Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Programa de Pós-Graduação em Engenharia Elétrica

Projeto e Implementação em Ambiente *Foundation Fieldbus* de Filtragem Estocástica Baseada em Análise de Componentes Independentes

Autor: Isabele Morais Costa
Orientador: Prof D.Sc. Adrião Duarte Dória Neto

Natal, Julho de 2006

Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Programa de Pós-Graduação em Engenharia Elétrica

Isabele Morais Costa

Projeto e Implementação em Ambiente *Foundation Fieldbus* de Filtragem Estocástica Baseada em Análise de Componentes Independentes

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Rio Grande do Norte, como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências.

Natal, Julho de 2006

Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Programa de Pós-Graduação em Engenharia Elétrica

Aprovada em 31 de julho de 2006 pela comissão examinadora, formada pelos seguintes membros:

Prof. D.Sc. Adrião Duarte Dória Neto (Orientador)
DCA - UFRN

Prof. D.Sc. Roberto Célio Limão de Oliveira (Examinador Externo)
UFPA

Prof. D.Sc. Jorge Dantas de Melo (Examinador Interno)
DCA - UFRN

Prof. D.Sc. Luiz Affonso Henderson Guedes de Olivera (Examinador Interno)
DCA - UFRN

Agradecimentos

Agradeço a Deus em primeiro lugar por sempre ter me dado forças para continuar nos momentos de cansaço. Aos meus pais pelo exemplo e por sempre acreditarem em mim, à toda a minha família pelo exemplo de união, que me deram base para enfrentar os desafios. Aos muitos amigos e namorado pelo conforto, incentivo, companheirismo e amizade verdadeira.

Agradeço aos professores que ajudaram na minha formação, em especial aos professores Adrião, Jorge, Affonso, Alberto com quem trabalhei nos últimos anos e que para mim são exemplos de pessoas e de amor à profissão.

Agradeço à FINEP, CENPES e CNPQ pelo apoio e suporte financeiro para a realização das atividades deste trabalho.

Agradeço aos meus colegas de laboratório que passaram a ser amigos e que sempre se mostraram dispostos a me ajudar e esclarecer minhas dúvidas.

Resumo

Este trabalho propõe o desenvolvimento de um sistema de filtragem composto por um algoritmo inteligente, capaz de separar informação e ruídos provenientes de sensores de campo interligados por uma rede *Foundation Fieldbus* (FF). A implementação do algoritmo será feita tanto em blocos funcionais padrão da própria rede FF, com treinamento *on-line* via OPC (*OLE for Process Control*), como em tecnologia embarcada em um DSP (*Digital Signal Processor*) que interage com os dispositivos *fieldbus*. A técnica ICA (*Independent Component Analysis*), que explora a possibilidade de separar uma mistura de sinais baseando-se no fato de que eles são estatisticamente independentes, foi escolhida para realizar este processo de separação cega de fontes (*Blind Source Separation - BSS*). O algoritmo e suas implementações serão apresentados, bem como os resultados obtidos.

Abstract

This work considers the development of a filtering system composed of an intelligent algorithm, that separates information and noise coming from sensors interconnected by Foundation Fieldbus (FF) network. The algorithm implementation will be made through FF standard function blocks, with on-line training through OPC (OLE for Process Control), and embedded technology in a DSP (Digital Signal Processor) that interacts with the fieldbus devices. The technique ICA (Independent Component Analysis), that explores the possibility of separating mixed signals based on the fact that they are statistically independent, was chosen to this Blind Source Separation (BSS) process. The algorithm and its implementations will be presented, as well as the results.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

No processo de extração da informação de uma rede de sensores é comum a adição de um sinal de ruído ao sinal correspondente à informação fornecida. O ruído e a informação, misturados, podem apresentar-se na mesma faixa de freqüência [Smaragdis, 1998], o que dificulta a utilização de filtragem seletiva em freqüência no processo de separação dos mesmos. Esta situação caracteriza-se como um problema de separação cega de fontes (Blind Source Separation - BSS) onde a partir do conjunto de sinais misturados é possível recuperar os sinais fontes.

Neste trabalho a proposta é a implementação de um sistema que, baseado na técnica de Análise de Componentes Independentes (ICA - Independent Component Analysis), a qual explora a possibilidade de separar uma mistura de sinais baseando-se no fato de que eles são estatisticamente independentes, apresenta uma solução eficaz ao problema da extração de ruído em uma rede de sensores.

A separação cega de fontes por Análise de Componentes Independentes recebeu atenção devido suas potenciais aplicações em processamento de sinais como em sistemas de reconhecimento de voz, telecomunicações e processamento de sinais médicos. O objetivo da técnica ICA é recuperar as fontes independentes a partir de sinais observados dos sensores que são misturas lineares desconhecidas dos sinais fontes independentes não-observados. Em contraste com as transformações baseadas em correlação, tais como a Análise do Componentes Principais (PCA - Principal Component Analysis), ICA não somente descorrelaciona os sinais (estatísticas de segunda ordem) mas também reduz as dependências estatísticas de alta ordem, tentando tornar os sinais tão independentes quanto possível.

A Técnica de Análise de Componentes Independentes foi introduzida no início dos anos 80 por J. Hérault, C. Jutten, e B. Ans [Hérault and Ans, 1984, Hérault et al., 1985, Ans et al., 1985]. Como revisto recentemente por Jutten [Jutten and Taleb, 2000], o problema surgiu, primeiramente, em 1982 em um ajuste neurofisiológico.

Em um modelo simplificado de código motor em uma contração muscular, as saídas $x_1(t)$ e $x_2(t)$ eram dois tipos de medidas de sinais sensoriais de contração muscular, e s_1, s_2 eram a posição angular e a velocidade dos movimentos, respectivamente. O sistema nervoso deve ser capaz de inferir os sinais de posição e velocidade, s_1 e s_2, através das medidas x_1 e x_2. Logo notou-se que uma possibilidade para isto é aprender o modelo inverso usando o princípio de descorrelação não-linear em uma simples rede neural.

Durante toda a década de 80, a técnica ICA foi mais conhecida entre os pesquisadores franceses, com limitadas influências internacionais. Os poucos trabalhos de ICA presentes nos Congressos Internacionais de Redes Neurais na metade da década de 80 se concentraram nas áreas de back-propagation, Redes de Hopfield e Mapas Auto-Organizáveis de Kohonen, que eram redes muito trabalhadas nessa época. Um outro campo relacionado foi o de Análise Espectral de alta ordem, em que o primeiro Workshop Internacional foi organizado em 1989. Neste Workshop, os artigos sobre ICA foram de J.F. Cardoso [Cardoso, 1989] e P.Comon [Cichocki et al., 1998]. Cardoso usava métodos algébricos, especialmente cumulantes de alta ordem, os quais eventualmente conduziram ao algoritmo JADE (*Joint Approximate Diagonalization of Eigenmatrices*) [Cardoso and Souloumiac, 1993]. O uso de cumulantes de quarta-ordem tinha sido proposto antes por J.L. Lacoume [Lacoume and Ruiz, 1988]. Na literatura de processamento digital de sinais, artigos clássicos propostos pelo grupo francês foram [Jutten and Hérault, 1991, Comon et al., 1991, E.Sorouchyari, 1991, Comon, 1994]. Uma lista mais completa das referências está em [Jutten and Taleb, 2000].

Em processamento digital de sinais, já existiam pesquisas relacionadas ao problema da deconvolução cega de sinais [Donoho, 1981, Shalvi and Weinstein, 1990]. Em particular, os resultados utilizados em deconvolução cega de multi-canais são muito similares à técnica ICA.

O trabalho dos cientistas na década de 80 foi continuado por, entre outros, A.Cichocki e R.Unbehauen, que foram os primeiros a proporem um dos mais populares algoritmos de ICA [Cichocki and Moszczynski, 1992, Cichocki and Unbehauen, 1996, Cichocki et al., 1994]. Outros artigos de ICA e separação de sinal da década de 90 são [Burel, 1992, Nadal and Parga, 1994]. O "PCA não-linear" aproximado foi introduzido pelos presentes autores em [Oja et al., 1991, Karhunen and Joutsensalo, 1994]. Assim, antes do meio da década de 90, ICA ia se consolidando através de pesquisas pequenas e estritas.

Na metade da década de 90, ICA alcançou maior atenção e interesse após A.J. Bell e T.J. Sejnowski publicarem sua aproximação baseada no princípio do INFOMAX [Bell and Sejnowski, 1994, Bell and Sejnowski, 1995]. Este algoritmo foi refinado por S.I. Amari e seus colegas de trabalho usando o gradiente natural [Amari et al., 1996], e foram estabelecidas suas conexões fundamentais para a estimação da máxima verossimilhança, assim como o algoritmo de Cichocki-Unbehauen. Alguns anos mais tarde, os presentes autores apresentaram o algoritmo de ponto-fixo ou FastICA, [Hywarinen and Oja, 1997, Hyvärinen, 1997, Hyvarinen, 1999], que tem contribuído para a aplicação da técnica ICA em vários problemas devido sua computação eficiente.

Desde a metade da década de 90, tem crescido o número de artigos, workshops e sessões especiais dedicadas à ICA. O primeiro workshop internacional de ICA foi realizado em Aussois, França, em janeiro de 1999, e o segundo workshop em Helsink, Finlândia, em junho de 2000. Ambos contaram com mais de 100 pesquisadores que trabalhavam com ICA e separação cega de sinais, e contribuíram para a transformação de ICA em um campo de pesquisa estável e maduro.

Neste trabalho, propõe-se a aplicação da técnica de Análise de Componentes Independentes no ambiente de redes industriais *Fieldbus* para a filtragem de ruídos provenientes de sensores ou de uma rede de sensores. Serão adotadas três abordagens. A primeira emprega blocos funcionais *Foundation Fieldbus* (FF) para executar o algoritmo FastICA [Hywarinen, 1999, Hywarinen and Oja, 1999] localmente nos sensores de campo, visando uma forma robusta de filtragem de ruído, mantendo o padrão de interoperabilidade do sistema. A segunda também visa interligação com instrumentos de campo *Foundation Fieldbus*, porém, desta vez, o algoritmo será acoplado a estes dispositivos através de sua embarcação em

uma placa de DSP. E por fim, a terceira propõe um treinamento *on-line* do algoritmo FastICA através do protocolo OPC, (*OLE for Process Control*) utilizando tecnologia OLE (*Object Linking and Embedding*), atualizando os pesos utilizados na filtragem realizada nos Blocos Funcionais.

O padrão OPC estabelece as regras para que sejam desenvolvidos sistemas com interfaces padrões para comunicação dos dispositivos de campo (controladores, sensores, etc) com sistemas de monitoração, supervisão e gerenciamento (SCADA, MES, ERP, etc). A tecnologia OLE foi desenvolvida pela Microsoft em meados de 1990, para suprir a necessidade de se integrar diferentes aplicações dentro da plataforma Windows, de forma a solucionar os problemas de desempenho e confiabilidade do até então utilizado padrão DDE (*Dynamic Data Exchange*).

O objetivo, além da extração de ruídos das medidas dos sensores validando o sistema proposto, é o desenvolvimento de novas funcionalidades para sensores inteligentes [Tian et al., 2000], conectados através de redes de dispositivos de campo (*Fieldbus Devices*) que possam ser interligados utilizando o protocolo *Foundation Fieldbus* [Foundation, 2003].

