéia da *Ethernet Industrial* é de prover uma estrutura homogênea de *hardware* (rede física) e *software* (protocolos), atendendo todos os setores de uma indústria, desde a área de escritórios até o ambiente industrial, possibilitando a integração total da corporação empresarial.

Para o acesso aos dados de uma determinada rede de chão-de-fábrica, neste trabalho utilizou-se o *OPC*. Dentre outras pesquisas pode-se destacar SENDELBACH (1999), BANGEMANN et al. (1998), SITAO (2000a), SITAO (2000b), JANKE (2000), KLAVSUTZ et al. (2000).

Atualmente é notável o aumento no número de ferramentas de apoio e de novas tecnologias em controle e instrumentação que podem ser aplicadas ao ambiente industrial. Dentre as pesquisas, BRANDÃO (2000) propôs a construção de um bloco funcional *Fieldbus Foundation* Estimador de Estados, visando à implementação de sistemas de controle moderno.

Dentre as estratégias de controle existentes, o controle adaptativo se distingue das outras estratégias principalmente pela capacidade única de tratar de problemas de variações relacionadas com o tempo. Não é raro que um sistema opere em situações variantes no tempo e que essas variações tenham efeitos importantes no desempenho do sistema. Além disso, variações na própria estrutura do processo podem também ser identificadas e compensadas nesse tipo de estratégia de controle.

Para acompanhar tais mudanças, bem como para o desenvolvimento de novas ferramentas de auxílio ao projeto de novos controladores e de estratégias e algoritmos de controle adaptativo, é necessário que se conheça o comportamento dinâmico real do sistema a ser controlado, ou seja, é necessário que se identifique o sistema.

A aplicação da teoria de identificação de sistemas em um ambiente de controle distribuído vem se tornando muito utilizada no ambiente industrial pela crescente demanda de técnicas de controle mais precisas e versáteis. Em ambientes *fieldbus*, onde todos os processamentos de variáveis são realizados no domínio discreto, é importante que o projeto de controladores e de algoritmos de identificação de sistemas seja estruturado neste domínio.

FURTADO et al. (2002), diz que a interpretação dos parâmetros do modelo no domínio discreto com relação ao correspondente modelo físico não é direta, portanto, é necessário identificar modelos no domínio contínuo no tempo, já que os mesmos possuem relação direta com os parâmetros de interesse. Essa transformação, de um modelo no domínio discreto para o domínio contínuo, é feita através do mapeamento do domínio da frequência para o domínio do tempo.

FURTADO et al. (2002) propôs ainda uma metodologia para a identificação de modelos contínuos, através do mapeamento de modelos discretos para o domínio da frequência. SÖDERSTRÖM et al. (1997), propôs a construção de um método de estimação paramétrica baseado no modelo linear, auto-regressivo, com entradas externas (*ARX*), onde se é possível estimar parâmetros para modelos em domínios contínuos, com base na estimação feita em

modelos no domínio discretos. Tal estimação paramétrica foi feita por meio de Variáveis Instrumentais.

FAN e SÖDERSTÖRM (1997) desenvolveram uma nova estrutura matemática para modelos no domínio contínuo baseado em regras de modelos auto-regressivos, conhecido como *CAR* (do inglês *Continuous-time Auto Regressive*), onde os parâmetros são estimados através do método dos Mínimos Quadrados não Recursivo (*LS*). Diversas pesquisas utilizam outros tipos de modelos, como JOHANSSON (1994) que se baseou em estruturas lineares, auto-regressivas, com média móvel e com entrada externa (*ARMAX*), para modelos no domínio contínuo. Essas implementações foram desenvolvidas para sistemas lineares. Já IWASE et al. (2002) propôs uma nova técnica de identificação de sistemas contínuos não-lineares. A não-linearidade era tratada como um problema de otimização aplicado a estruturas matemáticas *ARX*.

Para modelos com várias entradas e várias saídas (*MIMO*) SANTOS (2000), propôs a aplicação do método de identificação de sistemas para sistemas multivariáveis para processos químicos, processos contínuos de fermentação, e reações químicas. Os modelos utilizados foram às estruturas matemáticas *ARX* e *ARMAX*, e para a estimação paramétrica foram utilizados o algoritmo dos Mínimos Quadrados Recursivo (*RLS* – do inglês *Recursive Last Square*) e o método dos Mínimos Quadrados Recursivo Estendido (*RELS* do inglês *Recursive Extensible Least Square*). O autor afirma ainda que as estimativas de parâmetros feitas pelos modelos paramétricos obtiveram resultados razoáveis em alguns pontos de operação, uma vez que os sistemas identificados eram não-lineares. Para pesquisas futuras, SANTOS (2000) propõe a identificação de sistemas *MIMO* para sistemas não-lineares. Outras pesquisas envolvendo Identificação de Sistemas para sistemas multivariáveis estão em (TOUHAMI et al., 2000), (KAMWA, 2000), (RASMUSSEN e JØRGENSEN, 1999), (NINNESS et al., 1995) e (STOICA, 1999).

Pesquisas envolvendo Identificação de Sistemas aplicada a controladores tradicionais, como o *PID*, e no desenvolvimento de novos controladores, RIVERA e GAIKWAD (1996) propuseram a construção de um *PID* digital, onde os parâmetros do *PID*, *K*, *Ti* e *Td*, eram obtidos através dos modelos *ARX* e *ARMAX*. HUZMEZAN et al. (2001), utiliza modelos *ARX* para o desenvolvimento de um novo controlador de processos preditivo, o *MBPC* (do inglês *Model Based Predictive Controller*), desenvolvido para melhorar desempenhos de reatores, que às vezes, não são otimizados em controladores do tipo *PID*.

Dentre as pesquisas que envolvem a Identificação de Sistemas aplicados a Estimador de Estados, podemos destacar SIMANI et al. (2000), propôs a construção de um estimador de estados baseado em modelos *ARX*, para sistemas com múltiplas entradas e uma saída

(*MISO*), onde os parâmetros do modelo *ARX* são aplicados nas matrizes de estado do sistema.

Diversas pesquisas estão voltadas para a aplicação da Inteligência Artificial que valem como um método de identificação de sistemas ou até mesmo em outras áreas de aplicação em sistemas, por exemplo, controle de processos. PAIVA (1999) faz uma análise de aspectos funcionais relacionados a uma estrutura Neuro-Difusa, aplicada ao processo de identificação de sistemas. LENNOX et al. (2001), faz uma investigação sobre o uso de Redes Neurais Artificiais aplicadas em processos industriais, como monitoramento de processos, controle de processos e identificação de sistemas.

Conseqüentemente as técnicas para modelagem de sistemas dinâmicos foram aprimoradas para diversas classes de plantas industriais, bem como para outras áreas do conhecimento como a economia e a biologia. Tais técnicas também são utilizadas na identificação de sistemas multivariáveis e no desenvolvimento de novos controladores industriais (LJUNG, 1999) (SÖDERSTRÖM, 1991) (SANTOS, 2000).

A conexão entre o *OPC* e a Gerência de Negócios, acontece quando ocorre a integração dos processos de chão-de-fábrica com os setores corporativos da empresa, onde os dados fornecidos podem ser utilizados para aplicações cliente de modo a otimizar a gerência e a integração de todo o processo de manufatura (EMERSON, 2003).

Atualmente, com o crescimento das aplicações baseadas em Internet, surgiram pesquisas envolvendo acesso de dados via WEB. WANG e XIE (2002), KAPSALIS et al., (2002a), KAPSALIS et al. (2002b). YANG et al. (2003) propôs uma aplicação WEB para monitoramento e controle de plantas industriais onde os dados são disponibilizados através do padrão *OPC*. ZEILMANN (2002) propôs uma estratégia de controle e supervisão de processos industriais via Internet.

Existem diversas pesquisas envolvendo outras tecnologias de sistemas distribuídos, tais como SANZ e ALONSO (2001), que desenvolveram um padrão de comunicação baseado na tecnologia de sistemas distribuído *CORBA* (do inglês *Common Object Request Broker Architecture*), e KLEINES et al. (2001), que desenvolveu outro padrão de comunicação baseado em *OPC*, porém utilizando a linguagem *JAVA*, onde é possível a construção de aplicativos baseados em outros sistemas operacionais.

3 O PROTOCOLO *FOUNDATION FIELDBUS*

O *Foundation Fieldbus* consiste em um protocolo de rede concebido para a indústria de controle e processos, principalmente para se atingir algumas necessidades de seus usuários finais. Dentre algumas necessidades pode-se destacar que, o protocolo deve estar de acordo com o modelo *ISO / OSI* (do inglês *International Standard Organization / Open System Interconnection*), possuir segurança intrínseca em atmosferas perigosas, variáveis que expressam unidades de engenharia e variáveis com status que indiquem as condições das variáveis.

O protocolo foi proposto por volta de 1994 por uma fundação internacional conhecida como *Fieldbus Foundation* sediada nos Estados Unidos, composta pelas organizações *fieldbus* existentes, a *ISP* (do inglês *Interoperable System Protocol*), uma divisão da *WorldFIP* francesa juntamente com um grupo de empresas. E aliados aos organismos internacionais de normalização a *IEC* (do inglês *International Electrotechnic Commission*) e a Sociedade Internacional de Medidas e Controle (*International Society for Measurement and Control - ISA*), eles propuseram uma arquitetura baseada no modelo *ISO / OSI*, reduzido assim as três camadas, sendo o protocolo de camada física na norma *IEC 61158-2*, a camada de enlace no conjunto de normas *IEC FDIS 61158-3* e *4* e por fim a camada de aplicação na norma *IEC FDIS 61158-5* e *6*. Também está definida uma camada superior à camada de aplicação denominada camada de usuário, ou *User Layer*, onde nesta camada estão definidas as funções que determinam o controle e supervisão de um determinado processo.

Neste trabalho não será descrito, a estrutura das camadas ou seu completo algoritmo de funcionamento, esta descrição pode ser encontrada em *FOUNDATION* (2002). A Figura 01 apresenta um diagrama com a representação das três camadas que compõem o protocolo *Foundation Fieldbus*:

Figura 01 - Estrutura de camada do protocolo Foundation Fieldbus (FOUNDATION, 2002).

O nível físico define como as informações são transportadas fisicamente de um dispositivo para o outro. Isso inclui a forma do sinal, sua amplitude, tipos de fios, as distâncias máximas, impedâncias permitidas, etc.

No nível de enlace de dados, a mensagem contém o endereço de origem, o endereço de destino, os dados a serem transmitidos e uma sequência de verificação de moldura *FCS* (do inglês *Fieldbus Control System*), que garante a integridade dos dados.

O *FCS* é um polinômio calculado pelo dispositivo transmissor com todos os bits de moldura. O mesmo cálculo é executado pelo dispositivo receptor, que o compara com o cálculo recebido. Qualquer desvio entre os dois cálculos é indicativo de erro.

Na camada de aplicação o sistema de configuração do *fieldbus* atua e tem como objetivo a configuração dos dispositivos e o desenvolvimento do sistema. O desenvolvimento do sistema é baseado em Sistemas de Controle Distribuído (*DCS*), e depois de configurados os sistemas, os dispositivos são conectados aos Blocos Funcionais e as entradas e as saídas de cada dispositivo é configurado de acordo com um controle de estratégias contidas nos Blocos Funcionais.

Na camada de usuário, podemos destacar o *FBAP* (do inglês *Function Block Application Process*), que segundo FOUNDATION (2002), visam normalizar projetos e implementações de blocos funcionais de modo interoperável, implementações de blocos funcionais em diferentes transmissores e por fim a interoperabilidade em rede das aplicações de blocos durante a execução de um sistema.

Podemos destacar os três principais elementos desta camada: o *Resource Block*, o *Transducer Block* e os Blocos Funcionais.

O *Resource Block* tem como função descrever por completo o dispositivo para o sistema de controle. Essa descrição consiste na identificação do dispositivo, hardware, característica do dispositivo, disponibilidade de memória, proteção contra escrita, gerenciamento de falhas e alarmes.

O *Transducer Block* descreve por completo as informações sobre o sensor ou o atuador do dispositivo. É a ligação entre o meio físico e o mundo da informação; é responsável por informações funcionalidades específicas, por exemplo, uma medida da temperatura.

Os Blocos Funcionais possuem o papel de realizar o processo de automação básica, independentemente dos dispositivos do meio físico e da estrutura da rede. Os Blocos Funcionais processam parâmetros de entrada em função da sua configuração, gerando saídas que serão utilizadas por outros Blocos Funcionais (FOUNDATION, 2002).

Dentre os Blocos Funcionais podemos destacar o bloco de Entrada Analógica (*Analog Input - AI*), o de Saída Analógica (*Analog Output - AO*), os blocos responsáveis por realizar controles, como, por exemplo, Controle Proporcional, Integral e Derivativo, PID, o bloco funcional de Estimador de Estados (BRANDÃO, 2000).

Tendo em vista, que cada dispositivo deva ter suas próprias características, descrevendo as capacidades de seus próprios *Resource Block* e *Transducer Block*, as *Device Description (DD)* estabelecem uma forma automática de comunicação com os dispositivos da rede. Cada fabricante disponibiliza a *DD* de cada equipamento, ou também pode ser disponibilizada pela própria *Foundation Fieldbus*, permitindo acesso total a todas funções do dispositivo (EMERSON, 2003).