1.1 Organização da Dissertação

O presente trabalho está organizado da seguinte forma: no Capítulo 2, é apresentada uma visão geral sobre as redes industriais padronizadas pela *Fieldbus Foundation*. No Capítulo 3, aborda-se a tecnologia OPC e suas características. Já no Capítulo 4 é realizada uma explanação sobre a técnica de Análise de Componentes Independentes utilizada para a construção do sistema de filtragem. No Capítulo 5, descreve-se as abordagens propostas para inserção da técnica em ambiente *Foundation Fieldbus*. No Capítulo 6 são apresentados os experimentos realizados para validação das abordagens propostas e os resultados, no Capítulo 7 as conclusões e por último as referências bibliográficas que serviram de base de estudo para o desenvolvimento do trabalho.

Capítulo 2

Redes de Dispositivos de Campo *Foundation Fieldbus*

Uma das principais tendências na evolução de redes industriais vem sendo o incremento da inteligência integrada aos instrumentos de campo [Berge, 2001]. O processamento das funções de controle, que normalmente era efetuado por um equipamento centralizado, agora pode ser realizado por processadores incorporados a dispositivos, tais como sensores e atuadores localizados no chamado "chão de fábrica". Esta concepção aumenta a confiabilidade do sistema de controle, pois falhas localizadas em determinados setores de uma planta podem ser isoladas e os equipamentos, funcionando corretamente, podem adotar medidas para contornar os problemas gerados por outros instrumentos.

A tecnologia *Foundation Fieldbus* (FF) é atualmente considerada uma das melhores soluções dentro desta abordagem. As normas FF padronizam não apenas a arquitetura dos protocolos de comunicação entre os instrumentos de campo como também descreve componentes de software de alto nível, que permitem a construção de aplicações de controle pelo usuário. Os instrumentos FF possuem uma razoável capacidade interna de processamento que lhes permite o processamento de várias funções de controle baseadas em variáveis contínuas ou discretas. Esta capacidade também viabiliza a implementação de uma pilha de protocolos complexos que possibilitam a comunicação digital *full duplex* entre os vários equipamentos de campo ligados por um barramento FF.

Por ser um padrão aberto, o *Foundation Fieldbus* possibilita dispositivos de diferentes fabricantes serem integrados em um único sistema (interoperabilidade).

Isto é possível apenas quando todos os dispositivos seguem exatamente a especificação FF.

Os dispositivos são independentes do hardware de configuração devido à tecnologia de descritores de dispositivos (*DD - Device Descriptions*), isto é, qualquer sistema de controle ou computador pode operar com o dispositivo se tiver o DD dele, o que se assemelha aos *drivers* de um computador pessoal. Com a interoperabilidade, um dispositivo FF pode ser substituído por um dispositivo similar de um outro fornecedor com maior funcionalidade na mesma rede FF, mantendo as características originais. Isto permite aos usuários mesclarem dispositivos de campo e sistemas de vários fornecedores. Dispositivos FF individuais podem também transmitir e receber a informação de multi-variáveis, comunicando-se diretamente um com o outro pelo barramento FF, permitindo que novos dispositivos sejam adicionados ao barramento sem interromper o controle.

Entre os principais serviços e funções providos pela *Foundation Fieldbus*, estão:

- segurança intrínseca para áreas de risco;
- barramento com alimentação para os dispositivos de campo;
- topologia em linha ou em árvore;
- capacidade de comunicação multi-mestre;
- comportamento dinâmico determinístico;
- transferência de dados distribuída (DDT);
- modelo de blocos padronizado para interfaceamento uniforme dos dispositivos (interoperabilidade, intercambialidade) e
- opções de extensões flexíveis baseada em descritores de dispositivo (DD).

2.1 Arquitetura do Protocolo *Foundation Fieldbus*

A arquitetura FF é baseada no modelo ISO/OSI, em que são definidas diversas camadas de software em diferentes níveis. Cada nível provê serviços à camada

acima e serve-se de serviços disponibilizados pela camada imediatamente abaixo. A correspondência entre a pilha de protocolos FF e o modelo OSI é ilustrado pela Figura 2.1. No escopo deste trabalho, o interesse é apenas na camada de aplicação do usuário. Detalhes da pilha de comunicações podem ser obtidos em [FF-581F12, 2001].

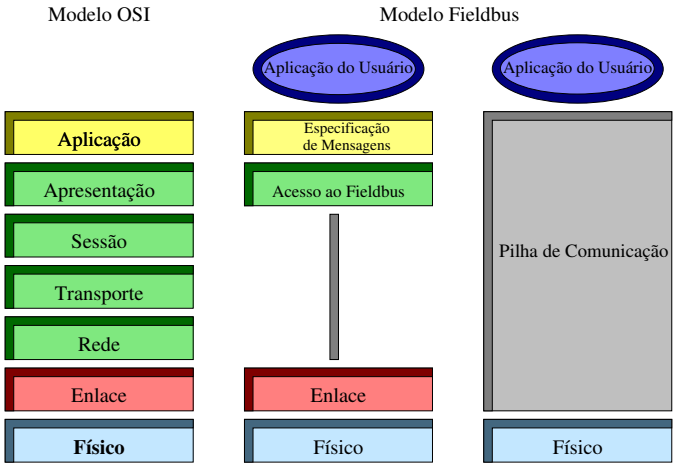

Figura 2.1: Modelo OSI.

Observa-se via Figura 2.1 que a pilha de protocolos FF inclui uma camada adicional no nível mais alto, chamada camada de aplicação do usuário, baseada em blocos, que define funções uniformes de dispositivos e interfaces para aplicação, de modo que outros dispositivos da rede e *softwares* aplicativos possam usar essas interfaces para acessar as funções e os parâmetros dos dispositivos de campo.

A *Foundation Fieldbus* associa todas as funções e dados dos dispositivos a três tipos de blocos, cuja associação depende do tipo do dispositivo. Dependendo da funcionalidade do dispositivo, ele pode ser descrito da seguinte maneira:

- bloco de recurso (*Resource Block*);
- um ou múltiplos blocos funcionais e
- vários blocos transdutores, se necessário.

O bloco de recurso descreve características de um dispositivo, como o nome, o fabricante, o número serial, as versões do *hardware* e *firmware*, etc.

Os blocos funcionais (BF) [FF-890F15, 2001], são objetos implementados por estruturas de software que realizam as funções comuns ao ambiente de controle de processo. São padronizados blocos funcionais FF tanto para funções simples, como entrada e saída, como para algumas funções de maior complexidade, como controladores. Lançando-se mão desses recursos, é possível se construírem aplicações de controle com certa facilidade já que os componentes básicos, representados pelos BF, já estão instalados nos equipamentos de campo e basta, apenas, que sejam instanciados e conectados adequadamente através de suas interfaces padronizadas.

A especificação FF define conjuntos de blocos funcionais padrão que podem ser usados para descrever as funcionalidades básicas. Eles estão listados na Tabela 2.1.

Tabela 2.1: Blocos funcionais típicos.

Sigla	Descrição do bloco
AI	*Analog Input* (entrada analógica)
AO	*Analog Output* (saída analógica)
B	*Bias* (bias)
CS	*Control Selector* (seletor de controle)
DI	*Discrete Input* (entrada discreta)
DO	*Discrete Output* (saída discreta)
ML	*Manual Loader* (carregador manual)
PD	*Proportional/Derivative* (proporcional/derivativo)
PID	*Proportional/Integral/Derivative* (proporcional/integral/derivativo)
RA	*Ratio* (taxa)

Os blocos transdutores expandem a complexidade e as aplicações possíveis de um dispositivo. Seus dados influenciam os parâmetros de entrada e saída dos blocos funcionais. Eles podem ser usados para calibrar, deslocar medidas, posicionar dados, linearizar características e converter unidades físicas.

Ao lado dos três tipos de blocos, os seguintes objetos também são definidos no modelo de blocos:

- *Link Objects* (ligações entre objetos) - definem as ligação entre os blocos funcionais, sendo essas internas em um dispositivo ou através da rede.
- *Alert Objects* (objetos de alerta) - permitem reportar alarmes e eventos na rede.

- *Trend Objects* (objetos de tendência) - permitem o acompanhamento de dados de blocos funcionais para acesso e análise a partir de sistemas de ato nível.
- *View Objects* (objetos de visualização) - são grupos predefinidos de dados e parâmetros de blocos que podem ser usados para ver e mostrá-los rapidamente de acordo com suas tarefas: controle de processos, configuração, manutenção, informações adicionais, etc.

Uma das aplicações mais comuns e ilustrativas em processos são os laços de controle. Normalmente, um laço de controle simples, implementado pelos componentes da tecnologia FF, é formado por um bloco de entrada, um bloco de controle e um bloco de saída. O bloco de entrada recebe e condiciona o valor lido pelo sensor. O bloco controlador calcula o valor de atuação e o bloco de saída ajusta aquele valor para as condições do atuador em uso.

Na Figura 2.2, um laço de controle típico é apresentado.

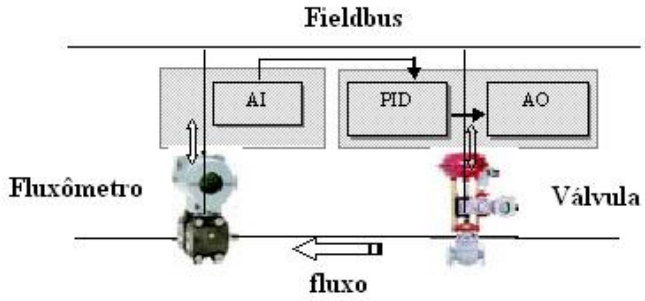

Figura 2.2: Laço de controle típico em FF.

Nesse laço, foram usados o bloco AI (*Analog Input*), integrado ao sensor de fluxo e os blocos PID (*Proporcional Integrativo Derivativo*) e AO (*Analog Output*), alojados na válvula. Em conjunto e comunicando-se pelo barramento FF, eles implementam um laço de controle para o fluxo na tubulação que passa pelo fluxômetro e é regulado pela válvula. O valor de fluxo lido pelo sensor é captado pelo bloco AI instanciado naquele instrumento. Nesse bloco, este valor é colocado em unidades de engenharia convenientes e limitado de acordo com os fatores de segurança requeridos. A saída do AI é transmitida periodicamente ao bloco PID situado na válvula, por meio de mensagens padronizados do protocolo FF. O bloco PID recebe a saída do AI através de um de seus parâmetros de entrada, executa o algoritmo próprio do bloco

PID, previamente configurado pelo usuário, e envia sua saída no bloco AO, situado também na válvula. No AO, o valor de atuação, recebido do PID, é condicionado em escala e unidades adequadas para as particularidades da válvula em uso. No instante seguinte, o fluxo afetado pela atuação da válvula é novamente lido pelo fluxômetro e o ciclo é reiniciado.

As interfaces dos blocos funcionais são formadas por parâmetros cujos valores podem transitar pela rede FF sempre que um bloco alojado em um determinado dispositivo precisar de valores gerados por blocos residentes em um outro dispositivo. A padronização das interfaces permite que BF implementados por diferentes fornecedores em diferentes equipamentos FF possam ser interligados sem problemas de compatibilidade. O usuário precisa apenas definir quais blocos deseja usar na sua aplicação, instanciá-los nos equipamentos que considere mais adequados, configurá-los e conectá-los convenientemente. A tarefa de conectar os blocos é normalmente realizada com auxílio de um software gráfico.

Vários blocos funcionais foram padronizados pela FF-890 Part 2 [FF-891F15, 2001], mas na FF-890 Part 3 [FF-892F15, 2001] são padronizados blocos de funções mais avançadas, incluindo o Bloco Aritmético, que foi utilizado neste trabalho. Esse bloco e o bloco AI são detalhados no Capítulo 5.

Capítulo 3

OLE for Process Control - OPC

Em 1995, algumas empresas se reuniram e criaram a OPC *Foundation* com o objetivo de desenvolver um padrão baseado na tecnologia OLE/DCOM para acesso a dados *on-line* dentro do sistema operacional Windows. A grande motivação para se criar o padrão OPC - *OLE for Process Control* [Fonseca and Seixas-Filho, 2005], foi a necessidade de se estabelecer um mecanismo padrão de comunicação entre diferentes fontes de dados em um Processo Industrial de Manufatura, sejam eles provenientes de equipamentos de campo ou até mesmo de outros arquivos de dados. Basicamente, o padrão OPC estabelece as regras para que sejam desenvolvidos sistemas com interfaces padrões para comunicação dos dispositivos de campo (CLPs, sensores, etc.) com sistemas de monitoração, supervisão e gerenciamento.

A arquitetura da Informação em um Processo Industrial de Manufatura, envolve três níveis, como observado na Figura 3.1.

Gerência de Campo: com o advento dos equipamentos de campo inteligentes, uma grande variedade de dados provenientes desses equipamentos, como dados de configuração e controle, podem ser disponibilizados para usuários ou mesmo para outras aplicações.

Gerência de Processo: a utilização de sistemas de controle do tipo Scada e SDCD permite o controle descentralizado de processos industriais. Os dados fornecidos podem ser considerados conjuntamente de modo a permitir uma gerência efetiva e integrada de todo o Processo Industrial.

Gerência do Negócio: é a integração das informações de "chão-de-fábrica"e dos dados de gerência individual de cada processo controlado com os dados corporativos

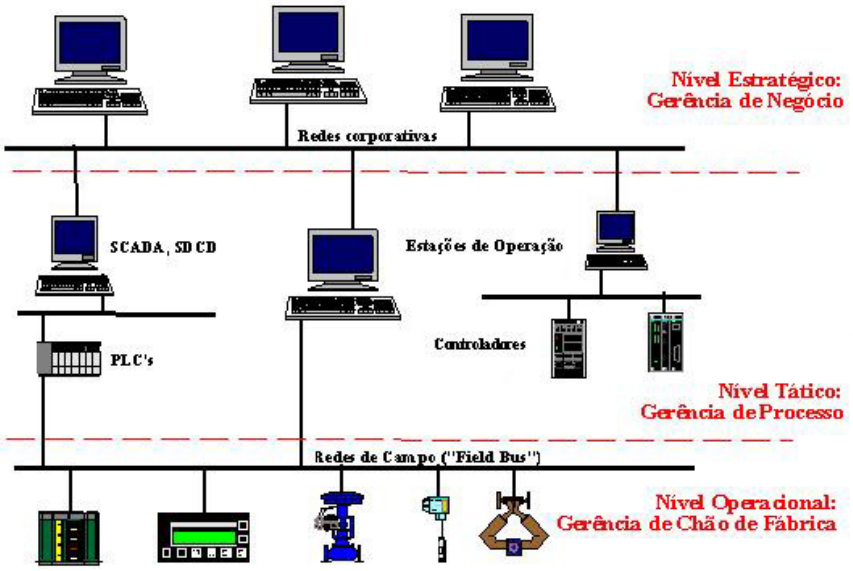

Figura 3.1: Níveis da informação no processo gerencial.

da empresa, administrativos e de aspectos financeiros. Os dados e informações podem ser utilizados por aplicações cliente de modo a otimizar a gerência e a integração de todo o processo de manufatura. Para realizar essa integração de modo efetivo, os fabricantes precisam ter acesso aos dados de chão-de-fábrica, dos processos industriais e integrá-los aos seus sistemas corporativos. Devem também poder utilizar as ferramentas disponíveis nos sistemas Scada, banco de dados utilizados, planilhas eletrônicas, etc., para atingir os objetivos da gerência integrada dos processos industriais.

A chave para a realização dessa integração é uma arquitetura aberta e efetiva de comunicação baseada no acesso a dados e não no tipo de dados.

A primeira especificação produzida pela OPC *Foundation* foi publicada em agosto de 1996 e chamada OPC *Specification Version* 1.0. O principal objetivo do grupo é atender às necessidades da indústria, através do aprimoramento e ampliação da especificação OPC. A estratégia adotada foi a criação de extensões à especificação existente, definição da adição de novas especificações e a realização de modificações para manter a compatibilidade máxima com as versões existentes. Em setembro de 1997 foi liberada a primeira atualização da especificação OPC que passou a ser chamada de OPC *Data Access Specification Version* 1.0*A* [Fonseca, 2002].

Atualmente existem as seguintes especificações publicadas ou em processo de

aprovação:

- OPC *Overview* - Descrição geral dos campos de aplicação das especificações OPC;
- OPC *Common Definitions and Interfaces* - Definição das funcionalidades básicas para as demais especificações;
- OPC *Data Access Specification* - Definição da interface para leitura e escrita de dados de tempo real;
- OPC *Alarms and Events Specification* - Definição da interface para monitoração de eventos;
- OPC *Historical Data Access Specification* - Definição da interface para acesso a dados históricos;
- OPC *Batch Specification* - Definição da interface para acesso aos dados de processos por batelada (*batch*). Esta especificação é uma extensão da OPC *Data Access Specification*;
- OPC *Security Specification* - Definição da interface para utilização de políticas de segurança;
- OPC *and XML* - Integração entre OPC e XML para aplicações via Internet.

Estas especificações têm a finalidade de orientar os desenvolvedores para a implementação das aplicações cliente e servidor.

A publicação das especificações para o padrão OPC possibilitou o desenvolvimento de diversos produtos para automação industrial, os quais se beneficiam das vantagens proporcionadas pelo padrão:

- Padronização das interfaces de comunicação entre os servidores e clientes de dados de tempo real, facilitando a integração e manutenção dos sistemas.
- Eliminação da necessidade de *drivers* de comunicação específicos (proprietários);

- Melhoria do desempenho e otimização da comunicação entre dispositivos de automação.
- Interoperabilidade entre sistemas de diversos fabricantes;
- Integração com sistemas MES, ERP e aplicações *Windows* (Excel, etc.);
- Redução dos custos e tempo para desenvolvimento de interfaces e *drivers* de comunicação, com conseqüente redução do custo de integração de sistemas.
- Facilidade de desenvolvimento e manutenção de sistemas e produtos para comunicação em tempo real, e
- Facilidade de treinamento.

Atualmente existem diversos produtos no mercado que utilizam o OPC para comunicação com dispositivos de chão de fábrica, de modo que o OPC está se tornando rapidamente o padrão de comunicação adotado pelo mercado de automação industrial e pela indústria.

3.1 Tecnologias OLE e DCOM

A tecnologia OLE - *Object Linking and Embedding* [Fonseca, 2002] foi desenvolvida pela Microsoft em meados de 1990, para suprir a necessidade de se integrar diferentes aplicações dentro da plataforma *Windows*, de forma a solucionar os problemas de desempenho e confiabilidade do até então utilizado padrão DDE - *Dynamic Data Exchange*.

Como uma continuação da tecnologia OLE, o DCOM - *Distribuited Component Object Model* surgiu junto com o sistema operacional *Windows NT* e foi logo aceito pela indústria. Basicamente, o DCOM é um conjunto de definições para permitir a implementação de aplicações distribuídas em uma arquitetura cliente-servidor. Desta forma, um cliente pode acessar diferentes servidores ao mesmo tempo e um servidor pode disponibilizar suas funcionalidades para diferentes clientes ao mesmo tempo.

Através da definição de interfaces, o DCOM permite que objetos sejam instanciados de forma distribuída e seus serviços e métodos (funções) sejam acessíveis

por diferentes programas. Para isso é necessário a utilização de uma linguagem especial, a IDL - *Interface Definition Language*. Isto significa que cada cliente pode chamar os métodos de qualquer objeto DCOM em um determinado servidor, independentemente do ambiente de programação (linguagem, compilador, versão, etc.) que os mesmos foram criados. Através de um identificador único (GUID - *Global Unique Identifier*), as interfaces são protegidas contra modificações após a sua publicação e a compatibilidade dos objetos DCOM é então garantida. Os objetos DCOM existem nos servidores DCOM. A forma de implementação dos servidores (DLL EXE, *InProcess* e *OutProcess*) determina como os objetos são carregados e gerenciados pelo servidor. Os objetos DCOM são acessíveis através de uma identificação CLSID - *Class Identifier* mantida pela Registry do sistema operacional. Através da CLSID os clientes podem lançar os componentes, solicitar as interfaces dos objetos e chamar os métodos desta interface. O ciclo de vida de um objeto DCOM é controlado pelo próprio componente, de forma que o mesmo se "auto-deleta" quando nenhum cliente está utilizando o mesmo, fazendo a liberação dos recursos do sistema.

3.2 A Arquitetura das Aplicações OPC

O padrão OPC implementa dois grandes módulos: OPC *Server* e OPC *Client* [Fonseca and Seixas-Filho, 2005]. Enquanto o OPC *Server* especifica interfaces padrão de acesso direto aos equipamentos ou aplicações, o OPC *Client* especifica a interface padrão para as aplicações terem acesso aos dados coletados.

Um exemplo dessa arquitetura de comunicação é mostrado na Figura 3.2.

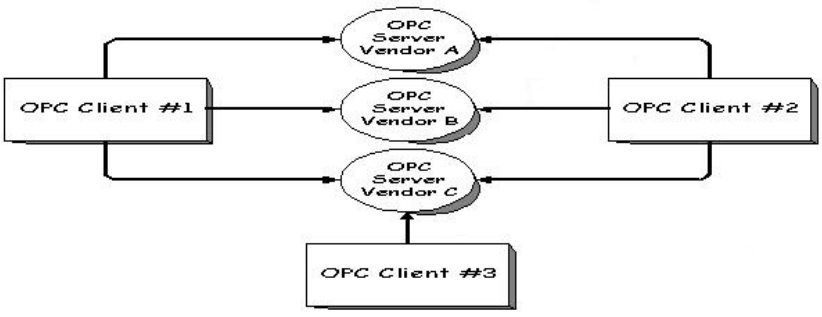

Figura 3.2: Estrutura Cliente-Servidor.

As aplicações são escritas atualmente em diversas linguagens, como Visual Basic, Delphi, Power Builder, etc. Os OPC *Servers* geralmente são escritos em C ou C++, aproveitando as características de encapsulamento dessas linguagens, fornecendo "objetos" que podem ser acessados por qualquer aplicação. Todas as especificações foram feitas de modo a facilitar o desenvolvimento dos OPC *Servers* e podem também ser escritos em outras linguagens. O acesso aos OPC *Servers* é realizado através dos componentes OLE/COM e OLE/DCOM, fornecidos pelo próprio sistema operacional *Windows*.

O desempenho da comunicação OPC se aproxima do desempenho apresentado por sistemas que utilizam *drivers* de comunicação específicos e otimizados. Normalmente, os *drivers* específicos possuem um ótimo desempenho após serem devidamente depurados e otimizados, o que via de regra não acontece em muitos casos. Como um servidor OPC nada mais é do que uma camada de *software* a mais para implementar as interfaces padrões e os mecanismos de comunicação com o cliente, é de se esperar que o desempenho do mesmo só seja afetado em relação à comunicação com o cliente e não com o dispositivo de campo.