Para o desenvolvimento de cada *DD*, foi criada a *Device Description Language*, que é uma linguagem dedicada somente para a criação de novas *DD*, tendo sua base na linguagem C.

Quando se gera o código da *DD*, é gerado um código em *DLL*, e compilado em uma linguagem de máquina por uma ferramenta conhecida como *tokenizer*, sendo que este código só pode ser carregado pelo dispositivo, ou em uma mídia externa como Cds, disquetes, e depois lido diretamente (FOUNDATION, 2002).

O protocolo *Fieldbus Foundation* possibilitou a indústria alguns benefícios, dentre quais pode-se destacar (ZEILMANN, 2002) (FOUNDATION, 2002):

- **Menores Gastos com Equipamentos Supervisórios:** Os dispostos na rede oferecem autodiagnóstico;
- **Fácil Configuração do Sistema:** Interação entre o operador e o protocolo é feita através de blocos funcionais;

- **Menor Custo na Implantação de Novas Malhas de Controle:** Na implantação de novas malhas de controle, protocolo necessita apenas da adição de novos dispositivos de campo;
- **Alto Desempenho Tecnológico:** Esse ganho se dá pela tecnologia de ponta existente nos dispositivos, pois os dispositivos são considerados dispositivos inteligentes, pela capacidade de tomarem decisões em estratégias de controle.
- **Facilidade de Supervisão:** Os dados dispostos no protocolo *Fieldbus Foundation* são dispostos em um servidor de dados *OPC*.

O padrão de comunicação *OPC*, atualmente é adotado como o padrão de comunicação entre softwares supervisórios e o protocolo *Foundation Fieldbus*.

4 O PADRÃO DE COMUNICAÇÃO OPC

4.1 HISTÓRICO

Antigamente, o desenvolvimento de ferramentas e aplicativos utilizados para o acesso aos dados de diversos dispositivos, era feito de maneira separada das demais aplicações. Para tal operação, era necessária a construção de módulos, que eram softwares desenvolvidos pelos fabricantes dos dispositivos.

Porém, esta arquitetura gerava alguns inconvenientes (SHIMANUKI, 1999);

- **Esforços Duplicados:** Muitas vezes, para um dispositivo, era necessária a construção de mais de um módulo;
- **Inconsistência entre Drivers:** Isso ocorria quando um determinado dispositivo possuía mais de um módulo (*driver*), com isso as informações dos dispositivos eram inconsistentes, ou seja, cada módulo fornecia informações diferentes para um dispositivo;
- **Conflito de Acesso:** Não era possível acessar um dispositivo de forma simultânea, ou por mais de uma aplicação;
- **Suporte e Manutenção:** Quando o dispositivo passava por algum tipo de manutenção, o módulo também teria que ser reajustado.

A Figura 02, mostra a representação do acesso aos dados de diversos dispositivos através de módulos:

Figura 02 - Acesso aos dados dos dispositivos através de Módulos (OPC FOUNDATION, 2002).

Existia a necessidade de se criar um padrão que tornasse possível o acesso aos diversos mecanismos de comunicação e das diferentes fontes de dados, que tornasse mais fácil o acesso aos dados de dispositivos de chão-de-fábrica e outras fontes de dados. Foi através dessa necessidade que surgiu o padrão de comunicação *OPC*. O *OPC* fornece uma fronteira de acesso aos dados de qualquer fonte, dispositivos de chão-de-fábrica ou a outras fontes de dados. (OPC FOUNDATION, 2002) A representação esquemática do *OPC* é representada na Figura 03:

Figura 03 - Acesso aos dados dos dispositivos através do padrão OPC (OPC FOUNDATION, 2002).

O *OPC* define um mecanismo de comunicação entre componentes de automação, dispositivos de chão-de-fábrica, com os níveis da Tecnologia da Informação, interagindo

com as diferentes ferramentas de Tecnologia de Informação existentes, como sistemas de gestão *ERP* (do inglês *Enterprise Resource Planning*), sistemas Supervisórios (*SCADA*).

Outro ponto importante, é que o *OPC* possibilita uma grande economia no tempo e em gastos com o desenvolvimento de novos aplicativos e na manutenção dos existentes.

4.2 HISTÓRICO DO PADRÃO OPC

O desenvolvimento do *OPC* teve início pela *Microsoft*, juntamente com um grupo conhecido como *WinSEM*, que tinha o objetivo de realizar pesquisas em diversos setores, como no de ciências, engenharia e manufatura. Este grupo era composto por diversas empresas com o interesse comum de desenvolver aplicativos que utilizasse o sistema operacional *Windows* da *Microsoft* como plataforma base, através de um fórum de desenvolvimento. Porém, era provável que o desenvolvimento através de fórum de desenvolvimento era um passo lento, e não era eficiente. Tendo em vista este problema, cinco companhias decidiram tomar a iniciativa para desenvolver um esboço de um padrão que se tornasse aberto a indústria de Controle e Processos.

Com um grupo pequeno de desenvolvedores e limitando a funcionalidade do padrão especificada no início da criação, se tornava possível desenvolver o padrão em um pequeno espaço de tempo. O padrão foi desenvolvido em alguns anos e foi introduzido pelos líderes da indústria de Controle de Processos. O sucesso do desenvolvimento do *OPC* levou a *Microsoft* a sugerir esta técnica para outros esforços de desenvolvimento de outros padrões para tecnologia *Windows* (REHBEIN, 2000).

Um aglomerado de empresas ligadas à automação de processos e manufatura, criaram uma especificação para o padrão de Ligação e Incorporação de Objetos (*OLE*), aplicado para controle de processos, e publicaram para mais de 250 empresas. Seguindo as regras da *Microsoft*, aplicados a outro padrão para Modelos de Componentes e Objetos (*COM*), e este padrão iria administrar três diferentes níveis de informação:

- Coleta de dados de dispositivos inteligentes;
- Processos de dados para *DCS*;
- Sistemas Supervisórios (*SCADA*).

O padrão *OPC* surgiu para eliminar a necessidade de fabricantes em construir softwares para dispositivos individuais, pois este padrão funciona em diferentes plataformas e pode ser integrados com diversos objetos ou dispositivos (BLACKWELL, 1996).

Finalmente, ocorreu a automatização entre as aplicações e os controles de processos da indústria, ou seja, vários sistemas, vários dispositivos e vários módulos podendo se conectar

e se comunicar livremente, sendo que no passado o custo para se ter vários servidores, drivers, e interfaces era muito elevado.

Existe um grupo sem fins lucrativo, formado por diversas empresas, por fabricantes de dispositivos e desenvolvedores de software, conhecida como *OPC Foundation*, que é a responsável pelas especificações publicadas ou especificações em processo de aprovação do *OPC*.

Para FONSECA (2002), o principal objetivo do grupo é atender às necessidades da indústria, através do aprimoramento e ampliação do *OPC*. A estratégia adotada foi à criação de extensões a especificações existentes, definição e adição de novas especificações e de realizar modificações em versões *OPC* existentes. A primeira especificação produzida pelo grupo foi publicada em agosto de 1996, chamada de *OPC Specification 1.0*. Em setembro de 1997 foi liberada a primeira atualização da especificação *OPC* que passou a chamar *OPC Data Access Specification Version 1.0A*.

Atualmente existem as seguintes especificações *OPC* publicadas (IWANITZ, 2003):

- *OPC Overview*: Descrição geral dos campos de aplicação das especificações *OPC*;
- *OPC Common Definitions and Interfaces*: Estão definidas as funcionalidades básicas para as demais especificações;
- *OPC Data Access Specification*: Definição para leitura e escrita de dados em tempo real;
- *OPC Alarms and Events Specification*: Definição da interface de monitoramento de eventos;
- *OPC Historical Data Access Specification*: Definição das interfaces de acesso a dados históricos;
- *OPC Batch Specification*: Definição das interfaces para o acesso aos dados de processos por batelada;
- *OPC Security Specification*: Definição das interfaces que serão utilizadas para gerar regras de segurança;
- *OPC and XML*: São definidas as interfaces para a integração entre o padrão *OPC* e a linguagem *XML* para aplicações Web.

4.3 FUNDAMENTOS DO PADRÃO OPC

A tecnologia *OLE* foi desenvolvida pela *Microsoft* em meados de 1990, para suprir a necessidade de se integrar diferentes aplicações dentro da plataforma *Windows*. (FONSECA, 2002). Segundo CANTÙ (2002), essa tecnologia permite copiar dados de um aplicativo

servidor para um aplicativo cliente, como por exemplo, informações a respeito do servidor ou uma referência às informações armazenadas no registro do *Windows*.

A tecnologia *COM* define um padrão de comunicação entre um módulo cliente e um módulo servidor, por meio de uma interface específica. CANTÙ (2002) diz que o módulo indica um aplicativo ou uma biblioteca (*DLL*); podendo ser executados no mesmo computador ou em máquinas diferentes conectadas por meio de uma rede.

A Figura 04 representa o esquema funcional do padrão *COM*:

Figura 04 – Objeto *COM*.

O objeto *COM* possui componentes bases para o seu funcionamento através de interfaces, dos quais podemos destacar:

- *GUID*: É composta de um número com 16 *bytes* ou 128 *bits*, ou um número composto por 38 algarismos, que possuem o papel de fazer a identificação de uma classe de servidor registrados no sistema operacional *Windows*. O registro do *Windows* armazena esse número, com as identificações do servidor, que pode estar armazenados em arquivos do tipo *DLL* ou em arquivos executáveis;
- *Interface*: Além de representar a identidade do objeto, uma *Interface* representa um conjunto de funções que estão relacionadas com os objetos. Essas funções funcionam independentes do tempo e do espaço, sendo acessadas através do identificador, ou, acessado pela *GUID*;
- *IUnknown*: Essa interface permite a criação de um objeto que ainda não está relacionado ao objeto *COM*. Todo o objeto *COM* é baseado nesta interface.

A Figura 05 mostra um exemplo do funcionamento de um objeto *COM*. Neste exemplo é ilustrado a criação de um objeto *COM*:

Figura 05 – Funcionalidade do objeto COM.

O objeto *DCOM* (Modelos de Componentes Objetos Descentralizados, do inglês *Decentralized Component Object Model*), além de utilizar os mesmos recursos da *COM*, permite o acesso aos dados de forma remota, tornando possível à instalação de diferentes pacotes de softwares em vários aplicativos. (SHIMANUKI, 1999).

O funcionamento da tecnologia *DCOM* é ilustrado na Figura 06:

Figura 06 – Objeto DCOM.

4.4 FUNCIONALIDADE DO PADRÃO OPC

Segundo FONSECA (2002), o padrão *OPC* define regras para que sejam desenvolvidos sistemas com interfaces padrões de comunicação dos dispositivos de campo (Controlador Lógico Programável - CLP, Sensores) com sistemas de monitoração, supervisão e gerenciamento (*SCADA*, *ERP*). Para SITAO (2000a) quando se usa o *OPC*, é possível selecionar diversos dispositivos ou produtos de diversos fabricantes juntos no sistema e o desenvolvimento das aplicações pode ser facilitado, minimizando assim o tempo de desenvolvimento, gerando estabilidade no software, e gerando o aumento da confiabilidade e habilidade do software, conforme a Figura 07:

Figura 07 – Forma de acesso de uma aplicação Cliente OPC.

O OPC é baseado na arquitetura Cliente/Servidor, e é dividido em duas partes, o *OPC Client* e *OPC Server*. O *OPC Server*, ou o Servidor OPC contém as especificações de um determinado fabricante que o desenvolveu, por exemplo, quais as características do hardware utilizado, quais os dispositivos que pertencem ao fabricante, especificações sobre o fabricante, dentre outras. (FAYAD, 2000). Os fabricantes podem gerar diversos servidores, e através de códigos internos dos fabricantes, são determinados o meio e a forma de acesso aos dados dos dispositivos, por exemplo, o nome dos dados, os detalhes de como os dados serão acessados fisicamente, dentre outros.

O *OPC Client* ou, o Cliente OPC é a interface de comunicação de diversos aplicativos, por exemplo, do tipo *HMI* ou *SCADA*, que se comunicam com os servidores, isto é, os *OPCs Server*, seguindo as normas de comunicação do padrão OPC (JANKE, 2000).

O *OPC Server* é composto por outros objetos, os grupos, *OPC Groups* e os itens, *OPC Items*. O *OPC Groups* fornece um meio de acesso para os clientes organizarem os acessos aos dados, por exemplo, um grupo pode representar um conjunto de itens de uma determinada análise.

Existem dois tipos de grupos, os públicos e os locais. Os grupos públicos são criados para o acesso distribuído de vários clientes, e os grupos locais por sua vez, são acessados pelos clientes locais, ou seja, na máquina local.

Nos grupos o cliente pode definir um ou mais itens através do *OPC Items*. Esses itens serão adicionados em grupos e depois analisados ou manipulados. A Figura 08 representa a funcionalidade do *OPC Groups* e o *OPC Items*:

Figura 08 – Objeto *OPC Groups* e objeto *OPC Items*.

Os *OPC Items* representam conexões de bases de dados com o servidor *OPC*. Um *OPC Item* não pode ser acessado por um *OPC Client*, pois não existe o acesso externo ao *OPC Items*; o acesso é feito somente através do *OPC Groups*, onde o *OPC Item* está definido (PATTLE, 1997).