No caso da comunicação com o dispositivo de campo, cada fornecedor pode implementar o *driver* e o protocolo que melhor se ajustem às necessidades do dispositivo e da rede de comunicação [Zheng and Nakagawa, 2002]. Desta forma, o desempenho do servidor OPC está mais relacionado à capacidade dos recursos de hardware da máquina que executa a aplicação do servidor do que propriamente do driver específico.

Como os recursos de hardware estão cada vez mais poderosos com respeito à capacidade de processamento, isto não tem se mostrado como um problema real. Entretanto, o que se tem verificado na prática é que muitos clientes e servidores OPC não implementam a comunicação de blocos de dados, fazendo a leitura de itens separadamente, o que ocasiona um grande *overhead* devido ao tratamento separado de *time stamp* e estado do dado para cada item OPC.

Outro ponto importante que muitos clientes OPC não implementam, consiste no agrupamento de dados que precisam ser lidos sob demanda, tais como janelas de operação de equipamentos, relatórios, etc. Os dados necessários para estes elementos (objetos) de monitoração, normalmente podem ser lidos sob demanda, de forma que somente quando o objeto estiver selecionado, será ativado o grupo OPC no servidor

para leitura dos dados. Quando o objeto não estiver selecionado, o grupo OPC ficará desativado, fazendo com que os dados não sejam lidos e melhorando o desempenho da comunicação.

Capítulo 4

Separação Cega de Fontes

O problema de separação cega de fontes (*Blind Source Separation - BSS*) tem por objetivo recuperar um conjunto de sinais fontes, independentes, a partir do conjunto de sinais observados. O termo cego indica que as fontes e a maneira como estas foram combinadas são desconhecidas [Amari, 1998]. Atualmente, o BSS tornou-se uma das principais áreas de pesquisa em processamento digital de sinais devido à sua aplicabilidade a diversos problemas reais. Os algoritmos baseados em Análise de Componentes Independentes (Independent Component Analysis - ICA) vêm fornecendo soluções efetivas aos problemas BSS [Amari et al., 1997] e sendo aplicados com sucesso em diversos campos, tais como: processamento de imagens, processamento de sinais biomédicos e sistemas de telecomunicação, entre outros [Cardoso, 1995].

4.1 Análise de Componentes Independentes

Para definir ICA, inicialmente se usará um modelo estatístico de "variáveis latentes". Suponha que foram observadas n misturas lineares $x_1, ..., x_n$ de n componentes independentes $s_1, ..., s_n$:

$$x_i = a_{i1} \cdot s_1 + a_i 2 \cdot s_2 + ... + a_i n \cdot s_n, \forall i \tag{4.1}$$

No modelo ICA, supõe-se que cada mistura x_i bem como cada componente independente s_i são variáveis aleatórias. Os valores observados $x_i(t)$, como por exemplo as medidas de sensores, são uma amostra destas variáveis aleatórias.

Sem perda de generalidade, assume-se que as variáveis observadas e as componentes independentes possuem média igual a zero [Haykin, 2001]. Se isso não for verdadeiro, subtrai-se do vetor de misturas a média e garante-se, assim, que ele possua média zero.

É conveniente usar a notação do vetor-matriz ao invés das somas como na Equação 4.1. Dessa forma, denota-se x pelo vetor aleatório cujos elementos são as misturas $x_1, ..., x_n$, e do mesmo modo s pelo vetor aleatório com elementos $s1, ..., sn$. Denota-se A, a matriz com elementos a_{ij}. Todos os vetores são compreendidos como vetores coluna. Assim, x^t, ou transposto de x, é um vetor linha. Usando esta notação de vetor-matriz, o modelo de mistura acima é escrito como:

$$x = A \cdot s \tag{4.2}$$

As componentes independentes, que são variáveis latentes, não podendo desta forma serem diretamente observadas, e a matriz de mistura A são desconhecidas. O único ítem conhecido é o vetor observado x, a partir do qual se deve estimar A e s.

Uma vez descrito o modelo geral que descreve como os dados observados são gerados a partir de um processo de mistura das componentes s_i, o objetivo da técnica ICA é encontrar uma matriz W de separação, ou seja, que realiza o processo inverso da matriz A, tal que o vetor de componentes s possa ser recuperado a partir dos dados observados. Ou seja:

$$s = W \cdot x \tag{4.3}$$

Observe na Figura 4.1 um diagrama de blocos onde se tem uma visão geral de todo o processo de Análise de Componentes Independentes explicado anteriormente.

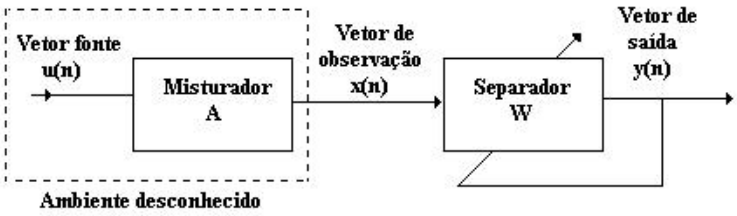

Figura 4.1: Diagrama de blocos da técnica ICA.

O vetor de saída y no diagrama acima representa o sinal recuperado que se aproxima o máximo possível do sinal original s.

O ICA possui duas restrições [Hywarinen and Oja, 1999] que devem ser levadas em consideração ou, caso contrário, podem comprometer seu funcionamento.

A primeira delas é que as componentes s_i devem ser estatisticamente independentes, ou seja, a informação sobre o valor de uma componente não depende da informação sobre o valor de outra componente. A não-correlacionalidade não é suficiente para separar as componentes. No caso de duas variáveis x_1 e x_2 serem estatisticamente independentes, implica que sua correlação será igual a zero, mas o contrário nem sempre é verdadeiro. Por isso que algumas técnicas, como o PCA (*Principal Component of Analysis*) que garantem apenas a não-correlacionalidade, nem sempre são capazes de separar os sinais.

A segunda diz que as componentes independentes não devem ser gaussianas uma vez que as variáveis observadas x também serão gaussianas. A distribuição de densidade de variáveis gaussianas é simétrica não contendo nenhuma informação sobre a direção das colunas da matriz A e não podendo assim ser estimada. Esses conceitos serão detalhados mais a seguir.

Observa-se, também, que o modelo ICA exposto pela Equação 4.3 possui ambigüidades. A primeira delas é que as variâncias (energias) das componentes independentes não podem ser determinadas, uma vez que sendo s e A desconhecidos, qualquer escalar multiplicado por uma das fontes s_i poderia sempre ser cancelado dividindo a coluna correspondente a_i de A pelo mesmo escalar. Então, sugere-se que, devido as componentes independentes serem variáveis aleatórias, a maneira mais fácil de se resolver este problema é supor que cada uma das componentes tem variância unitária: Assim, a matriz A estará adaptada ao método ICA e esta ambigüidade contornada. Vale ressaltar que a solução anterior não contorna completamente a ambigüidade já que ao multiplicar uma componente independente por -1 não afeta o modelo. Mas essa exceção é, felizmente, insignificante na maioria das aplicações.

Outra ambigüidade que encontramos no modelo é que não se pode determinar a ordem das componentes. A razão disso é que, outra vez ambos s e A sendo desconhecidos, pode-se livremente mudar a ordem dos termos na soma em 4.1, e chamar algumas das componentes independentes primeiro. Formalmente, uma

matriz de permutação P e sua inversa podem ser substituídas no modelo de modo que $x = AP^{-1}Ps$. Os elementos de $\boldsymbol{Ps}$ são as variáveis independentes originais s_i, mas em uma outra ordem e a matriz AP^{-1} é uma nova matriz de mistura desconhecida.

Para ilustrar o modelo de ICA em termos estatísticos será utilizado um exemplo proposto em [Hywarinen and Oja, 1999]. Considere duas componentes independentes, s_1 e s_2, que têm as seguintes distribuições uniformes:

$$\rho(s_i) = \begin{cases} \frac{1}{2\sqrt{3}} \text{ se } |s_i| \leqq \sqrt{3} \\ 0 \text{ caso contrario} \end{cases} \tag{4.4}$$

A escala dos valores para esta distribuição uniforme foi escolhida de forma que a média é zero e a variância igual a um, como discutido anteriormente. A densidade comum de s_1 e de s_2 é então uniforme e disposta em um quadrado. Isto provém da definição básica que a densidade comum de duas variáveis independentes é justamente o produto de suas densidades marginais, necessitando simplesmente computar o produto. A densidade comum é ilustrada na Figura 4.2 mostrando os pontos de dados extraídos aleatoriamente desta distribuição.

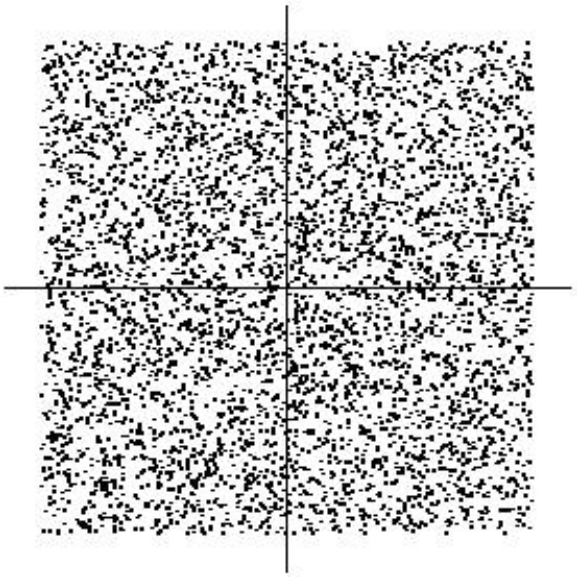

Figura 4.2: Distribuição comum das componentes independentes s_1 e s_2 com distribuições uniformes. A linha central horizontal representa s_1 e a linha central vertical s_2.

Com a matriz de mistura abaixo é realizada a mistura das duas componentes, resultando nas variáveis misturadas x_1 e x_2:

$$A = \begin{bmatrix} -0.815345 & -0.91101 \\ 0.452653 & -0.43908 \end{bmatrix} \tag{4.5}$$

Observa-se que os dados misturados têm uma distribuição uniforme em um paralelogramo, como mostrado na Figura 4.3. As variáveis aleatórias x_1 e x_2 não são mais independentes o que é provado claramente quando partindo dos valores máximos ou mínimos de x_1 é possível determinar completamente o valor de x_2, sendo assim conseqüentemente não independentes.

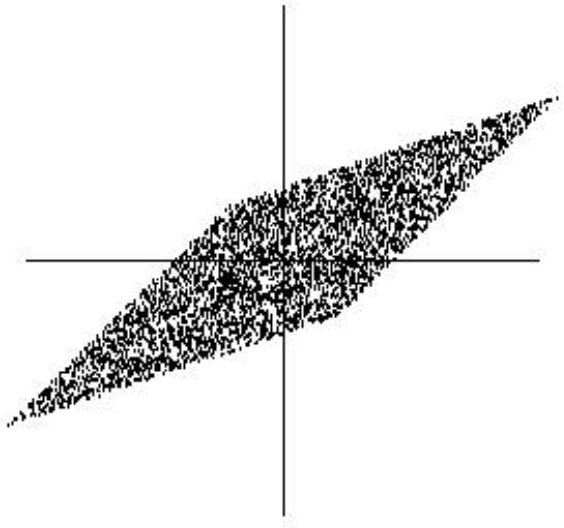

Figura 4.3: Distribuição comum das misturas observadas x_1 e x_2. A linha central horizontal representa x_1 e a linha central vertical x_2.