Para cada *OPC Item*, é associado um Valor, onde são definidos todos os tipos de dados *Variant* (Variante) suportados pela interface *DCOM*. A definição do valor de tempo é obtida através da estrutura padrão da plataforma *Windows*, a *UTC* (do inglês *Universal Time Coordinated*), podendo também ser aplicado ao tempo do dispositivo de campo *TS* (do inglês *Time Stamp*), ou por outra aplicação.

Os valores *Quality* são informações sobre o estado da conexão entre o *OPC Client* com cada item a ser analisado. Neste valor são reservados dois *bytes* para codificação do estado do dado fornecido pelo servidor, sendo que ainda só é utilizado um *byte* menos significativo. Dois *bits* definem a qualidade do dado; os resultados podem ser *Good*, onde ocorre a validação dos dados, e a comunicação não possui nenhum tipo de problema. Para resultados do tipo *Bad* acontecem quando ocorre perda do *link* com o servidor. Para resultados *Uncertain* não existe a comunicação, por exemplo, a incerteza se um determinado dispositivo esteja comunicando com o servidor, ou esta fora de operação.

Existem ainda quatro *bits* que fornecem os detalhes do estado apresentado como *Not Connected* e *Last Usable Value*, sendo que os dois últimos *bits* podem conter dados de diagnóstico, tais como a falha de um sensor (FONSECA, 2002).

A Figura 09, mostra a representação da funcionalidade do padrão *OPC* e diversas aplicações:

Figura 09 – Estrutura funcional do padrão OPC.

A leitura de dados pode ser feita de duas formas, síncrona na qual o grupo depende de uma confirmação para uma nova leitura, e assíncrona, quando ao contrário da leitura síncrona, o grupo não depende de confirmação. A partir da versão 2.0 da especificação para o servidor de dados, é possível fazer a seleção no *OPC Client* para a leitura dos dados da memória *cache* do servidor ou diretamente do dispositivo. Nos itens podemos definir se um item está ativo ou inativo, através do estado dos itens escolhidos pelo usuário; no estado ativo, o item está habilitado à comunicação, e no estado inativo, não ocorre à comunicação. A leitura cíclica ou por Mudança de Estado, ocorre quando o *OPC Client* define se os dados do servidor serão lidos de forma cíclica, onde o cliente realiza aquisições aos dados independentes se os dados sofreram mudanças ou não, ou por mudança de estado, onde o servidor é o responsável por indicar ao cliente quais os itens sofreram mudanças de seu estado. A Banda Morta é onde são definidos os limites de transição para os itens de um determinado grupo, para qual o servidor fará o envio para os clientes quando a alteração dos valores dos itens estiver fora da banda especificada (FONSECA, 2002).

Pelo fato do OPC ser baseado na tecnologia de comunicação entre aplicativos OLE/COM, e assumindo que todo objeto COM é acessado através de interfaces, o OPC

Client vê somente as interfaces, a *OPC Server* e a *OPC Groups*. Assim o *OPC Client* descreve uma representação lógica da conexão com o servidor.

A Figura 10 ilustra o modelo do objeto *OPC Client* e suas interfaces de comunicação:

Figura 10 – Interfaces do objeto *OPC Client*.

Essas interfaces são:

- *IOPCDataCallback*: Esta interface notifica se o *OPC Client* conseguiu a conexão com o *OPC Server*;
- *IOPCServerShutdown*: Notifica ao *OPC Client* que o *OPC Server* foi desconectado.

A Figura 11 ilustra o modelo do objeto *OPC Server* e suas interfaces de comunicação:

Figura 11 – Interfaces do objeto *OPC Server*.

Abaixo, é descrita a função de cada item da interface do *OPC Server*. Existem algumas interfaces do objeto *OPC Server*:

- *IUnknown*: Como todo objeto *COM*, o *Server* prove a interface padrão *IUnknown*, que é a interface padrão para objetos *COM*, determinado pela *Microsoft*;
- *IOPCCommon*: Nesta interface são os eventos distribuídos, como alarmes, utilizados no desenvolvimento de aplicativos;
- *IOPCServer*: Esta é a principal interface do objeto *OPC Server*, onde o servidor é registrado no sistema operacional. É essa interface deve conter todas as funcionalidades do Servidor;

- *IOPCServerPublicGroups* (opcional): Esta interface é responsável pelo gerenciamento dos grupos públicos, que são criados pelos clientes;
- *IOPCBrowseServerAddressSpace* (opcional): Esta interface fornece um método para visualização de todos os dados dos itens que estão no servidor, através listagem contendo a identificação dos itens;
- *IPersistFile* (opcional): Esta interface salva as configurações do servidor em alguma área do sistema, porém, não salva as configurações de clientes;
- *IOPCItemProperties*: Esta interface é acessada pelos clientes, é responsável pela manipulação dos itens contidos no servidor;
- *IConnectionPointContainer*: Esta interface provê o acesso de todos os pontos conectados no servidor.

A Figura 12 ilustra um modelo do objeto do tipo *OPC Groups* e suas interfaces de comunicação:

Figura 12 – Interfaces do objeto OPC Group.

Abaixo, é descrita a função de cada item da interface do *OPC Groups*:

- *IUnknown*: Como todo objeto *COM*, o *OPC Group* prove a interface padrão *IUnknown*, que é a interface padrão para objetos *COM*, determinado pela *Microsoft*;
- *IOPCItemMgt*: Esta interface permite um cliente remover, adicionar, e controlar o comportamento de itens em um grupo;
- *IOPCGroupStateMgt*: Esta interface permite gerenciar o estado do grupo.
- *IOPCPublicGroupStateMgt* (opcional): Quando um grupo é criado, ele automaticamente se torna um grupo apenas visualizado de forma privativa. Esta interface permite alterar o estado de um grupo, ou seja, se ele está em um estado privativo, para o estado público;

- *IOPCSyncIO*: Permite ao cliente para realizar escrita e leitura de dados de forma síncrona;
- *IOPCASyncIO2*: Permite ao cliente para realizar escrita e leitura de dados de forma assíncrona. A intenção da *OPC Foundation* é atualizar a interface *IOPCASyncIO*, para corrigir alguns problemas relacionados à escrita e leitura de dados;
- *IOPCASyncIO* (opcional, interface antiga): Permite ao cliente para realizar escrita e leitura de dados de forma assíncrona;
- *IDataObject* (opcional, interface antiga): Permite o acesso aos dados, permitindo a criação de uma conexão segura entre o cliente e um método eficiente de transferência de dados, conhecido como *OPC Data Stream Formats*, onde é estabelecido se o dado que está sendo manipulado, é seguro ou não;
- *IConnectionPointContainer* (opcional): Esta interface provê o acesso de todos os pontos conectados no servidor.

5 IDENTIFICAÇÃO DE SISTEMAS

Conceitualmente, a identificação de sistemas é um procedimento simples; com base nos sinais de entrada $u(k)$ e nos sinais de saída real de um sistema a ser identificado $y(k)$, um modelo matemático será obtido, e este será responsável por representar o comportamento do sistema a ser identificado, seja ele total ou de forma aproximada (AGUIRRE, 2000), conforme na Figura 13:

Figura 13 – Representação de um sistema dinâmico.

De uma forma geral, a Identificação de Sistemas inicia-se com a coleta dos dados experimentais, e suas respectivas excitações (ex. degrau, impulso). Após a aquisição de dados, determina-se uma estrutura apropriada para o modelo, por exemplo, uma equação polinomial de certa ordem, e então se estimam parâmetros correspondentes a tal modelo. O ajuste de parâmetros é feito através de algoritmos. A estimação, paramétrica e a do modelo, são feitas de forma iterativa, isto é, a estrutura é escolhida e os parâmetros são estimados. Após a escolha do modelo, este é validado, e caso não se chegue a um resultado esperado, são feitas novas estimações até que o modelo seja validado. Um exemplo de modelo é descrito na equação (1), onde u e y são os respectivos sinais de entrada e saída do sistema, a e b são os parâmetros, na e nb são as ordens dos polinômios, e k o número de amostras:

$$y(k) + a_1 y(k-1) + \dots + a_{na} y(k-na) = b_1 u(k-1) + \dots + b_{nb} u(k-nb) \quad (1)$$

5.1 ETAPAS DA IDENTIFICAÇÃO DE SISTEMAS

Na Identificação de Sistemas, a modelagem pode ser feita de duas formas, *off-line* e *on-line*. Na *off-line*, os sinais experimentais são processados e armazenados, e a partir desses dados armazenados, será feita a identificação. Dentre as técnicas de modelagem *off-line*, podemos destacar Resposta em Frequência (ex. Diagrama de Bode, Diagrama Polar), Resposta Temporal e Identificação Paramétrica. Já na *on-line*, os dados são processados, e a identificação é feita juntamente com o sistema em funcionamento. Dentre as técnicas de modelagem *on-line*, podemos destacar a Identificação Paramétrica via Método dos Mínimos Quadrados Recursivo, Variáveis Instrumentais, Redes Neurais Artificiais, dentre outras.

A seqüência de etapas da Identificação de Sistemas pode ser apresentada da seguinte maneira (AGUIRRE, 2000), (RODRIGUES, 2000):

- Testes dinâmicos e aquisição de dados;
- Aplicação de testes aos dados obtidos para detecção de não-linearidades;
- Escolha da representação da estrutura matemática;
- Determinação da estrutura de modelo;
- Estimção paramétrica;
- Validações do modelo identificado.

As etapas acima apresentadas são implementadas tanto para sistemas lineares quanto para sistemas não-lineares, sendo que as principais diferenças entre eles se devem à forma como cada procedimento é implementado. Este trabalho, no entanto tem aplicação somente em sistemas lineares. Aplicações com sistemas não-lineares podem ser encontrados em (RODRIGUES, 2000), (LJUNG, 1999), (AGUIRRE, 2000).

Na etapa de testes dinâmicos e aquisição de dados são obtidos os dados do modelo a ser identificado. Assim é de grande importância se definir bem quais são os passos necessários, tais como a aplicação e manipulação das entradas pré-determinadas utilizadas, o tempo de amostragem, a decisão quanto ao número de amostras adquiridas e o pré-processamento dos dados amostrados. Segundo RODRIGUES (2000), os sinais de entradas a serem utilizados para excitar o sistema devem ser projetados para satisfazer determinadas propriedades, ou seja, os sinais de entrada devem possuir um espectro de frequência adequado para excitar os diversos constituintes da planta, garantindo assim a adequabilidade dos dados obtidos.

Na etapa de aplicação de testes aos dados obtidos para detecção de não-linearidades, RODRIGUES (2000), afirma que todos os sistemas reais em última instância são não-lineares, pois a não-linearidade existe mesmo sendo de uma forma bem suave, o que torna

possível sua representação por de modelos lineares. Este autor utiliza técnicas para testar as não-linearidades de um sistema baseado no método da correlação cruzada não-linear, porém no presente trabalho não serão discutidas tais técnicas de detecção de não-linearidades, pois este lida somente de estruturas lineares. Tais técnicas podem ser esclarecidas em (HABER, 1985).

Na escolha da representação da estrutura matemática é feita a escolha de qual modelo matemático utilizar para a descrição do sistema. Tal escolha deve ser fundamentada no conhecimento do processo de identificação, assim como o conhecimento *à priori* sobre o sistema a ser identificado. De fato, os modelos matemáticos como foi visto anteriormente, são utilizados para descrever as propriedades de um sistema, sejam essas em partes ou de forma aproximada, o que explicaria a relação de causa e efeito sobre os dados experimentais (AGUIRRE, 2000), sendo essa etapa uma das mais importantes.

Um modelo linear polinomial pode ser representado pela seguinte estrutura:

$$y(k) = \sum_{i=1}^n \varphi_i^T(k) \theta_i + e(k) \quad (2)$$

onde:

- φ é definido como o vetor de regressores contendo os valores experimentais de entrada a saída do sistema a ser identificado, conforme (3);

$$\varphi = [y(k-1) \quad \dots \quad y(k-na) \quad u(k-1) \quad \dots \quad u(k-nb)] \quad (3)$$

onde $y(k)$ e $u(k)$ são respectivamente os sinais de saída(s) e entrada(s), na representa a quantidade de regressores de saída e nb a quantidade de regressores de entrada;

- θ é definido como o vetor com os parâmetros que devem ser estimados de acordo com a estrutura escolhida;
- $e(k)$ é definido como o erro de predição, definido pela diferença entre as respostas do sistema identificado com o sistema real.

No final da escolha da representação a ser utilizada são determinados, para sistemas lineares, a quantidade de pólos e zeros e o atraso puro do sistema. Já para sistemas não-lineares, deve-se verificar e determinar a quantidade de termos dos modelos polinomiais, uma vez que o número de termos em modelos polinomiais cresce de acordo com a não-linearidade.

Depois de determinada a estrutura, a próxima etapa é a escolha de qual método de identificação paramétrica utilizar. Nesta etapa são estimados os parâmetros da estrutura

matemática escolhida, estrutura esta responsável por representar o comportamento do sistema dinâmico a ser identificado. Na estrutura matemática apresentada em (2), ϕ define os regressores do modelo, que correspondem aos diferentes termos no polinômio, θ seus respectivos parâmetros e $e(k)$ o erro de predição, calculado pela diferença entre a resposta do sistema a ser identificado e o sistema real.