O problema de estimar o modelo de dados do ICA é então estimar a matriz de mistura A usando somente as informações contidas nas misturas x_1 e x_2. Analisando-se a Figura 4.3 pode-se deduzir uma forma intuitiva para estimar a matriz: as bordas do paralelogramo estão nos sentidos das colunas de A. Isto significa que, a princípio, é possível estimar o modelo ICA primeiramente estimando a densidade comum de x_1 e de x_2, e depois encontrando as bordas. Assim, o problema parece ter uma solução.

Mas, na verdade, este seria um método que traria resultados insatisfatórios porque trabalha apenas com variáveis que possuem distribuições uniformes. Além disso, seria computacionalmente custoso, quando o que se busca é um método que trabalhe de forma rápida e confiável para todas as distribuições das componentes independentes.

4.1.1 Conceitos e Propriedades

Para definir o conceito da independência estatística, considere duas variáveis aleatórias y_1 e y_2. Basicamente, as variáveis y_1 e y_2 seriam independentes se a informação do valor de y_1 não apresentar nenhuma informação no valor de y_2, e vice-versa. Este é o caso das variáveis s_1 e s_2 na Figura 4.2, mas não se aplica às variáveis da mistura x_1 e x_2 da Figura 4.3, como dito antes.

Tecnicamente, a independência pode ser definida pelas densidades da probabilidade. Denota-se por $\rho(y_1, y_2)$ a função comum de densidade da probabilidade (fdp) de y_1 e de y_2. E ainda por $\rho_1(y_1)$ a fdp marginal de y_1:

$$\rho_1(y_1) = \int \rho(y_1, y_2) dy_2 \tag{4.6}$$

e similarmente para y_2. Então defini-se que y_1 e y_2 são independentes se e somente se a fdp da junção se dá da seguinte maneira:

$$\rho_1(y_1, y_2) = \rho(y_1) \cdot \rho(y_2) \tag{4.7}$$

Esta definição estende-se naturalmente para todas as n variáveis aleatórias.

A definição pode ser usada para derivar a propriedade mais importante de variáveis aleatórias independentes. Dado duas funções, h_1 e h_2, tem-se sempre:

$$E\{h_1(y_1)h_2(y_2)\} = E\{h_1(y_1)\} \cdot E\{h_2(y_2)\} \tag{4.8}$$

Uma forma mais fraca da independência é a não-correlação. Duas variáveis aleatórias y_1 e y_2 seriam não-correlacionadas, se sua covariância fosse zero:

$$E\{y_1 y_2\} - E\{y_1\} \cdot E\{y_2\} = 0 \tag{4.9}$$

Se as variáveis forem independentes também são não-correlacionadas, é o que se segue diretamente da Equação 4.8, tomando $h_1(y_1) = y_1$ e $h_2(y_2) = y_2$.

Por outro lado, o não-correlacionamento não implica a independência. Como exemplo, suponha que (y_1, y_2) são valores discretos e seguem tal distribuição que o par ordenado possui probabilidade $\frac{1}{4}$, igual para qualquer um dos seguintes valores:

$(0,1)$,$(0,-1)$,$(1,0)$,$(-1,0)$. Então, y_1 e y_2 são não-correlacionados, como pode simplesmente ser calculado. Mas analisando a Equação 4.10, tem-se:

$$E\{y_1^2 y_2^2\} = 0 \neq \frac{1}{4} = E\{y_1^2\} \cdot E\{y_2^2\} \tag{4.10}$$

Nota-se que a condição da Equação 4.9 foi violada, e as variáveis não podem ser independentes.

A limitação fundamental em ICA é que as componentes independentes devem ser não-gaussianas para que a técnica seja possível.

Para se verificar porque as variáveis gaussianas tornam a aplicação do ICA impossível, suponha que a matriz de mistura seja ortogonal e s_i gaussiana. Então, x_1 e x_2 são gaussianas, não-correlacionadas, e de variância igual a 1. Sua densidade comum é dada aproximadamente por:

$$\rho(x_1, x_2) = \frac{1}{2\pi} \exp(-\frac{x_1^2 + x_2^2}{2}) \tag{4.11}$$

Esta distribuição é ilustrada na Figura 4.4. A figura mostra que a densidade é completamente simétrica. Conseqüentemente, não contém nenhuma informação dos sentidos das colunas da matriz de mistura A e assim ela não pode ser estimada.

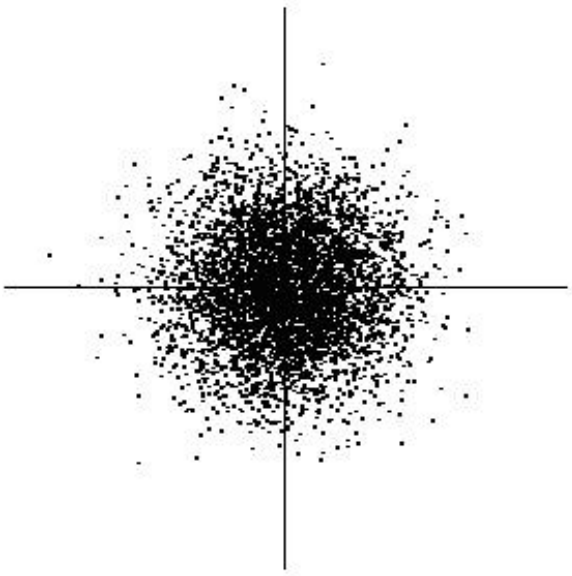

Figura 4.4: A distribuição multi-variável de duas variáveis gaussianas independentes.

Mais rigorosamente, pode-se provar que a distribuição de qualquer transformação ortogonal de gaussianas (x_1,x_2) tem exatamente a mesma distribuição que (x_1,x_2), e que x_1 e x_2 são independentes. Assim, no exemplo de variáveis gaussianas, nós podemos somente estimar o modelo ICA para uma transformação ortogonal. Ou seja, a matriz A não é identificada para componentes independentes gaussianas. Mas, nas pesquisas mais recentes foi comprovado que se apenas uma das componentes independentes for gaussiana, o modelo ICA pode ainda ser estimado.

Os algoritmos ICA são baseados na medida de não-gaussianidade das componentes independentes e, além de descorrelacionar os sinais (estatística de segunda ordem), também reduz a dependência de estatísticas de ordens superiores.

Atualmente, sem a não-gaussianidade a estimação de ICA não seria possível em qualquer aplicação, como mencionado anteriormente. Talvez este seja o motivo do ressurgir atrasado de pesquisas sobre ICA uma vez que na maioria das teorias estatísticas clássicas, assume-se que variáveis aleatórias possuem distribuições gaussianas, impossibilitando assim todos os métodos relacionados a ICA.

O teorema do limite central [Hywarinen and Oja, 1999], um resultado clássico na teoria da probabilidade, diz que a distribuição de uma soma de variáveis aleatórias independentes tende para uma distribuição gaussiana, sob determinadas circunstâncias. Assim, uma soma de duas variáveis aleatórias independentes tem geralmente uma distribuição mais próxima da gaussiana do que qualquer uma das duas variáveis aleatórias originais.

Para usar a não-gaussianidade na estimação de ICA, nós devemos ter uma medida quantitativa da não-gaussianidade de uma variável aleatória y. Para simplificar as coisas, deixe-nos supor que y possui média zero e variância igual a um.

4.1.2 Medidas de Não-gaussianidade

Algumas das medidas mais comuns de não-gaussianidade são a Kurtosis, Negentropia, Informação Mútua e Maximização de Verossimilhança. A seguir serão apresentadas algumas dessas medidas.

Kurtosis

A medida clássica de não-gaussianidade é a kurtosis ou cumulante de quarta ordem. A kurtosis de y é definida classicamente por:

$$kurt(y) = E\{y^4\} - 3(E\{y^2\})^2 \tag{4.12}$$

Como foi suposto que y possui variância unitária, então o lado direito da Equação 4.12 pode ser simplificado por $E\{y^4\} - 3$. Isto mostra que a kurtosis é simplesmente uma versão normalizada do momento de quarta ordem, $E\{y^4\}$.

Para um y gaussiano, os momentos de quarta ordem são $3(E\{y^2\})^2$. Assim, a kurtosis é zero para uma variável aleatória gaussiana. Para outras variáveis aleatórias não-gassianas, a kurtosis é não nula [Haykin, 2001].

A kurtosis pode ser positiva ou negativa. As variáveis aleatórias que têm kurtosis negativa são chamadas subgaussianas, e aquelas com kurtosis positiva são chamadas supergaussianas. As variáveis aleatórias supergaussianas têm uma fdp "spiky" com caudas pesadas, isto é, a fdp é relativamente grande em zero e em valores mais altos da variável, e é pequena para valores intermediários.

A não-gaussianidade é medida tipicamente pelo valor absoluto da kurtosis. O quadrado da kurtosis pode também ser usado. Estes valores são zero para uma variável gaussiana, e diferente de zero para a maioria de variáveis aleatórias não-gaussianas, como dito antes, mas há as variáveis aleatórias não-gaussianas que têm a kurtosis zero, só que estes casos são muito raros.

A kurtosis tem sido usada amplamente como uma medida de não-gaussianidade em ICA e nos campos relacionados. A razão principal é sua simplicidade computacional e teórica. Computacionalmente, a kurtosis pode ser estimada simplesmente usando o momento de quarta ordem das amostras de dados. A análise teórica é simplificada por causa da seguinte propriedade de linearidade: Se x_1 e x_2 forem duas variáveis aleatórias independentes, temos:

$$kurt(x_1 + x_2) = kurt(x_1) + kurt(x_2) \tag{4.13}$$

e

$$kurt(\alpha \cdot x_1) = \alpha^4 \cdot kurt(x_1) \tag{4.14}$$

onde α é um escalar. Estas propriedades são facilmente provadas usando a definição de kurtosis, mostrada anteriormente.

Na prática a aplicação dessa medida seria feita da seguinte forma: partindo de alguma matriz de peso W, computa-se o sentido em que a kurtosis de $y = Wx$ está crescendo mais fortemente, kurtosis positiva, ou diminuindo mais fortemente, kurtosis negativa, baseando-se nas amostras disponíveis $x(1), ..., x(T)$ do vetor da mistura x, e usando o método do gradiente ou alguma de suas extensões para encontrar uma nova matriz W.

Entretanto, a kurtosis tem também alguns inconvenientes na prática, quando seu valor tem que ser estimado de uma amostra observada. O problema principal é que a kurtosis pode ser muito sensível aos *outliers*.

Designa-se, habitualmente, por *outliers* as observações que apresentam um grande afastamento das restantes ou são inconsistentes com elas. No caso da kurtosis, seu valor pode depender somente de algumas observações nas caudas da distribuição, que podem ser observações errôneas ou irrelevantes. Conclui-se, então, que a kurtosis não é uma medida robusta da não-gaussianidade não sendo assim muito recomendada para o uso nos algoritmos ICA.

Negentropia

Uma segunda medida muito importante para determinar a não-gaussianidade é a negentropia, a qual é baseada na quantidade de informação teórica de uma variável dada pela entropia diferencial.