Seja o sistema *SISO* (*Single Input Single Output*) representado na Figura 13. O algoritmo dos Mínimos Quadrados é utilizado para calcular os parâmetros θ a cada amostra $y(k)$, tendo como base amostras obtidas quando o sistema estava em funcionamento, processadas em batelada em um segundo momento (*off line*). Podemos expressar a equação (2), pela equação diferença, conforme (4):

$$y(k) = a_1 y(k-1) + a_2 y(k-2) + \dots + a_{na} y(k-na) + b_1 u(k-1) + b_2 u(k-2) + \dots + b_{nb} u(k-nb) \quad (4)$$

onde a e b são os parâmetros do sistema a ser identificado, e $y(k)$ e $u(k)$ são os dados experimentais obtidos do sistema. Os termos na e nb definem a ordem da equação.

Escrevendo a equação (4) na forma matricial, obtém-se a equação (5), onde n representa o número mínimo de amostras necessárias para iniciar o processamento através das equações diferenças e N representa a grandeza total de amostras consideradas:

$$\begin{bmatrix} y_p(n) \\ y_p(n+1) \\ \vdots \\ y_p(N) \end{bmatrix} = \begin{bmatrix} y(n-1) & \dots & y(0) & u(n-1) & u(0) \\ y(n) & \dots & y(1) & u(n) & u(1) \\ \vdots & \dots & \vdots & \vdots & \vdots \\ y(N-1) & \dots & y(N-n) & u(N-1) & u(N-1) \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_{na} \\ b_1 \\ \vdots \\ b_{nb} \end{bmatrix} + \begin{bmatrix} e(n) \\ e(n+1) \\ \vdots \\ e(N) \end{bmatrix} \quad (5)$$

A partir da equação (5), define-se (6), (7), (8) e (9) como:

$$Y = \begin{bmatrix} y_p(n) \\ y_p(n+1) \\ \vdots \\ y_p(N) \end{bmatrix}_{N-n+1 \times 1} \quad (6)$$

$$\Psi \equiv \begin{bmatrix} y(n-1) & \cdots & y(0) & u(n-1) & \cdots & u(0) \\ y_p(n) & \cdots & y(1) & u(n) & \cdots & u(1) \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ y(N-1) & \cdots & y(N-n) & u(N-1) & \cdots & u(N-1) \end{bmatrix}_{(N-n+1)X(2n)} \quad (7)$$

$$\Theta \equiv \begin{bmatrix} a_1 \\ \vdots \\ a_{na} \\ b_1 \\ \vdots \\ b_{nb} \end{bmatrix}_{(2n)X(1)} \quad (8)$$

$$E \equiv \begin{bmatrix} e(n) \\ e(n+1) \\ \vdots \\ e(N) \end{bmatrix}_{(N-n+1)X(1)} \quad (9)$$

Escrevendo (5) de forma compacta, tem-se:

$$Y(k) = \Psi(k)\Theta + E(k) \quad (10)$$

onde as variáveis são definidas:

- Y – Vetor linha das medidas do sistema;
- Ψ – Matriz linha contendo os regressores, obtidos através das medidas de entrada e saída amostradas;
- Θ – Vetor coluna dos parâmetros estimados;
- E – Vetor coluna contendo o erro entre a resposta real e a resposta estimada;

Definindo-se o funcional J como erro quadrático médio, tem-se a equação (11):

$$J = \frac{1}{2} \sum_{k=n}^N e^2(k) \quad (11)$$

ou na forma matricial em (12):

$$J = \frac{1}{2} E^T E \quad (12)$$

Para se evitar a necessidade de se calcular os Mínimos Quadrados novamente a cada amostra adquirida, é necessário atualizar meramente a prévia da estimação calculada no tempo $\Theta(t)$, para obter uma nova estimação $\Theta(t+1)$. Portanto, o vetor Θ , definido em (8) será processado recursivamente para cada conjunto de n amostras, a partir de uma suposição inicial de um vetor de parâmetros, com pelo menos a quantidade mínima de parâmetros necessária para processar a primeira iteração, obtendo-se um novo vetor Θ para cada amostra. Deste modo, a resposta atual do algoritmo é uma previsão baseada em parâmetros processados pela planta, com o sistema em funcionamento, caracterizando o processamento *on-line*. Este método é conhecido como Mínimos Quadrados Recursivo (RLS) (LJUNG, 1999). Desta forma, obtêm-se a equação (13):

$$Y(k) = \Psi(k)\Theta(k-1) \quad (13)$$

Inicialmente, pode-se obter um valor alto para o erro, dependendo do conjunto de parâmetros iniciais escolhidos. Para solucionar tal problema, recomenda-se trabalhar com a ordem correta do sistema a ser identificado, assim, teoricamente, o erro tenderá a zero, realizando a convergência da saída aproximada para a saída real do sistema, conforme (14):

$$Y(k) = \Psi(k-1)\Theta(k-1) + E(k-1) \quad (14)$$

Aplicando os procedimentos de minimização do erro quadrático em (11) e (12) para o método dos Mínimos Quadrados, obtêm-se a seguinte equação (15):

$$\Theta^{N+1} = \Theta^N + K_{N+1}(Y^{N+1} - \Psi^{N+1}\Theta^N) \quad (15)$$

onde as variáveis são definidas como:

- Θ^{N+1} – Vetor coluna $(2n) \times (1)$ dos parâmetros estimados;
- Θ^N – Matriz coluna $(2n) \times (1)$ dos parâmetros estimados anteriores;
- K_{N+1} – Vetor $(2n) \times (1)$ de ganho de Kalman (LJUNG, 1999), contendo as ponderações a serem aplicadas no vetor Θ . O vetor de ganho de Kalman é definido em (16);

$$K_{N+1} = \frac{(\Psi^{NT}\Psi^N)^{-1}\Psi^{N+1T}}{\left[1 + \Psi^{N+1}(\Psi^{NT}\Psi^N)^{-1}\Psi^{N+1T}\right]} \quad (16)$$

- Y^{N+1} – Escalar com a medida atual da saída;
- Ψ^N – Matriz linha $(1) \times (2n)$ contendo as medidas de entrada e saída anteriores a Ψ^{N+1} ;
- E – Vetor coluna contendo o erro entre a saída real e a saída estimada;

Dentre os métodos de identificação paramétrica, podemos citar o Método dos Mínimos Quadrados Recursivo, Variáveis Instrumentais, Redes Neurais Artificiais (PAIVA, 1999), sendo esses aplicados de acordo com a necessidade de identificação de sistema.

Para se validar o modelo identificado é necessário verificar se este incorpora ou não as características de interesse do sistema original. AGUIRRE (2000) ressalta a importância de se comparar os modelos entre si e decidir se há algum candidato melhor, tornando-se assim uma etapa subjetiva e o resultado dependerá da aplicação pretendida para o modelo e da quantidade de informação disponível sobre o sistema original. A verificação e a validação do modelo é feita comparando as saídas do modelo com as respostas do sistema real para um conjunto de sinais de entrada (RODRIGUES, 2000), (SÖDERSTRÖM, 1991).

RODRIGUES (2000) cita duas maneiras de se validar um modelo: o Teste Cruzado e o Teste de Linearidade. No Teste Cruzado, são utilizadas duas séries diferentes de dados, e estimam-se parâmetros para o primeiro conjunto de dados. Assim a saída é calculada para o segundo conjunto de dados usando os parâmetros estimados do primeiro conjunto de dados.

A validação pelo Teste de Linearidade, os ensaios experimentais devem ser repetidos com outras amplitudes do sinal de entrada, introduzindo um sinal de excitação (ex. degrau ou impulso) de amplitude positiva e uma negativa, para verificar os limites do modelo identificado, para diferentes condições de operação.

O fluxograma, com a representação esquemática com as etapas do ciclo da técnica de Identificação de Sistemas, é apresentado na Figura 14 (LJUNG, 1999):

Figura 14 – Etapas da Ciclo da Identificação de Sistemas.

6 IDENTIFICAÇÃO DE SISTEMAS APLICADA EM ESTRUTURAS MATEMÁTICA DE VARIÁVEIS DE ESTADO

Em estratégias de controle adaptativo, pode-se representar um modelo através de variáveis de estado.

Existem técnicas para obter-se a representação por variáveis de estado a partir de sistemas discretos, obtidos pela Identificação de Sistemas.

Seja o sistema representado em (17), onde $u(k)$ e $y(k)$ são respectivamente as entradas e as saídas do sistema, no instante k .

$$\begin{aligned} y(k) + a_1 y(k-1) + a_2 y(k-2) + \dots + a_{na} y(k-na) \\ = b_0 u(k) + b_1 u(k-1) + \dots + b_{nb} u(k-nb) \end{aligned} \quad (17)$$

Aplicando a transformada Z (OGATA, 1994) na equação (17) obtêm-se a equação na forma de função de transferência conforme a equação (18):

$$\frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + \dots + b_{nb} z^{-nb}}{1 + a_1 z^{-1} + \dots + a_{na} z^{-na}} \quad (18)$$

ou

$$\frac{Y(z)}{U(z)} = \frac{b_0 z^{nb} + b_1 z^{nb-1} + \dots + b_{nb}}{z^{na} + a_1 z^{na-1} + \dots + a_{na}} \quad (19)$$

Existem muitas maneiras de se representar variáveis de estado por meio de sistemas discretos obtidos pela Identificação de Sistemas, conforme a equação (17). As formas canônicas são técnicas responsáveis por se obter representações por variáveis de estado, por meio de sistemas discretos. As formas canônicas podem ser obtidas pela aplicação de uma

Transformação Linear, conforme a equação (20), de forma que o sistema possa ser representado através da equação (21):

$$x_T(k)Tx(k) \quad (20)$$

$$\begin{cases} x_T(k+1) = G_T x_T(k) + H_T u(k) \\ y(k) = C_T x_T(k) + Du(k) \end{cases} \quad (21)$$

onde:

$$\begin{aligned} G_T &= TGT^{-1} \\ H_T &= TH \\ C_T &= CT \end{aligned} \quad (22)$$

Através dessas transformações, obtêm-se as seguintes Formas Canônicas (OGATA, 1994):

- *Forma Canônica Diagonal:*

$$\begin{array}{ccc} G_T & H_T & C_T \\ \begin{bmatrix} z_1 & 0 & 0 & \cdots & 0 \\ 0 & z_2 & 0 & \cdots & 0 \\ 0 & 0 & z_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & z_n \end{bmatrix} & \begin{bmatrix} b_{1D} \\ b_{2D} \\ b_{3D} \\ \vdots \\ b_{nD} \end{bmatrix} & \begin{bmatrix} C_{1D} \\ C_{2D} \\ C_{3D} \\ \vdots \\ C_{nD} \end{bmatrix} \end{array} \quad (23)$$

- *Forma Canônica de Controlabilidade:*

$$\begin{array}{ccc} G_T & H_T & C_T \\ \begin{bmatrix} 0 & 0 & \cdots & 0 & -a_n \\ 1 & 0 & \cdots & 0 & -a_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -a_2 \\ 0 & 0 & \cdots & 1 & -a_1 \end{bmatrix} & \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} C^T h \\ C^T Gh \\ \vdots \\ C^T G^{n-2} h \\ C^T G^{n-1} h \end{bmatrix} \end{array} \quad (24)$$

- *Forma Canônica Controlável:*

$$\begin{array}{ccc}
 G_T & H_T & C_T \\
 \begin{bmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ -a_n & -a_{n-1} & \cdots & -a_2 & -a_1 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} & \begin{bmatrix} b_n \\ b_{n-1} \\ \vdots \\ b_2 \\ b_1 \end{bmatrix}
 \end{array} \quad (25)$$

- *Forma Canônica de Observabilidade:*

$$\begin{array}{ccc}
 G_T & H_T & C_T \\
 \begin{bmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ -a_n & -a_{n-1} & \cdots & -a_2 & -a_1 \end{bmatrix} & \begin{bmatrix} C^T h \\ C^T G h \\ \vdots \\ C^T G^{n-2} h \\ C^T G^{n-1} h \end{bmatrix} & \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}
 \end{array} \quad (26)$$

- *Forma Canônica Observável:*

$$\begin{array}{ccc}
 G_T & H_T & C_T \\
 \begin{bmatrix} 0 & 0 & \cdots & 0 & -a_n \\ 1 & 0 & \cdots & 0 & -a_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -a_2 \\ 0 & 0 & \cdots & 0 & -a_1 \end{bmatrix} & \begin{bmatrix} b_n \\ b_{n-1} \\ \vdots \\ b_2 \\ b_1 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}
 \end{array} \quad (27)$$

7 TESTES E RESULTADOS

Para os testes deste trabalho foi desenvolvido o aplicativo do tipo cliente *OPC*, o Sintonizador. Suas características de desenvolvimento (plataforma, linguagem de programação) e suas características funcionais (módulos operacionais) encontram-se no apêndice A.

7.1 APLICAÇÃO DO SINTONIZADOR NO PROCESSO DE IDENTIFICAÇÃO DE SISTEMAS SIMULADOS

Nos testes iniciais com o Sintonizador, são utilizados sinais de um sistema simulado desenvolvido no aplicativo *Simulink*. Tal sistema é do tipo *SISO* de terceira ordem, e os dados de entrada e saída gerados ao longo da simulação, são registrados.

O diagrama de blocos do sistema simulado, está ilustrado na Figura 15, e a sua resposta a uma entrada degrau unitária, está ilustrada na Figura 16:

Figura 15 – Sistema Dinâmico Simulado.

Figura 16 – Resposta do Sistema Simulado a uma Entrada Degrau Unitária.

O dados registrados são processados no módulo de identificação de sistemas *Off-Line*, na opção *Off-Line*, dentro de Identificação, na opção Dados, do Menu Principal. O módulo de Identificação está configurado com 3 regressores de entrada e 3 regressores de saída, com taxa de amostragem igual a 0,1 segundos.

Assim, os parâmetros identificados pelo Sintonizador, para o sistema simulado descrito, estão na equação (28), bem como a função de transferência ilustrada na equação (29), e o modelo *ARX* na equação (30):

$$\theta = [0,6547 \quad 0,1625 \quad -0,2937 \quad 0,1213 \quad 0,2259 \quad 0,1293] \quad (28)$$

$$\frac{Y}{U}(z) = \frac{0,1213z^2 + 0,2259z + 0,1293}{z^3 - 0,6547z^2 - 0,1625z + 0,2937} \quad (29)$$

$$y(k) = 0,6547y(k-1) + 0,1625y(k-2) - 0,2937y(k-3) + 0,1213u(k-1) + 0,2259u(k-2) + 0,1293u(k-3) \quad (30)$$

Para a validação do modelo identificado, foi feita a comparação entre a resposta gerada pelo modelo identificado com a resposta do sistema simulador, para o mesmo sinal de entrada, o Teste Cruzado. A resposta do sistema simulado é ilustrado na Figura 16 e a resposta obtida pelo modelo identificado, é ilustrada na Figura 17. O erro quadrático médio, conforme (11) é ilustrado na Figura 18:

Figura 17 – Resposta do Sistema Identificado.

Figura 18 – Erro Quadrático Médio de Resposta do Sistema Identificado.

7.2 APLICAÇÃO DO SINTONIZADOR NO PROCESSO DE IDENTIFICAÇÃO DE SISTEMAS EM AMBIENTE FIELDBUS

Nos testes de campo foram utilizados dois sistemas, um sistema dinâmico térmico e um túnel de aquecimento.

O sistema térmico dinâmico é composto por um forno elétrico, instrumentado com transmissores *Foundation Fieldbus*. Para tal, foram utilizados um transmissor de temperatura, e um conversor *fieldbus* – 4 a 20ma. O esquema do sistema está representado na Figura 19:

Figura 19 – Representação do Forno Elétrico (BRANDÃO, 2000).

O túnel de aquecimento é constituído por um tubo de PVC, de 2 metros de comprimento e 53,5 mm de diâmetro. No seu interior, é composto por um motor 12 Vdc, que é responsável por gerar o fluxo de ar dentro do tubo, por duas resistências elétricas e dois termopares do tipo K. Tal sistema é instrumentado por dois transmissores de temperatura e dois conversores *fieldbus* – 4 a 20ma. A Figura 20 mostra o esquema do túnel de aquecimento:

Figura 20 – Representação do Túnel de Aquecimento.

Nos dois sistemas, o transmissor de temperatura está ligado a um termopar do tipo K, responsável por medir a temperatura interna do forno e o conversor de *fieldbus* para 4 a 20ma aciona uma chave estática, que alimenta a resistência elétrica que fornece potência ao forno. Os transmissores estão dispostos em um canal *fieldbus*, ligado a uma placa de interface PCI302. Para a configuração dos sistemas, foi utilizado o software Syscon da Smar.

Nos experimentos, o Sintonizador estabelece uma conexão remota com o servidor *OPC* que está conectado a rede *fieldbus* em outra máquina. A Figura 21, mostra a tela de conexão do Sintonizador com servidor *OPC*:

Figura 21 – Conexão com Servidor *OPC* da Rede *Fieldbus*.

Após a conexão, o Sintonizador automaticamente cria um grupo lógico, responsável por organizar os itens de interesse nas aquisições e nas análises. Por fim, o Sintonizador gera

uma lista contendo os itens disponíveis do servidor *OPC*, onde estes serão adicionados no grupo referente a sua seleção.

Para os testes de identificação com o Sintonizador em ambientes *fieldbus*, são selecionadas as variáveis experimentais de entrada e saída do sistema, onde essas são registradas, juntamente com as respectivas excitações. A Figura 22 mostra a tela do Sintonizador realizando as aquisições dos sistemas descritos:

Figura 22 – Aquisição de Dados Experimentais na Rede *Fieldbus*.

Para o forno elétrico, a potência em regime de 20%, igual a 7,2 mA a uma temperatura de 52,15 °C, foi excitada com o aumento de potência de 20% para 50%, de 7,2 mA para 12 mA, estabilizando em 128,2 °C. A duração do ensaio foi de 11,1 minutos com uma taxa de amostragem de 0,1 segundos. A resposta do forno é ilustrada na Figura 23:

Figura 23 – Resposta do Forno Elétrico a uma Entrada Degrau Unitária.

O dados registrados são processados no módulo de identificação de sistemas *Off-Line*, na opção *Off-Line*, dentro de Identificação, na opção Dados, do Menu Principal. O módulo de Identificação está configurado com 3 regressores de entrada e 3 regressores de saída.

Os parâmetros calculados pelo Sintonizador a partir dos dados experimentais do túnel de aquecimento estão na equação (31):

$$\theta = [0,9555 \quad 5,69E-5 \quad 0,0437 \quad 8,475E-6 \quad -1,7871E-9 \quad 0,0006704] \quad (31)$$

A partir de (31), obtêm-se a função de transferência em (32) e o modelo *ARX* em (33):

$$\frac{Y}{U}(z) = \frac{8,475E-6z^2 - 1,7871E-9z + 0,0006704}{z^3 - 0,9555z^2 - 5,69E-5z - 0,0437} \quad (32)$$

$$y(k) = 0,955y(k-1) + 5,69E-5y(k-2) + 0,0437y(k-3) + 8,475E-6u(k-1) - 1,7871E-9u(k-2) + 0,0006704u(k-3) \quad (33)$$

A Figura 24 ilustra a validação feita comparando a resposta gerada pelo modelo identificado com a resposta do forno elétrico, para o mesmo sinal de entrada:

Figura 24 – Resposta do Sistema Identificado com dados Experimentais do Forno Elétrico.

O erro quadrático médio, conforme (11), calculado entre a resposta do modelo identificado e a resposta do forno elétrico, é ilustrado na Figura 25:

Figura 25 – Erro Quadrático Médio da Resposta do Sistema Identificado com a Resposta do Forno Elétrico.

Nos ensaios com o túnel de aquecimento mediu-se a temperatura no ponto C, onde a potência da resistência em regime de 25% ou 8 mA e a temperatura 43 °C. Excitando-se a resistência da potência de 25% para 75%, de 8 mA para 16 mA, com duração de 7 minutos. A Figura 26 ilustra a resposta do túnel de aquecimento à uma entrada degrau:

Figura 26 – Resposta do Túnel de Aquecimento a uma Entrada Degrau Unitária.

Os parâmetros calculados pelo Sintonizador a partir dos dados experimentais do túnel de aquecimento estão na equação (34), também sua função de transferência em (35) e a equação diferença descrita pelo modelo ARX em (36):

$$\theta = [0,9664 \quad 1,735E-5 \quad 0,03146 \quad 0,00077 \quad -7,5960E-5 \quad 0,00150] \quad (34)$$

$$\frac{Y}{U}(z) = \frac{0,0007z^2 - 7,596E-5z + 0,00150}{z^3 - 0,9664z^2 - 1,735E-5z - 0,03146} \quad (35)$$

$$y(k) = 0,9664y(k-1) + 1,735E-5y(k-2) + 0,03146y(k-3) + 0,00077u(k-1) - 7,5960E-5u(k-2) + 0,00150u(k-3) \quad (36)$$

Para validar o modelo identificado, compara-se a resposta gerada pelo modelo identificado com a resposta do túnel de aquecimento, para o mesmo sinal de entrada. Tal comparação é ilustrada na Figura 27:

Figura 27 – Resposta do Sistema Identificado a partir de Dados Experimentais do Túnel de Aquecimento.

Na Figura 28 é ilustrado o erro quadrático médio, conforme (11), calculado com a resposta obtida pelo modelo identificado e a resposta do túnel de aquecimento:

Figura 28 – Erro Quadrático Médio da Resposta do Sistema Identificado com a Resposta do Túnel de Aquecimento.

Observa-se que o erro quadrático inicialmente obtido em todos os ensaios apresenta um valor inicial relativamente alto. Entretanto, trabalhando com a ordem correta do sistema, teoricamente esse erro tenderá a decrescer a um valor ínfimo com a evolução do número de iterações, realizando a convergência entre a resposta simulada para a resposta real do sistema.

8 CONCLUSÕES

Primeiramente conclui-se a facilidade no desenvolvimento de aplicativos do tipo cliente OPC, viabilizando a construção de ferramentas abertas para o desenvolvimento de novas estratégias de controle adaptativo, resultado este obtido na construção do aplicativo Sintonizador.

Como visto anteriormente, uma das grandes vantagens da técnica experimental de Identificação de Sistemas é a facilidade na implementação dos algoritmos que a compõe. Como estudo de caso real, este trabalho apresentou a base teórica, a concepção, a implementação do aplicativo computacional, o Sintonizador, que realizou a identificação paramétrica de sistemas dinâmicos de até terceira ordem em tempo real e (ou) por meio de dados coletados com o sistema quando estava em funcionamento, pelo método dos Mínimos Quadrados Recursivo (RLS) aplicado a estrutura matemática de sistemas lineares ARX, responsável pela predição do sistema identificado em tempo real.

Este trabalho apresentou e exemplificou uma alternativa para o usuário de um sistema de instrumentação *Foundation Fieldbus* viabilizar o projeto de novas estratégias de controle.

A finalidade da Identificação de Sistemas dinâmicos em tempo real é o estudo e o desenvolvimento de estratégias de controle adaptativas aplicadas ao ambiente *Foundation Fieldbus*. O procedimento de validação Teste Cruzado, mostrou que a convergência do algoritmo de estimação paramétrica proposta é imune aos ruídos de medição da planta e que a velocidade de convergência do modelo estimado em relação à planta real apresenta um comportamento logarítmico frente à velocidade desta última. Este resultado viabiliza o uso do aplicativo proposto em estratégia adaptativas de controle para sistemas dinâmicos lentos sem a perda da precisão da estimação paramétrica em regime permanente de operação da planta.

8.1 SUGESTÕES PARA TRABALHOS FUTUROS

Devido à compatibilidade entre *fieldbus* e o controle moderno (BRANDÃO, 2000), bem como a compatibilidade entre *fieldbus* e a técnica experimental de Identificação de Sistemas, pode se fazer sugestões para trabalho futuros nesta linha:

- Ampliar o uso do Sintonizador para outros tipos de sistemas, como por exemplo, sistemas não-lineares, sistemas do tipo *MIMO*;
- Aperfeiçoar o Sintonizador em outras estratégias de controle adaptativo, estratégias de controle avançado (Redes Neurais, Lógica *Fuzzy*);
- Interoperabilidade entre o Sintonizador e outros protocolos *fieldbus*;
- Desenvolvimento de interface WEB aplicada a estratégias de controle adaptativo;
- Desenvolvimento de Blocos Funcionais aplicados a estratégias de controle adaptativo.

APÊNDICE A – O APLICATIVO SINTONIZADOR

O software Sintonizador apresentado neste trabalho é uma ferramenta desenvolvida na plataforma de programação da Borland® o Delphi Vs 6.0, para sistemas operacionais Windows. O Sintonizador é uma aplicação do tipo cliente OPC, e foi desenvolvida seguindo as normas para o desenvolvimento de aplicações cliente OPC da OPC Foundation (OPC FOUNDATION, 2002).

Dentre algumas características importantes do Sintonizador é de ser uma ferramenta aberta de comunicação com diversas bases de dados do tipo OPC, e também como ferramenta de apoio para o desenvolvimento e configuração de estratégias de controle. A Figura 29, contem a tela inicial do software Sintonizador na versão 1.0:

Figura 29 – Tela inicial do aplicativo Sintonizador.

O Sintonizador é composto por quatro módulos operacionais responsáveis pela comunicação e aquisição de dados de servidores OPC, responsáveis pela aplicação do Método Experimental de Identificação de Sistemas e finalmente um módulo responsável pelas configurações necessárias para o funcionamento do aplicativo.

O módulo *Server* é responsável pela conexão entre o aplicativo Sintonizador com os diversos servidores *OPC*. Nesta opção, especifica-se o nome do servidor e, caso o servidor *OPC* seja remoto, isto é, o servidor está em um ponto diferente em uma rede, é especificado o nome do computador onde o servidor se encontra. Esse módulo é acessado através da opção *Server* do menu principal, ilustrado na Figura 30:

Figura 30 – Tela de conexão com servidores *OPC*.

Caso a conexão seja bem sucedida, o aplicativo informa que o servidor está conectado, caso contrário, o aplicativo retorna que a conexão não foi executada.

O módulo de Dados está dividido em dois outros módulos, o de Identificação de Sistemas, e o de Aquisição de Dados. Esse módulo é acessado pela opção Dados do menu principal.

Todas as rotinas necessárias para a implementação do Método Experimental de Identificação de Sistemas estão implementadas no módulo de Identificação e acessadas pela opção Identificação do menu Dados.