A entropia de uma variável aleatória observada pode ser interpretada como o grau de informação que essa variável carrega. Quanto mais aleatória, ou seja, imprevisível e desestruturada a variável, maior a sua entropia. Para uma variável Y, aleatória e discreta, a entropia é definida como:

$$H(Y) = -\sum_i P(Y = a_i) \cdot \log P(Y = a_i) \tag{4.15}$$

onde, a_i é o possível valor Y. Esta definição muito bem conhecida pode ser generalizada para variáveis aleatórias de valores contínuos e vetores, que no caso é chamada de entropia diferencial. A entropia diferencial H de um vetor aleatório y com densidade $f(y)$ é definida por:

$$H(Y) = -\int f(y) \cdot \log f(y) \tag{4.16}$$

Um resultado fundamental da teoria de informação é que uma variável gaussiana tem a maior entropia entre todas as variáveis aleatórias com a mesma variância. Isto significa que a entropia poderia ser usada como uma medida da não-gaussianidade. Em fato, isto mostra que a distribuição gaussiana é a mais aleatória e menos estruturada de todas as distribuições. A entropia é pequena para as distribuições que são concentradas em determinados valores, isto é, quando a variável é aglomerada, ou tem uma fdp que seja muito "spiky", isto é, com picos.

Para obter uma medida da não-gaussianidade, que é zero para uma variável gaussiana e sempre não negativa, usa-se freqüentemente uma versão modificada da definição da entropia diferencial, chamada negentropia. Observe abaixo:

$$J(y) = H(y_{gauss}) - H(y) \tag{4.17}$$

Na Equação 4.17, y_{gauss} é uma variável aleatória gaussiana com uma mesma matriz de covariância que y. Devido às propriedades acima mencionadas, negentropia é sempre não negativa, e é zero se e somente se y tem uma distribuição gaussiana.

A vantagem de usar negentropia, ou outro método equivalente à entropia diferencial, como uma medida de não-gaussianidade é que está bem justificada pela teoria estatística. Negentropia é um estimator ótimo da não-gaussianidade, tanto quanto as propriedades estatísticas. O problema em usar negentropia é que esta medida é computacionalmente árdua. Estimar negentropia usando a definição requereria uma estimativa (possivelmente não paramétrica) da fdp. Conseqüentemente, aproximações mais simples da negentropia são muito úteis.

Uma aproximação é usada para aproximar negentropia de maneira que seja computacionalmente mais simples do que o cálculo direto da negentropia, mas ainda

mais estável do que a kurtosis. A seguinte equação aproxima negentropia:

$$J(x) \approx \sum_{i-1}^{\rho} k_i[mean(G_i(x)) - mean(G_i(\nu))]^2 \tag{4.18}$$

Nesta equação, $G(x)$ é alguma função não-quadrática, v é uma variável gaussiana com a variância 1 e média zero, e k_i é um valor constante. Há as funções que podem ser usadas para G que dá uma aproximação boa da negentropia e são menos sensíveis aos "outliers" do que a kurtosis. Geralmente, escolhe-se um G que não cresce demasiadamente rápido, obtendo assim um estimador mais robusto. As seguintes escolhas de G provaram ser muito úteis:

$$G_1(u) = \frac{1}{a_1} \cdot \log\cosh(a_1 u) \tag{4.19}$$

$$G_2(u) = -\exp(-\frac{u^2}{2}) \tag{4.20}$$

onde $1 \leq a_1 \leq 2$, é uma constante.

Assim, essa aproximação da negentropia gera um acordo entre as propriedades das duas medidas clássicas de não-gaussianidade dadas pela kurtosis e negentropia. É conceitualmente simples, computação rápida e robusta.

4.1.3 Pré-processamento para ICA

Antes de aplicar um algoritmo ICA nos dados, é geralmente muito útil fazer algum pré-processamento [Hywarinen and Oja, 1999], que fazem o problema da estimação de ICA tornar-se mais simples e o condicionarem melhor.

Centralização

O pré-processamento mais básico e o mais necessário é o que deve centralizar x, isto é, subtrair seu vetor médio $m = E\{x\}$ para fazer com que x seja uma variável de média zero. Isto implica que s também possui média zero.

Este pré-processamento é feito unicamente para simplificar os algoritmos de ICA, não significa que a média não possa ser estimada. Após ter estimado a matriz de

mistura A com dados centralizados, nós podemos terminar a estimação adicionando o vetor médio novamente ao conjunto de dados centralizados de s. O vetor médio de s é dado aproximadamente por $A^{-1}m$, onde m é a média que foi subtraída no pré-processamento.

Branqueamento

Uma outra estratégia útil do pré-processamento em ICA é o branqueamento das variáveis observadas. Isto significa que antes da aplicação do algoritmo ICA, e após a centralização, o vetor observado x_{linear} é transformado de modo que seja obtido um vetor novo x' que seja branco, onde suas componentes são descorrelacionadas e sua variância igual a 1.

Uma ilustração gráfica do efeito de branqueamento pode ser vista na Figura 4.5, em que os dados da Figura 4.2 foram branqueados. O quadrado que define a distribuição é agora uma versão rotacionada do quadrado original da Figura 4.2. O resultado é a estimação de um único ângulo que dá a rotação.

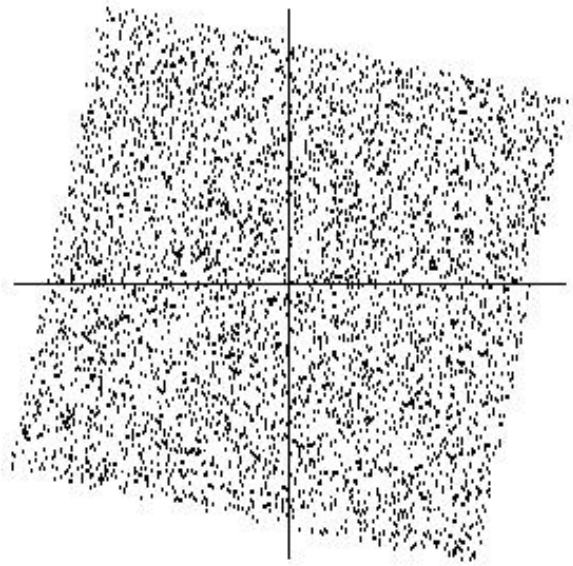

Figura 4.5: Distribuição comum das misturas branqueadas.

4.2 Algoritmo FastICA

O processo descrito na seção anterior, o ICA, pode ser interpretado como um modelo neural [Karhunen, 1996] onde o vetor x é a camada de entrada da rede neural e o vetor y a camada de saída da mesma. A matriz W é a matriz de pesos sinápticos. Observe a arquitetura da Figura 4.6:

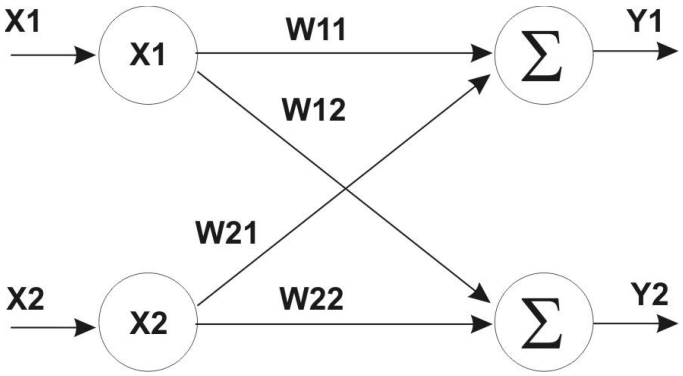

Figura 4.6: Modelo Neural do ICA.

- Número de neurônios na camada de entrada: 2;
- Número de neurônios na camada de saída: 2;

Essa modelagem neural torna possível a implementação do processo através de um algoritmo neural. O FastICA [Hywarinen and Oja, 1999] foi o algoritmo escolhido para a implementação do trabalho.

Inicialmente será mostrada a versão do algoritmo para apenas uma unidade. Entende-se por "unidade" uma unidade computacional, eventualmente um neurônio artificial, tendo um vetor de peso w que o neurônio atualiza através de uma regra de aprendizagem. A regra de aprendizagem do FastICA encontra um sentido, isto é, um vetor w tal que $w^T x$ maximize a não-gaussianidade. Esta, neste caso, é medida pela aproximação da negentropia, já estudada anteriormente.

A regra de aprendizagem do FastICA, é descrita a seguir:

1. Obtém-se, aleatoriamente, um vetor W inicial;
2. $\mathbf{W}^+ = E\{\mathbf{X} \cdot G(\mathbf{W}^t \cdot \mathbf{X})\} - E\{G'(\mathbf{W}^t \cdot \mathbf{X})\} \cdot \mathbf{W}$;

3. $\mathbf{W} = \mathbf{W}^{+}/||\mathbf{W}^{+}||$;

4. Repetir passo 2 até que haja convergência.

A função não quadrática utilizada no algoritmo e sua derivada são, respectivamente:

$$G(u) = 1/a_1 \cdot \log\cosh(a_{1u}) \tag{4.21}$$

$$G'(u) = \tanh(a_{1u}) \tag{4.22}$$

Como a constante a_1 é definida no intervalo de 1 à 2, fixou-se seu valor em 1. Antes de inicializar o treinamento do FastICA deve ser feito o pré-processamento que consiste na aplicação das técnicas de centralização e branqueamento.

O algoritmo acima estima apenas uma das componentes independentes. Para estimar diversas componentes independentes é necessário aplicar o algoritmo de uma unidade às diversas unidades, neurônios, com vetores de peso $w_1, ..., w_n$.

Para evitar que vetores diferentes convirjam para os mesmos máximos, deve-se descorrelacionar as saídas $w_1^T x, ..., w_n^T x$ após cada iteração. Existem vários métodos que realizam essa descorrelação. Uma alternativa mais simples é o seguinte algoritmo iterativo:

1. $W = W/\sqrt{\|W \cdot W^T\|}$;

2. $W = \frac{3}{2}W - \frac{1}{2}WW^TW$;

3. Repetir passo 2 até que haja convergência.

4.2.1 Propriedades do Algoritmo de FastICA

O algoritmo FastICA possui propriedades vantajosas quando comparado com outros métodos existentes para ICA.

- A convergência é cúbica (ou pelo menos quadrática), sob a suposição do modelo dos dados de ICA. Isto contrasta com os algoritmos ICA baseados nos

métodos (estocásticos) da descida do gradiente, onde a convergência é somente linear. Isto significa uma convergência muito rápida, como foi confirmado por simulações e por experiências em dados reais.

- O algoritmo é fácil de usar uma vez que ao contrário dos algoritmos baseados no gradiente, não há uma etapa para escolha do tamanho dos parâmetros.

- O algoritmo encontra diretamente componentes independentes (praticamente) de qualquer distribuição não-gaussiana usando qualquer g não-linear. Isto contrasta com muitos algoritmos, onde uma estimativa da função de distribuição de probabilidade tem que estar primeiramente disponível, e a não-linearidade deve ser escolhida conformemente.

- O desempenho do método pode ser otimizado escolhendo uma função g não-linear apropriada. Particularmente, ela pode obter algoritmos que são robustos e/ou de variância mínima.

Capítulo 5

Implementação de Filtragem Inteligente em Rede FF

O objetivo do trabalho é o desenvolvimento e incorporação da técnica inteligente de filtragem de ruídos, baseada no método de Análise de Componentes Independentes e representada pelo algoritmo FastICA, ao ambiente de rede industrial *Foundation Fieldbus* com o intuito de remover possíveis ruídos adicionados à informação dos sensores de campo e possibilitar melhoras significativas à estrutura atual.

Há várias formas de incorporar novas funcionalidades em um ambiente *Foundation Fieldbus*. A forma mais convencional seria a incorporação das funcionalidades em um sistema supervisório, de forma que o processamento ocorreria nas estações de supervisão de um processo.

Uma outra possibilidade de implementação consiste na incorporação de hardware à rede para realizar o processamento dos sinais analógicos provenientes dos sensores. Desta forma, a nova funcionalidade seria inteiramente transparente à arquitetura *Foundation Fieldbus*.

Por outro lado, uma rede *Fieldbus* é capaz de executar estratégias de controle nos próprios instrumentos de campo. Assim sendo, a maneira mais elegante e que traria mais flexibilidade seria a implementação da nova funcionalidade como blocos funcionais, que seriam executados pelos próprios equipamentos FF, visando uma forma robusta de filtragem de ruído e mantendo o padrão de interoperabilidade do sistema.

A seguir serão detalhadas cada uma das possibilidades de implementação propostas no trabalho.

5.1 Implementação do Algoritmo de Filtragem Embarcado em DSP

Esta abordagem propõe a implementação do algoritmo FastICA como um dispositivo de software embarcado, que possa ser adicionado ao dispositivo de campo, tendo como núcleo um processador digital de sinais.

Os dispositivos DSP (*Digital Signal Processors*) podem ser considerados como sendo uma espécie de microcontroladores adaptados para aplicações de processamento Digital de Sinais, no sentido em que possuem uma arquitetura otimizada para aplicações que requerem uma computação intensiva, além de periféricos com funções especializadas integrados ao *chip*. Há, no mercado, um grande número de fabricantes destes dispositivos, cabendo ao usuário escolher o melhor modelo, baseado nas características da sua aplicação.

O módulo de desenvolvimento utilizado neste trabalho foi o TMS320lf2407 EVM da *SPECTRUM DIGITAL*, que incorpora como núcleo central de processamento o DSP, de 16 bits, TMS320LF2407 da Texas Instruments Inc. Dentre as principais características desta família de DSP's destacam-se: freqüência máxima de operação de 40 Mhz, memória *SRAM* de 128 *KWords*, memória Flash EEPROM, conversor A/D com resolução de 10 bits e 16 canais de entrada, e capacidade de executar conversões em um tempo aproximado de 500 ns.

Há ainda, aproximadamente 40 canais digitais, de propósito geral, cada um podendo ser programável para executar sua função primária, ou comportar-se como um canal digital normal (neste caso, o canal pode ser programado para operar como entrada ou saída).

Todos estes módulos, bem como suas funções, podem ser controlados através da configuração dos registradores correspondentes, e é basicamente disto que consiste a programação deste DSP.

Algumas funções já implementadas no módulo de desenvolvimento facilitam a implementação do algoritmo e a avaliação nos testes da filtragem. Dentre estas

funções destacam-se principalmente o conversor D/A (digital/analógico) de 16 bits e quatro saídas analógicas. Toda a programação foi feita em linguagens C e Assembler, utilizando o ambiente de desenvolvimento *Code Composer*, fornecido pela *Texas Instruments Inc.*

A arquitetura proposta, como pode ser vista no diagrama apresentado na Figura 5.1, se constitui na inclusão do módulo de desenvolvimento TMS320lf2407 EVM aos instrumentos de campo ligados através do protocolo de transmissão de dados FF.

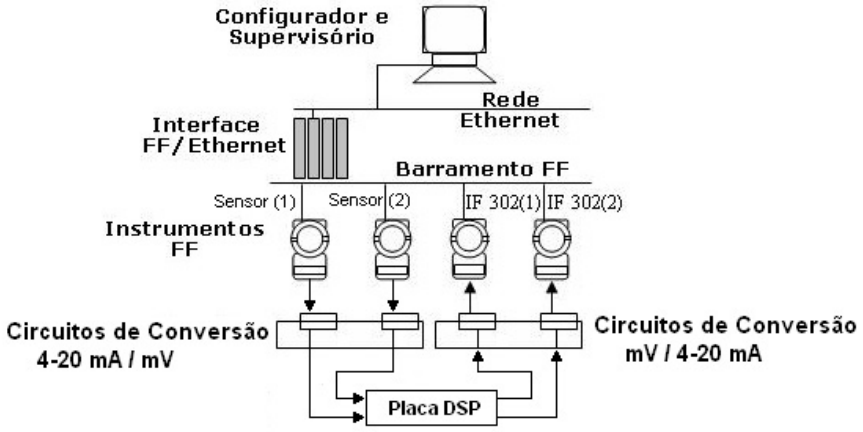

Figura 5.1: Arquitetura do Sistema.

A inserção de hardware dedicado no ambiente FF não é trivial, existindo a necessidade de conformação bidirecional de sinais de tensão para níveis de 4 a 20mA, além da comunicação entre o hardware e o protocolo de rede, realizada através de dispositivos conversores corrente/*fieldbus*.

O módulo é conectado aos sensores de campo *Fieldbus* através de uma interface conversora que mapeia os níveis de corrente 4 a 20mA, provenientes dos sensores, em níveis de tensão. Essa interface é implementada utilizando o *chip* RCV 420 da *Texas Instruments INC*, como mostrado na Figura 5.2.

Para que os sinais dos sensores possam ser filtrados pelo algoritmo FastICA embarcado no DSP, é realizada a digitalização dos mesmos a partir de uma rotina de conversão A/D (analógico/digital).

O processo de filtragem é realizado após o treinamento *off-line* do algoritmo FastICA onde é obtida a matriz de pesos sinápticos. Essa matriz é aplicada aos

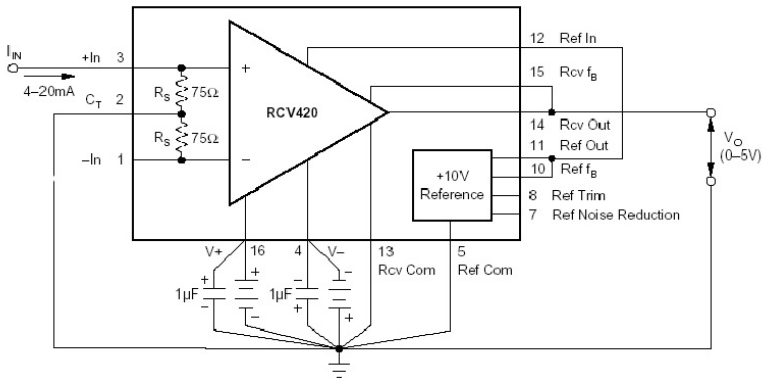

Figura 5.2: Conversor corrente/tensão.

sinais digitalizados dos sensores, eliminando o ruído e completando a filtragem.

Os sinais, resultantes desse processamento, após passarem por uma interface D/A (digital/analógico) e tomarem a forma de tensão, são convertidos em níveis de corrente 4 a 20mA e devolvidas para o meio *Fieldbus* pelo instrumento IF-302.

O módulo é conectado à rede *Fieldbus* através de uma interface capaz de converter níveis de tensão em níveis de corrente 4 a 20mA, implementada com o *chip* XTR 115U da *Texas Instruments INC*, como mostrado na Figura 5.3, e instrumentos de campo IF-302 que convertem níveis de corrente 4 a 20mA em sinais *Foundation Fieldbus*.

As estações de trabalho, no topo do esquema da Figura 5.1, estão configuradas para fazer a leitura das informações injetadas na rede.

5.2 Implementação do Algoritmo de Filtragem por Meio de Blocos Funcionais FF

O algoritmo FastICA possui natureza neural podendo ser representado como uma rede neural onde, são necessários vários neurônios artificiais interligados para formar a arquitetura desejada. Dentre os blocos funcionais (BF) padronizados, os mais importantes para a implementação do algoritmo em questão, são os blocos AI (*Analog Input*) e ARITH (aritmético). Através da configuração e interligação desses dois blocos funcionais, pode-se representar o algoritmo.

Os blocos AI fornecem um estado correspondente ao valor dos sinais provenientes

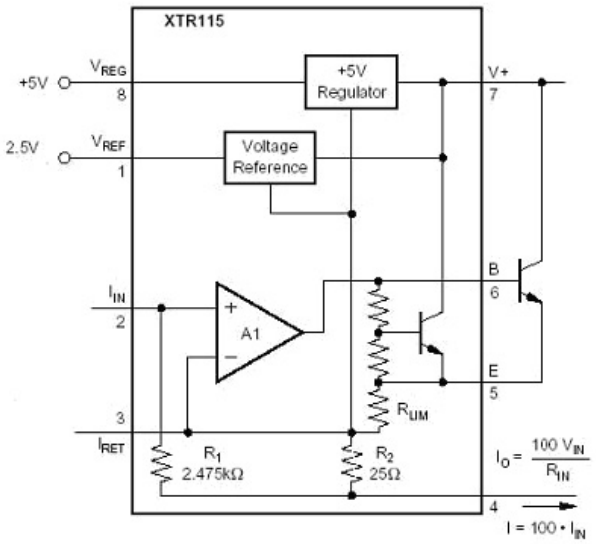

Figura 5.3: Conversor tensão/corrente.

dos sensores, contendo uma informação sobre a sua confiabilidade. Adicionalmente, através da configuração do bloco AI, o usuário pode limitar a sua saída para uma faixa de valores considerada segura, filtrando valores fora desta faixa e colocando-os em escala conveniente para o processamento pelo bloco seguinte. A Figura 5.4 mostra o esquema geral do bloco AI e os seus principais parâmetros configuráveis.

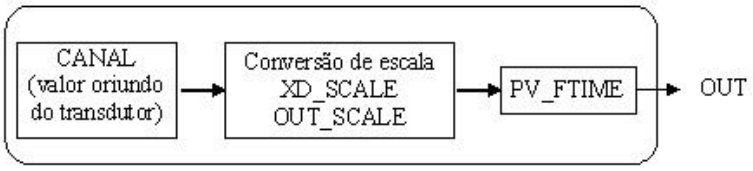

Figura 5.4: Bloco AI.

O Bloco ARITH pode facilmente implementar um neurônio artificial cuja função de ativação seja linear. Este objeto FF possui três entradas e uma saída. Além disso, existe também um sub-bloco rotulado de "*Algorithm Type*" que pode assumir diversos comportamentos, executando várias funções matemáticas, tais como a média aritmética, raiz quadrada do produto das entradas, o quociente entre duas delas ou outras funções mais específicas do ambiente de controle. Neste trabalho,

escolheu-se a soma tradicional, que efetua a soma do produto das entradas por coeficientes configurados pelo usuário. Os coeficientes exercem o papel dos pesos sinápticos do neurônio. A Figura 5.5 mostra os principais detalhes do bloco aritmético usados nesta aplicação.

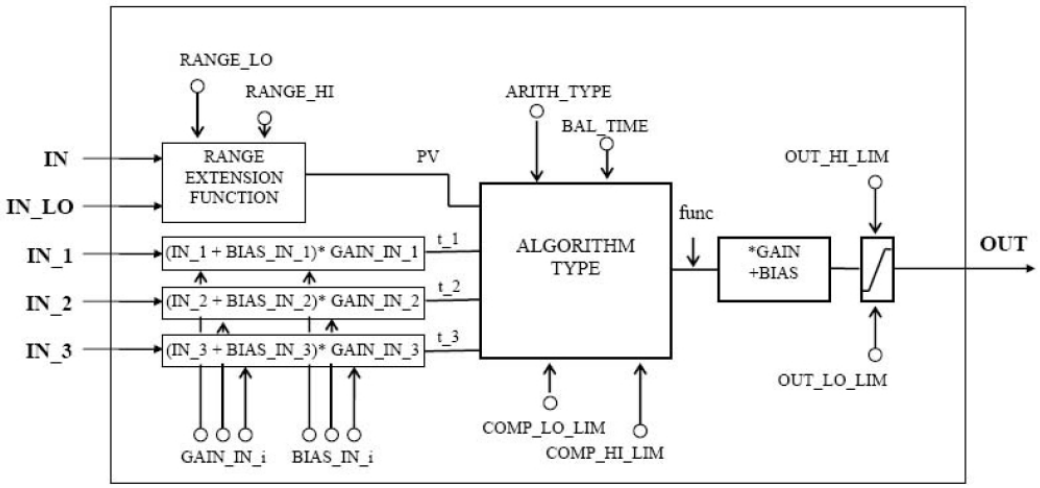

Figura 5.5: Bloco ARITH.