Nesta opção, existem duas formas de implementação da Identificação de Sistemas, a *Off-line* e a *On-Line*. A primeira forma, a identificação é feita com base nos dados armazenados em arquivos do tipo texto, onde estes são processados e um conjunto de parâmetros utilizados na configuração das matrizes dinâmicas do sistema é encontrado.

A validação é feita graficamente, calculando as respostas, a simulada e a preditiva conforme AGUIRRE (2000). A Figura 31 ilustra a tela com a opção de identificação de sistemas *Off-Line*:

Figura 31 – Tela de Identificação de Sistemas *Off-Line*.

Já na segunda forma, a identificação *on-line*, o processo é o mesmo da identificação *off-line*, porém os dados utilizados para a identificação são obtidos com o sistema em funcionamento, ou seja, os dados são obtidos em tempo real. Em adicional, são definidas quais as variáveis que serão armazenadas e as variáveis que serão utilizadas para o processo de identificação. A Figura 32 contém a tela com a identificação de sistemas *On-Line* em funcionamento:

Figura 32 – Tela de Identificação de Sistemas *On-Line*.

A validação do modelo é feita de maneira idêntica ao módulo de identificação *Off-Line*.

Na opção Aquisição do menu principal, são realizadas aquisições de dados do servidor *OPC*, por meio do módulo de Aquisição de Dados. Neste módulo, todos os itens disponíveis para aquisição e manipulação do servidor, são dispostos em uma lista, e selecionados de acordo com o interesse. A Figura 33 mostra a tela da opção de Aquisição de Dados:

Figura 33 – Tela do Módulo de Aquisição de Dados.

Neste módulo é possível visualizar os dados graficamente, selecionando a opção de Gráfico, na tela principal. Através da opção Dados do menu principal, são especificados, o diretório e o nome do arquivo onde os dados serão armazenados e a taxa de amostragem. A Figura 34 contém a tela, onde são feitas as especificações para a aquisição de dados:

Figura 34 – Tela com as Especificações para Aquisição de Dados.

Na opção Configurações, são configuradas as variáveis necessárias para o funcionamento do Sintonizador. A Figura 35 mostra a tela da opção de Configurações do menu principal:

Figura 35 – Tela de Configurações.

Na opção Ordem do Sistema, é configurada a ordem do sistema a ser utilizado. Na opção de Matrizes, são configuradas as matrizes de estado, por meio do módulo de Matrizes, onde estas são utilizadas pelo bloco funcional Estimador de Estados.

Tais matrizes podem ser alimentadas manualmente ou por meio de processos de Identificação de Sistemas contidos no Sintonizador, e atualizadas no servidor *OPC*. Neste

módulo são calculadas as matrizes de ganho de realimentação de estados, com base nos requisitos de desempenho do controlador em malha fechada.

A tela com configuração das matrizes dinâmicas é ilustrada na Figura 36:

Matriz A	Matriz B	Matriz C	Matriz D	Matriz K	Matriz L
0	0	0.04366950850635			
1	0	5.680109148E-5			
0	1	0.95500974235192			

Ordem do sistema = 3

Download

OPC

Figura 36 – Tela de configuração das matrizes do bloco Estimador de Estados.

Na opção Requisitos de Desempenho do Controlador em Malha Fechada, é determinado o comportamento do sistema em malha fechada, com base em requisitos de desempenho do controlador em malha fechada. A Figura 37 ilustra a tela da configuração dos requisitos de desempenho do sistema em malha fechada:

Requisitos de Desempenho do Sistema em Malha Fechada

0.4 Tempo de Subida (tr) 0.3 Tempo de Atraso (td) 10 Máximo Sobre-Sinal (Mp) %

0.6 Instante de Pico (tp) 0.5 Frequência de Amortecimento (xi) 1 Tempo de Acomodação (ts) 2% 5%

5.5 Frequência Natural não Amortecida (wn) 5.2 Frequência Natural Amortecida (wd)

Gravar Resposta

Calcular Variáveis

Função Transferência em Malha Fechada - ITAE 3ª Ordem

$$\frac{Y(s)}{U(s)} = \frac{214,626}{s^3 + 11,3175s^2 + 60,63175s + 214,626}$$

Figura 37 – Tela da Configuração dos Requisitos de Desempenho do Sistema em Malha Fechada.

Depois de especificados os parâmetros de desempenho, o Sintonizador calcula a função de transferência em malha fechada e o gráfico referente à resposta do sistema em malha fechada é gerado, conforme a Figura 38:

Figura 38 – Resposta do Sistema em Malha Fechada.

Na opção de Diretório de Aquisições, especifica-se o diretório onde todas as aquisições são armazenadas. Na opção de *Regressores (Entrada/Saída)*, são especificadas as ordens dos regressores de entrada e de saída utilizados nos algoritmos de Identificação de Sistemas.

ANEXO A – FUNDAMENTOS TEÓRICOS

1 – Modelos Matemáticos

Os modelos matemáticos são utilizados para descrever o comportamento de um determinado processo, e são classificados como:

- *Modelos Estáticos e Modelos Dinâmicos*: denominam-se como modelos estáticos os que visam representar o estado de um sistema em um instante ou que em suas formulações não se leva em conta a variável de tempo, já os modelos dinâmicos descrevem como o efeito de um sinal de entrada pode influenciar no sinal de saída, durante um certo período;
- *Modelos de Entrada Simples e Saída Simples (SISO) e modelos multivariáveis (MIMO)*: os modelos *SISO* a descrição do modelo é dada pela relação de uma variável de entrada e uma variável de saída, enquanto para modelos *MIMO* existem mais variáveis envolvidas;
- *Modelos Lineares e Modelos Não-Lineares*: os sistemas Lineares seguem a lei da propriedade aditiva ou da superposição (SANTOS, 2000), onde um sinal de saída de um determinado sistema é proporcional à entrada, ao contrário de sistemas não-lineares;
- *Modelos Paramétricos e Modelos Não-Paramétricos*: os modelos paramétricos são descritos por parâmetros, ao contrário dos modelos não-paramétricos que são descritos por funções ou gráficos;
- *Modelos Determinísticos e Modelos Estocásticos*: os modelos determinísticos o valor de saída é obtido de maneira exata, assim que o sinal de entrada é conhecido. Modelos estocásticos possuem termos aleatórios, responsáveis pela dificuldade de se obter o valor de saída (AGUIRRE, 2000);
- *Modelos Dominantes no Tempo e na Frequência*: os modelos dominantes no tempo são aqueles formados por equações algébricas, diferenciais, equações diferenças. Modelos dominantes na frequência são aqueles que resultam numa representação gráfica, tais como densidade espectral, diagrama de Bode, dentre outros.

- *Modelos Discretos e Modelos Contínuos* - os modelos discretos são aqueles em que o avanço da contagem de tempo se dá na forma de incrementos cujos valores podem ser definidos em função da ocorrência dos eventos ou pela determinação de um valor fixo, nesses casos só é possível determinar os valores das variáveis de estado do sistema nos instantes de atualização da contagem de tempo; para modelos contínuos, o avanço da contagem de tempo na simulação dá-se de forma contínua, o que possibilita determinar os valores das variáveis de estado a qualquer instante.

2 – Estruturas Discretas de Modelos Lineares Polinomiais Utilizadas

Dentre alguns tipos de modelos dinâmicos lineares utilizados no método experimental de Identificação de Sistemas, temos:

- Estruturas matemáticas *ARMAX*: para modelos *SISO*, tem-se a estrutura geral da estrutura matemática *ARMAX* de ordem N , conforme (37):

$$y(k) = a_1 y(k-1) + \dots + a_{na} y(k-na) + b_1 u(k-1) + \dots + b_{nb} u(k-nb) + e(k-1) + c_1 e(k-1) + \dots + c_{nc} e(k-nc) \quad (37)$$

onde:

$y(k)$ – Sinal de Saída;

$u(k)$ – Sinal de Entrada;

a , b e c – Parâmetros Estimados;

$e(k)$ – Erro, ou ruído branco;

Para modelos *MIMO*, obtém-se a seguinte estrutura *ARMAX* de ordem n , conforme equação (38):

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \end{pmatrix} (k) = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1j} \\ a_{21} & a_{22} & \dots & a_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \dots & a_{ij} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \end{pmatrix} (k-1) + \dots + \begin{pmatrix} a_{N11} & a_{N12} & \dots & a_{N1j} \\ a_{N21} & a_{N22} & \dots & a_{N2j} \\ \vdots & \vdots & \ddots & \vdots \\ a_{Ni1} & a_{Ni2} & \dots & a_{Nij} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \end{pmatrix} (k-N) + \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1j} \\ b_{21} & b_{22} & \dots & b_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ b_{i1} & b_{i2} & \dots & b_{ij} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_i \end{pmatrix} (k-1) + \dots + \begin{pmatrix} b_{N11} & b_{N12} & \dots & b_{N1j} \\ b_{N21} & b_{N22} & \dots & b_{N2j} \\ \vdots & \vdots & \ddots & \vdots \\ b_{Ni1} & b_{Ni2} & \dots & b_{Nij} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_i \end{pmatrix} (k-N) + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_i \end{pmatrix} (k) + \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1j} \\ c_{21} & c_{22} & \dots & c_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ c_{i1} & c_{i2} & \dots & c_{ij} \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_i \end{pmatrix} (k-1) + \dots + \begin{pmatrix} c_{N11} & c_{N12} & \dots & c_{N1j} \\ c_{N21} & c_{N22} & \dots & c_{N2j} \\ \vdots & \vdots & \ddots & \vdots \\ c_{Ni1} & c_{Ni2} & \dots & c_{Nij} \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_i \end{pmatrix} (k-N) \quad (38)$$

- Estruturas matemáticas *ARX*: para modelos *SISO*, tem-se a estrutura geral da estrutura matemática *ARX* de ordem n , em (39):

$$y(k) = a_1 y(k-1) + \dots + a_{na} y(k-na) + b_1 u(k-1) + \dots + b_{nb} u(k-nb) + e(k-1) \quad (39)$$

onde:

$y(k)$ – Sinal de Saída;

$u(k)$ – Sinal de Entrada;

a e b – Parâmetros Estimados;

$e(k)$ – Erro, ou ruído branco;

Para modelos *MIMO*, obtém-se a seguinte estrutura matemática *ARX* de ordem N , em

(40):

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \end{pmatrix} (k) = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1j} \\ a_{21} & a_{22} & \dots & a_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \dots & a_{ij} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \end{pmatrix} (k-1) + \dots + \begin{pmatrix} a_{N11} & a_{N12} & \dots & a_{N1j} \\ a_{N21} & a_{N22} & \dots & a_{N2j} \\ \vdots & \vdots & \ddots & \vdots \\ a_{Ni1} & a_{Ni2} & \dots & a_{Nij} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \end{pmatrix} (k-N) + \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1j} \\ b_{21} & b_{22} & \dots & b_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ b_{i1} & b_{i2} & \dots & b_{ij} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_i \end{pmatrix} (k-1) + \dots + \begin{pmatrix} b_{N11} & b_{N12} & \dots & b_{N1j} \\ b_{N21} & b_{N22} & \dots & b_{N2j} \\ \vdots & \vdots & \ddots & \vdots \\ b_{Ni1} & b_{Ni2} & \dots & b_{Nij} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_i \end{pmatrix} (k-N) + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_i \end{pmatrix} (k) \quad (40)$$

ANEXO B – TEORIA DE CONTROLE

1 – Representação em Espaço de Estados

A representação em espaço de estados é baseada no cálculo de um sistema de n equações diferenciais de 1ª ordem, composta por duas equações matriciais que relacionam as entradas do sistema aos estados e os estados com as saídas do sistema.

Para tal teoria, faz-se necessário o conhecimento de alguns conceitos básicos:

- *Estado*: o estado de um sistema dinâmico representa o menor conjunto de variáveis, da onde é possível determinar o completo comportamento do sistema com base nos valores no instante $t = t_0$ e $t > t_0$;
- *Variáveis de Estado*: variáveis de estado representam o conjunto de estados, responsáveis por definir o sistema dinâmico. Se para n variáveis, $x_1, x_2, x_3, \dots, x_n$, são necessárias para determinar por completo o comportamento do sistema, então as n variáveis são denominadas Variáveis de Estado;
- *Vetor de Estados*: se n variáveis são necessárias para descrever por completo o comportamento de um determinado sistema, então estas n variáveis podem ser consideradas n variáveis de um vetor x , denominado *Vetor de Estados*.
- *Espaço de Estados*: o espaço n -dimensional onde estão as coordenadas $x_1, x_2, x_3, \dots, x_n$, é denominada como *Espaço de Estados*. Um estado é representado como um ponto no espaço de estado;
- *Equações por Espaço de Estados*: Na análise por variáveis de estado, existem três tipos de variáveis que estão envolvidas na modelagem do sistema dinâmico: variáveis de entrada, variáveis de saída e variáveis de estado. Para sistemas discretos variantes no tempo, a equação de estados pode ser representada em (41), a equação de saída está definida em (42), e finalmente para sistemas discretos lineares e variantes no tempo, a equação de estados são definidas em (43):

$$x(k+1) = f[x(k), u(k), k] \quad (41)$$

$$y(k) = g[x(k), u(k), k] \quad (42)$$

$$\begin{aligned}x(k+1) &= G(k)x(k) + H(k)u(k) \\ y(k) &= C(k)x(k) + D(k)u(k)\end{aligned}\quad (43)$$

onde:

$x(k)$ = vetor n -dimensional – vetor de estado;

$y(k)$ = vetor m -dimensional – vetor de saída;

$u(k)$ = vetor r -dimensional – vetor de entrada;

$G(k)$ = matriz $n \times n$ – matriz de estado;

$H(k)$ = matriz $n \times r$ – matriz de entrada;

$C(k)$ = matriz $m \times n$ – matriz de saída;

$D(k)$ = matriz $m \times r$ – matriz de transmissão direta;

O surgimento da variável k nas matrizes $G(k)$, $H(k)$, $C(k)$ e $D(k)$, significa que estas matrizes são variantes no tempo, caso contrário, as matrizes podem ser invariantes no tempo, ou constantes. Assim, se o sistema é invariante no tempo, então a equação (43) pode ser reescrita como:

$$\begin{aligned}x(k+1) &= Gx(k) + Hu(k) \\ y(k) &= Cx(k) + Du(k)\end{aligned}\quad (44)$$

Para sistemas contínuos, a equação (43), pode ser representada em (45), e reescrevendo a equação (45), para sistemas contínuos e variantes no tempo, obtêm-se (46). Para sistemas invariantes no tempo, obtêm-se (47):

$$\begin{aligned}\dot{x}(t) &= f[x(t), u(t), t] \\ y(t) &= g[x(t), u(t), t]\end{aligned}\quad (45)$$

$$\begin{aligned}\dot{x}(t) &= A(t)x(t) + B(t)u(t) \\ y(t) &= C(t)x(t) + D(t)u(t)\end{aligned}\quad (46)$$

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned}\quad (47)$$

A Figura 39, ilustra a o diagrama de blocos para sistemas de controle discretos definidos pela equação (43).

Figura 39 – Diagrama de blocos para sistemas de controle discreto via Espaço de Estado.

A Figura 40, ilustra o diagrama de blocos para sistemas de controle contínuos, conforme a equação (47):

Figura 40 – Diagrama de blocos para sistemas de controle contínuos via Espaço de Estados.

2 – Realimentação e Estimação De Estados

Da equação (43), define-se 4 matrizes, a matriz dinâmica G , a matriz de controle H , a matriz de saída C e a matriz de transmissão direta D , definidas em (48):

$$\begin{aligned}
 G_{n \times n} &= \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} & H_{n \times r} &= \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & b_{22} & \cdots & b_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nr} \end{bmatrix} \\
 C_{m \times n} &= \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{bmatrix} & D_{m \times r} &= \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1r} \\ d_{21} & d_{22} & \cdots & d_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mr} \end{bmatrix}
 \end{aligned} \tag{48}$$

As matrizes acima são para sistemas do tipo *MIMO*, porém para sistemas do tipo *SISO*, a matriz H é do tipo coluna e a matriz C é do tipo linha. Na maioria dos sistemas, a matriz de transmissão direta D é nula, e a equação de saída pode ser reescrita em (49)

$$y(k) = Cx(k) \quad (49)$$

As características dinâmicas em um sistema contínuo, são determinadas pela localização dos pólos e zeros no plano $\delta - j\omega$. Em sistemas discretos, isso é feito através da localização dos pólos através do mapeamento $z = e^{sT}$, onde $s = \delta - j\omega$ (OGATA, 1994). Os pólos do sistema λ , são determinados através dos autovalores da matriz G , dado pela equação (50), onde I é a matriz identidade de dimensão n . O resultado de (50) é igual ao conjunto de autovalores de G .

$$Q(\lambda) = |\lambda I - G| = 0 \quad (50)$$

Assim, o comportamento do sistema pode ser mudado, alterando a localização dos pólos, através de um sistema controlador. Esse controlador tem a função de descrever para o sistema dinâmico o comportamento dinâmico de projeto ideal, alterando assim a dinâmica real do sistema. Uma das maneiras de se regular um sistema, é através da técnica de alocação de pólos, conhecida como Realimentação de Estados. Esta técnica consiste em realizar aquisições das variáveis, ou os estados, e multiplica-los por ganhos específicos e adicioná-los na entrada (OGATA, 1994). A Figura 41 ilustra o funcionamento da técnica de realimentação de estados:

Figura 41 – Realimentação de Estados.

Adotando a matriz de ganho K de dimensão $n \times k$, o vetor de entradas u , conforme (51), e reescrevendo a equação (43), obtemos a equação do sistema em malha fechada, em (52) (OGATA, 1994):

$$u = r + Kx \quad (51)$$

$$x(k+1) = (G - HK)x(k) \quad (52)$$

Obtemos então a nova G_f matriz dinâmica em malha fechada em (53), assim são obtidos novos autovalores do sistema, em função dos elementos da matriz K . O projeto de alocação de pólos consiste em calcular os elementos da matriz K , onde a matriz K é do tipo linha:

$$G_f = (G - HK)x(k) \quad (53)$$

Uma condição inicial para o cálculo da matriz K , é verificar se o sistema é totalmente controlável. Um sistema é dito controlável se existir uma ação de controle capaz de levar o sistema do seu estado inicial $x(kT)$ para um estado final $x(kT + \Delta t)$. Por definição, o sistema é completamente controlável, se através de uma ação de controle, ou uma excitação, todos seus estados se modifiquem (OGATA, 1994).

Numericamente, pode-se definir a controlabilidade de um sistema através da matriz de controlabilidade, definida na equação (54):

$$M_c = \begin{bmatrix} H & GH & G^2H & \dots & G^{n-1}H \end{bmatrix} \quad (54)$$

onde o posto da matriz de controlabilidade deve ser igual ao número de estados n , do sistema a ser controlado, para que exista a controlabilidade. Assim, é definida a condição de controlabilidade pela equação (55):

$$\text{posto}(M_c) = \text{posto}(H \quad GH \quad G^2H \quad \dots \quad G^{n-1}H) = n \quad (55)$$

Porém, um inconveniente da técnica de alocação de pólos por realimentação de estados é que a técnica assume que todos os estados estão disponíveis para aquisição, sendo fisicamente ou economicamente inviável a sua medição para alguns sistemas de controle.

Para solucionar tal problema, criou-se uma nova técnica, o Observador de Estados; O Observador de Estados é um dispositivo computacional que avalia os estados do sistema baseados nos dados de entrada e saída, $u(k)$ e $y(k)$. Assim como a realimentação de estados, onde o sistema tem que ser completamente controlável, a reconstrução dos estados do observador de estados, só será possível se o sistema for completamente observável.

Para verificar a observabilidade de um sistema, é necessário construir a matriz de observabilidade, com base nos dados experimentais de entrada e de saída do sistema:

$$M_o = \begin{bmatrix} C & GC & G^2C & \dots & G^{n-1}C \end{bmatrix} \quad (56)$$

A diferença do posto de M_o ao número de estados do sistema, é igual ao número de estados não observáveis do sistema. A equação (57) define a condição de observabilidade.

$$\text{posto}(M_o) = \text{posto} \begin{pmatrix} C & GC & G^2C & \dots & G^{n-1}C \end{pmatrix} = n \quad (57)$$

A proposta do observador é de se utilizar os estados reconstituídos denominados $\hat{x}$ na realimentação, conforme a equação (58):

$$u(k) = -K \hat{x} \quad (58)$$

O método utilizado para reconstruir os estados é similar em tempo real às equações dinâmicas e de saída com as mesmas condições de entrada e saída do sistema real. Porém o sistema real esta sujeito a perturbações que o modelo simulado não está (OGATA, 1994).

Então a diferença entre a saída real $y(k)$ e a saída simulada $\tilde{y}$ também é uma entrada para o observador, conforme a Figura 42:

Figura 42 – Observação de Estados.

A equação (59) ilustra a nova equação dinâmica e a equação de saída do observador, e a matriz L , denominada como a matriz de ganho do observador, deve ser projetada de modo que $\hat{x}(k)$ convirja para $x(k)$ em um espaço de tempo desejável, conforme (60)

$$\hat{x}(k+1) = G\hat{x}(k) + Hu(k) + L[y(k) - \tilde{y}(k)] \quad (59)$$

$$e(k) = x(k) - \tilde{x}(k) \quad (60)$$

Aplicando a realimentação de estados da equação (58), na equação (59), obtêm-se a nova equação do sistema em malha fechada com o observador em (61), obtendo também a nova equação característica do sistema, conforme (62):

$$\begin{bmatrix} x(k+1) \\ e(k+1) \end{bmatrix} = \begin{bmatrix} G - HK & HK \\ 0 & G - LC \end{bmatrix} \begin{bmatrix} x(k) \\ e(k) \end{bmatrix} \quad (61)$$

$$\begin{vmatrix} zI - G + HK & -HK \\ 0 & zI - G + LC \end{vmatrix} = 0 \quad (62)$$

A Figura 43 ilustra um sistema regulado com o observador:

Figura 43 – Realimentação de estados estimados.

Nota-se que os pólos da realimentação, da matriz K , e os pólos do observador, da matriz L , podem ser projetados de maneira independente, onde o comportamento da dinâmica do sistema será uma composição da dinâmica do sistema realimentado com a dinâmica do observador.

3 – Sistemas de Controle com Entrada Referência

Quando se desenvolve projeto de observadores de estado e por realimentação de estados, deve-se projetar esses controladores para que siga mudanças definidas pela entrada referência, conforme a Figura 44:

Figura 44 – Realimentação de estados estimados com entrada referência.

Observando a dinâmica do sistema em malha fechada com a entrada referência, a entrada afetará a planta e o estimador no mesmo instante, mantendo o erro nulo do estimador nulo durante a fase transitória, não excitando os seus pólos (OGATA, 1994).

Sabemos que a alocação de pólos determina a resposta desejada do sistema, com as n equações características para n ordem. Já na parte do observador, são determinadas as equações contendo o erro do estimador. Assim o produto de ambas as partes resulta na equação característica geral do sistema.

É necessário calcular um ganho ótimo, K_0 , que será aplicado na entrada do sistema, onde serão obtidos novos ganhos para o sistema, que determinarão os comportamentos dos estados a uma entrada unitária. A equação (63) determina o cálculo de K_0 da entrada referência:

$$\lim_{k \rightarrow \infty} y(k) = \lim_{z \rightarrow 1} \left[(1 - z^{-1}) Y(z) \right] \quad (63)$$

LITERATURA REFERENCIADA

- AGUIRRE, L. A. (2000) *Introdução à Identificação de Sistemas – Técnicas Lineares e Não-Lineares Aplicadas a Sistemas Reais*. Belo Horizonte, Editora UFMG.
- AGUIRRE, L. A.; RODRIGUES, G. G.; JÁCOME, C. R. F. (1997). Identificação de Sistemas Não-Lineares Utilizando Modelos NARMAX Polinomiais – Uma Revisão e Novos Resultados. In: SBA CONTROLE E AUTOMAÇÃO, 1997, Belo Horizonte, *Anais*, Belo Horizonte, v.9, n.2, p.90-106.
- BANGEMANN, T.; HAHNICHE, J.; NEUMANN, P. (1998). Integration of Fieldbus Systems in Computer-Aided Facility Management. In: PROCEEDINGS OF THE 24TH ANNUAL CONFERENCE OF THE IEEE INDUSTRIAL ELECTRONICS SOCIETY, 1998, Aachen, *Proceedings*, v. 3, p.1835-1840.
- BERGE, J. (2000). Building Homogenous System Using the Fieldbus Programming Language and HSE. In: THE INTERNATIONAL FORUM FOR ADVANCING MEASUREMENT AND CONTROL TECHNOLOGIES – ISATECH 2000, 2000, New Orleans, *Proceedings*, New Orleans, ISA, p. 1-10.
- BLACKWELL, R. (1996). Microsoft Solutions for Manufacturing Industry. In: IEE COLLOQUIUM ON NEXT GENERATION MANUFACTURING: FUTURE TRENDS IN MANUFACTURING AND SUPPLY CHAIN MANAGEMENT, 1996, London, *Proceedings*, London, p.647-727.
- BLEVINS, T. L. (1999). Advanced Control Capability of Fieldbus Devices. In: THE INTERNATIONAL FORUM FOR ADVANCING MEASUREMENT AND CONTROL TECHNOLOGIES – ISATECH 1999, 1999, Philadelphia, *Proceedings*, Philadelphia, p. 1-10.
- BOSSLEY, K. M. (1997). Neurofuzzy Modelling Approaches in System Identification, Tese de Doutorado, University of Southampton, Southampton.
- BRANDAO, D. (2000). *Bloco Funcional para Controle Fieldbus por Variáveis de Estado*. 117p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2000.
- CANTÙ, M. (2002). *Dominando o Delphi 6, “A Bíblia”*. São Paulo, Makron Books.

- EMERSON PROCESS MANAGEMENT (2003). Foundation Fieldbus in the Real World. Disponível em: <<http://www.easydeltav.com/keytechnologies/fieldbus/seminar.asp>>. Acesso em 15 jan. 2003.
- EMERSON PROCESS MANAGEMENT (2003). Success Stories – OPC. Disponível em: <<http://www.easydeltav.com/successstories>>. Acesso em 05 fev. 2003.
- FAN, H.; SÖDERSTÖRM, T. (1997). Parameter Estimation of Continuous-Time Ar Processes Using Integrated Sampling, In: IEEE CONFERENCE ON DECISION AND CONTROL, 1997, San Diego, *Proceedings*, San Diego, v.4, p.3474-3475.
- FAYAD, C. A. (2000). Building Homogenous System Using the Fieldbus Programming Language and HSE. In: THE INTERNATIONAL FORUM FOR ADVANCING MEASUREMENT AND CONTROL TECHNOLOGIES – ISATECH 2000, 2000, New Orleans, *Proceedings*, New Orleans, p. 1-10.
- FIELDBUS FOUNDATION (2002). *Technical Overview – Foundation Fieldbus*. Austin, Texas.
- FONSECA, M. O. (2002). Comunicação OPC – Uma Abordagem Prática. In: VI Seminário de Automação e Processos, Associação Brasileira de Metalurgia e Materiais, 2002, Vitória, *Anais*, ABM, p.41-51.
- FURTADO, E. C.; MENDES, E. M. A. M.; NEPOMUCENO, E. G.; SILVA, V. V. R. (2002). Identificação de Sistemas Dinâmicos Não-Lineares Contínuos Utilizando Modelos NARAX: Estudo de Caso de um Forno a Arco Elétrico. In: XIV – CONGRESSO BRASILEIRO DE AUTOMÁTICA, 2002, São João Del Rei, *Anais*, São João Del Rei, p. 2150-2155.
- HABER, R. (1985). Nonlinear Tests for Dynamic Process. In: IFAC IDENTIFICATION AND SYSTEM PARAMETER ESTIMATION, York, *Proceedings*, York, p. 409-413.
- HADELLIS, L.; KOUBIAS, S.; MAKIOS, V. (2002). An Integrated Approach for an Interoperable Industrial Networking Architecture Consisting of Heterogeneous Fieldbuses. *Computers in Industry*, Patras, v.49, n.3, p.283-298, Dec.
- HUZMEZAN, M.; GOUGH, W. A.; DUMONT, G. A.; KOVAC, D. (2001). Time Delay Integrating Systems: a Challenge for Process Control Industries. A Practical Solution. *Control Engineering Practice*, Vancouver, v.10, n.10, p.1153-1161, Oct.
- IWANITZ, F.; LANGE, J. (2002). *OPC Fundamentals, Implementation and Application*. Segunda Edição, Munique, Hüthling Verlag Heidelberg.
- IWASE, M.; IKUBO, H.; HATAKEYAMA, S.; FURUTA, K. (2002). An Identification Method for Continuous-Time Transfer Functions Based on Nonlinear Optmization, In: IEEE CONFERENCE OF THE 2002 28TH ANNUAL INDUSTRIAL ELECTRONICS SOCIETY, 2002, Tokyo, *Proceedings*, Tokyo, v.3, p. 1978-1983.

- JANKE, M. (1999). OPC – Simple Software Integration for Legacy Systems. In: IEEE ADVANCED PROCESS CONTROL APPLICATIONS FOR INDUSTRY WORKSHOP, 1999, Vancouver. *Workshop*. Vancouver, p.9-14.
- JANKE, M. (2000). OPC – Plug and Play Integration to Legacy Systems. In: IEEE CONFERENCE RECORD OF 2000 ANNUAL PULP AND PAPER INDUSTRY TECHNICAL CONFERENCE, 2000, Atlanta. *Conference*. Atlanta, p.68-72.
- JOHANSSON, R. (1994). Identification of Continuous-Time Models, *IEEE Transactions on Signal Processing*, Lund, v. 42, n.4, p.887-897, Apr.
- KAMWA, I. (2000). Using MIMO System Identification for Modal Analysis and Global Stabilization of Large Power Systems, In: IEEE POWER ENGINEERING SOCIETY SUMMER MEETING, Seattle, *Proceedings*, Seattle, v.2, p. 817-822.
- KAPSALIS, V.; CHARATSI, K.; KALOGERAS, A.P.; PAPADOPOULOS, G. (2002). Web Gateway: A Platform for Industry Services over Internet. In: IEEE INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS, 2002, Patras, IEEE, v.1, p.73-77.
- KAPSALIS, V.; KOUBIAS, S.; PAPADOPOULOS, G. (2002). OPC-SMS: A Wireless Gateway to OPC-Based Data Sources. *Computers Standards and Interfaces in Computing*, Patras, v.24, n.5, p.437-451, Apr.
- KLAUSUTZ, I.; MEDVEDKOV, V.; ANISIMOV, V. (2000). Information Systems in Demand Side Management. In: IEEE THE 4TH KOREA-RUSSIA INTERNATIONAL SYMPOSIUM ON SCIENCE AND TECHNOLOGY, 2000, Korus, *Proceedings*, Korus, v. 2, p. 97-102.
- KLEINES, H.; WUSTNER, P.; SETTKE, K.; ZWOLL, K. (2001) Access to Industrial Process Periphery via Java for Process Control (JFPC). *IEEE Transactions On Nuclear Science*. Valencia, v.49, n.2, p.465-468, Apr.
- LENNOX, B.; MONTAGUE, G.A.; FRITH, A. M., GENT, C.; BEVAN, V. (2001). Industrial Application of Neural Networks - An Investigation, *Journal of Process Control*, Manchester, v.11, n.5, p.497-507, Oct.
- LING, Z.; YU J. (2002). The Design of SCADA Based on Industrial Ethernet. In: IEEE PROCEEDINGS OF THE 4TH WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION, 2002, Shanghai. *Proceedings*. Shanghai, IEEE, v.4, p.10-14.
- LJUNG, L. (1999) *System Identification – Theory for the User*. Segunda Edição, Englewood, Prentice Hall.
- NINNESS, B.; GOMEZ, J.C.; WELLER, S. (1995). MIMO System Identification Using Orthornormal Basis Functions, In: PROCEEDINGS OF THE 34ST IEEE CONFERENCE ON DECISION AND CONTROL, New Orleans, *Proceedings*, New Orleans, v. 1, p. 703-708.

- OGATA, K. (1994). *Discrete-time control systems*. Segunda Edição, New Jersey, Pretence-Hall.
- OGATA, K. (1997). *Engenharia de Controle Moderno*. Terceira Edição. Rio de Janeiro, Prentice Hall do Brasil.
- OPC FOUNDATION (2003). OLE for Process Control, Disponível em: <<http://www.opcfoundation.org>> , Acesso em 10 ago. 2003.
- PAIVA, R. P. P. C. (1999). *Identificação Neuro-Difusa – Aspectos de Interpretabilidade*. 163p. Dissertação (Mestrado) – Departamento de Engenharia Informática, Universidade de Coimbra, Coimbra, 1999.
- PATTLE, R.; RAMISCH, J. (1997). OPC the Factory Standard for Real Time Communication. In: PROCEEDINGS OF THE JOINT WORKSHOP ON PARALLEL AND DISTRIBUTED REAL-TIME SYSTEMS, Geneva, *Proceedings*, Geneva, p. 289-294.
- RASMUSSEN, K. H.; JØRGENSEN, S. B. (1999). Parametric Uncertainty Modeling for Robust Control: A Link to Identification. *Computers & Chemical Engineering*, Lyngby, v.23, n.8, p.987-1003, Aug.
- REHBEIN D. A. (2000). OLE for Process Control (OPC) Primer, Disponível em: <<http://www.easydeltav.com>> Austin, Acesso em 05 fev. 2003.
- RIVERA, D.E; GAIKWAD, S.V. (1996). Digital PID Controller Design Using ARX Estimation, *Computers & Chemical Engineering*, Arizona, v.20, n.11, p.1317-1334, Sep.
- RODRIGUES, G. G. (2000). *Identificação de Sistemas Dinâmicos Não-Lineares Utilizando Modelos NARMAX Polinomiais – Aplicação a Sistemas Reais*. 105p. Dissertação (Mestrado) – Universidade Federal de Minas Gerais, Belo Horizonte, 2000.
- SANTOS, S. C. (2000). *Identificação Multivariável Aplicada aos Processos Químicos*. 166p. Dissertação (Mestrado) – Universidade Federal de São Carlos, São Carlos, 2000.
- SANZ, R.; ALONSO, M. (2001) Corba for Control Systems. *Annual Reviews in Control*, Madri, v.25, p.169-181.
- SENDELBACH, S. P. (1999). Utilizing Dynamic Simulation as a Test Environment for Advanced Control Application . In: THE INTERNATIONAL FORUM FOR ADVANCING MEASUREMENT AND CONTROL TECHNOLOGIES – ISATECH 1999, 1999, Philadelphia, *Proceedings*, Philadelphia, p. 1-8.
- SHIMANUKI, Y. (1999). Ole for Process Control (OPC) for New Industrial Automation Systems. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS, 1999, Tokio. *Proceedings*. Tokio, v.6, p.1048-1050.

- SIMANI, S.; FANTUZZI, C.; BEGHELLI, S. (2000). Diagnosis Techniques for Sensor Faults of Industrial Processes, *IEEE Transactions on Control Systems Technology*, Ferrara, v.8, n.5, p.848-855, Sep.
- SITAO, W.; QINGQUAN, Q. (2000). Combining OPC With Autonomous Decentralized Systems. In: IEEE INTERNATIONAL WORKSHOP ON AUTONOMOUS DECENTRALIZED SYSTEMS, 2000, Chengdu. *Proceedings*. Chengdu, p.126-129.
- SITAO, W.; QINGQUAN, Q. (2000). Using Device Driver Software In SCADA Systems. In: IEEE POWER ENGINEERING SOCIETY WINTER MEETING, 2000, Sichuan. *Conference*. Sichuan, v.3, p.23-27.
- SJÖBERG, J., HJALMARSSON, H., LJUNG, L. (1994). Neural Networks in System Identification, In: PROCEEDINGS OF THE 10TH IFAC SYMPOSIUM ON SYSTEM IDENTIFICATION, 1994, *Proceedings*, p. 207-212.
- SÖDERSTRÖM, T.; FAN, H.; CARLSSON, B.; BIGI, S. (1997). Least Square Parameter Estimation of Continuous-Time ARX Model from Discrete-Time Data, *IEEE Transactions on Automatic Control*, Uppsala, v.42, n.5, p.659-673, May.
- STOICA, P.; JANSSEN, M.; (1999). A Transfer Function Approach to MIMO System Identification, In: PROCEEDINGS OF THE 38ST IEEE CONFERENCE ON DECISION AND CONTROL, Phoenix, *Proceedings*, Phoenix, v. 3, p. 2400-2405.
- TOUHAMI, O.; IBTIOUEN, R.; GUECHTOULI, A.; IUNG, C. (2000). Identification of SIMO System: Application to an Induction Machine, In: THE 2000 IEEE INDUSTRY APPLICATIONS CONFERENCE, Rome, *Conference*, Rome, v. 2, p. 1148-1152.
- WANG, S.; XIE, J. (2002) Integrating Building Management System and facilities Management on the Internet. *Automation in Construction*, Hong Kong, v.11, n. 6, p.707-715, Oct.
- YANG, S. H.; CHEN, X.; ALTY, J. L. (2003). Design Issues and Implementation of Internet-Based Process Control Systems. *Control Engineering Practice*, Leicestershire, v.11, n.6, p.709-720, Jun.
- YOUNG, B. R.; ALLEN, R. M. (2000). Multi-input, Multi-Output Identification of a Pilot-Plant Climbing Film Evaporator. *Control Engineering Practice*, Sydney, v.3, n.8, p. 1067-1073, Feb.
- ZEILMANN, R. P. (2002). *Uma Estratégia para Controle e Supervisão de Processos Industriais via Internet*. 140p. Dissertação (Mestrado) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.
- ZHU, Y. (2002). Estimation of Nonlinear ARX Models, In: PROCEEDINGS OF THE 41ST IEEE CONFERENCE ON DECISION AND CONTROL, Holand, *Proceedings*, Holand, v. 2, p. 2214-2219.

LITERATURA CONSULTADA

- ABATE, M. et al. (1994). Application of Some New Tools to Robust Stability Analysis of Spark Ignition Engines: A case Study. *IEEE Trans. Contr. Syst. Technol.* v.2, n.1, p.22-30, Mar.
- BARCLAY, K, A. (1990). *ANSI C: problem-solving and programming*. Edinburgh, Prentice Hall
- BRANDÃO, D.; DA CUNHA, M. J.; PINOTTI JÚNIOR, M. Analysis of Fieldbus Communication Traffic based on Real Time Bus Monitoring and Configuration Data. WTR 2004 - VI Workshop de Tempo Real. 2004, Gramado - RS.
- BRANDÃO, D.; DA CUNHA, M. J.; PINOTTI JÚNIOR, M. Fieldbus Control System Project Support Tool based on Experimental Analysis and Modeling of Communication Bus. IEEE ICIT'04 Tunisia 2004.
- BRANDÃO, D.; DA CUNHA, M. J.; PINOTTI JÚNIOR, M. Simulação Operacional de Bloco Funcional Foundation Fieldbus. 3º Congresso Internacional de Automação, Sistemas e Instrumentação - ISA Show. 2003, São Paulo.
- DA CUNHA, M. J.; BRANDÃO, D.; PINOTTI JÚNIOR, M. Simulação Operacional em Tempo Real de Sistemas de Controle Fieldbus Foundation. VI Seminário de Automação de Processos, 2002, Vitória.
- HEMERLY, E, M. (2000). *Controle com Computador de Sistemas Dinâmicos*. Segunda Edição, São Paulo, Edgard Blücher.
- SMAR EQUIPAMENTOS INDUSTRIAIS (1999). *Function Block instruction manual*. Sertãozinho.