Depois da configuração desses dois blocos funcionais e a interligação entre suas entradas e saídas, a estrutura estará pronta para receber os valores dos pesos sinápticos, obtidos do treinamento *off-line* do algoritmo FastICA, representando assim, o processo de filtragem.

5.2.1 Treinamento *On-line* do Algoritmo

Ambas as soluções, tratadas anteriormente, apresentam um treinamento *off-line* do algoritmo. Um treinamento *on-line* do FastICA também é proposto de forma a tornar o processo mais inteligente e impedir que os pesos do algoritmo se tornem obsoletos para o tipo de sinal que os sensores estejam fornecendo.

O treinamento *on-line* proposto é executado como um processo paralelo à filtragem por blocos funcionais, coletando periodicamente, através do protocolo OPC, janelas de medidas dos sensores utilizadas como entradas para o treinamento do algoritmo FastICA realizado no supervisório. A matriz de pesos sinápticos, resultante desse treinamento, é instanciada no ambiente *Fieldbus* através protocolo

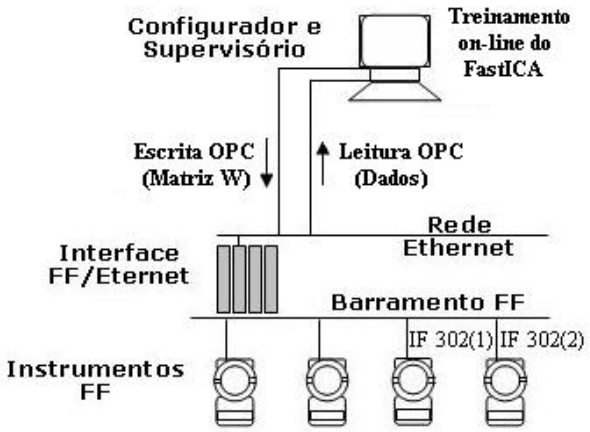

Figura 5.6: Treinamento on-line do algoritmo via protocolo OPC.

OPC e ao substituir pesos anteriores garante uma atualização periódica no processo de filtragem. A Figura 5.6 apresenta um diagrama de blocos dessa operação.

Essa abordagem é importante uma vez que os sinais resultantes dos sensores podem sofrer variações bruscas e a matriz de pesos sinápticos, em operação no momento, se tornar inaplicável para as novas medidas, prejudicando o processo de filtragem.

Capítulo 6

Testes e Resultados

Neste capítulo são apresentados os resultados dos experimentos escolhidos para validação da ferramenta proposta.

O trabalho propõe diferentes formas de inserção da técnica de filtragem, baseada em Análise de Componentes Independentes, em ambiente *Foundation Fieldbus*. Uma proposta emprega blocos funcionais FF para executar o algoritmo FastICA localmente nos sensores de campo, mantendo o padrão de interoperabilidade do sistema e visando uma forma robusta de filtragem de ruído. A outra, visa executar o algoritmo em tecnologia embarcada através de uma placa de DSP interligada com os instrumentos de campo *Foundation Fieldbus*.

A seguir serão detalhados os testes realizados com as diferentes propostas e os respectivos resultados alcançados.

6.1 Implementação em Placa DSP

A arquitetura utilizada no experimento para a implementação do algoritmo FastICA em placa DSP é apresentada na Figura 6.1.

O módulo de desenvolvimento TMS320lf2407 EVM conecta-se à rede *Fieldbus* através da interface conversora e dispositivos IF-302. Também, ligado ao módulo, através de uma placa de aquisição de dados, está o PC cuja finalidade é gerar e enviar sinais que simulam medidas ruidosas de sensores reais para que sejam processadas pelo algoritmo de filtragem instanciado na placa DSP.

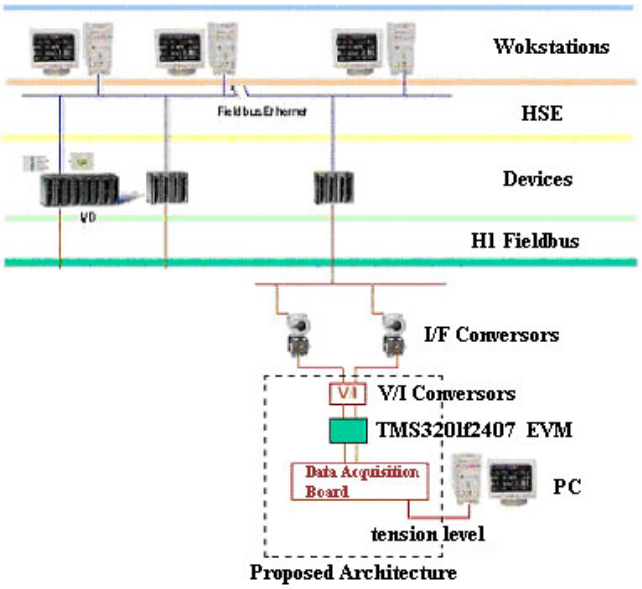

Figura 6.1: Arquitetura do Sistema.

Na simulação dos sinais ruidosos para a filtragem, arquivos de dados formatados, contendo amostras de uma senóide e um ruído uniformemente distribuído, ilustrados na Figura 6.2, foram gerados para realizarem o papel dos sinais fontes a serem recuperados.

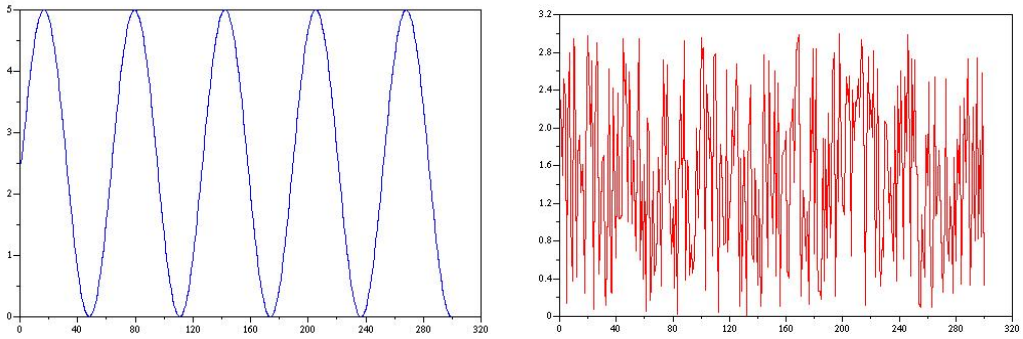

Figura 6.2: Sinais Originais.

Estes sinais foram misturados, aleatoriamente, como visto na Figura 6.3, através da matriz A, e utilizados para o treinamento *off-line* do algoritmo FastICA, como já detalhado no Capítulo 4.

$$A = \begin{bmatrix} -0.815345 & -0.91101 \\ 0.452653 & -0.43908 \end{bmatrix} \tag{6.1}$$

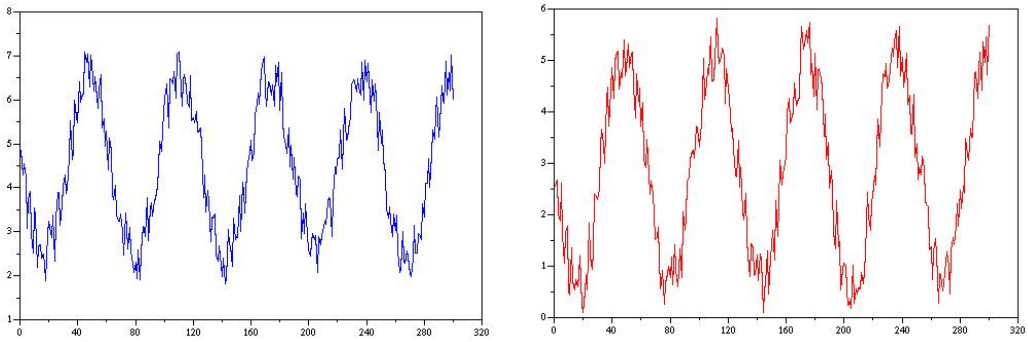

Figura 6.3: Sinais Misturados.

O treinamento resultou na matriz W de pesos sinápticos. Essa matriz foi, então, instanciada na placa DSP para que pudesse realizar a operação de separação das componentes misturadas.

$$W = \begin{bmatrix} 0.560568 & -0.603342 \\ 0.62634 & 0.58525 \end{bmatrix} \tag{6.2}$$

Através de um supervisório pôde-se visualizar, Figura 6.4, as saídas provenientes da implementação do FastICA como um dispositivo de software embarcado em um DSP, para as entradas geradas.

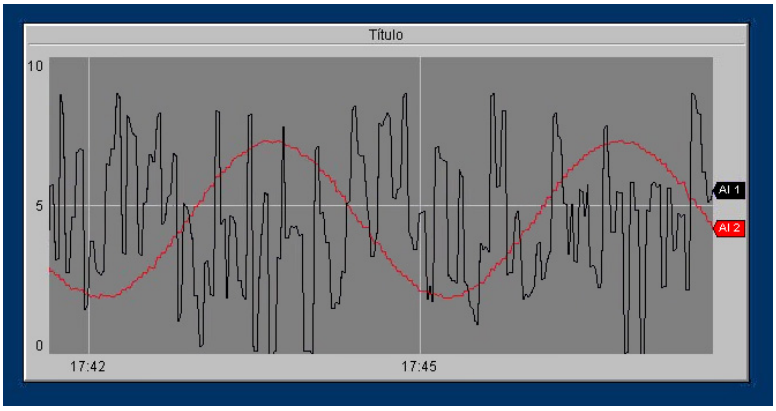

Figura 6.4: Sinais separados pelo algoritmo embarcado em DSP.

Pelo gráfico da Figura 6.4, percebe-se que o ruído, adicionado às senóides na entrada quase desaparece totalmente nas curvas de saída, verificando-se que essa

implementação apresentou bons resultados completando com êxito o processo de separação cega dos sinais.

6.2 Implementação em Blocos Funcionais FF

Neste experimento a filtragem é aplicada em medidas de dois sensores de temperatura, estrategicamente localizados em uma planta de medição de vazão e BSW (*Basic Sediments and Water*), e interligados através de uma rede *Foundation Fieldbus*, com o objetivo de extrair possíveis ruídos que venham a reduzir a confiabilidade da medida.

A planta onde são capturados os valores dos sensores localiza-se no Laboratório de Avaliação de Medição em Petróleo [Salazar et al., 2004] e tem como objetivo permitir a simulação de diferentes condições de operação dos medidores em campo, ou seja, simular misturas de água e óleo em proporções e vazões variadas. Para realizar estas simulações o laboratório possui 5 tanques: óleo, água, misturador, auditor (fundamental para a avaliação dos medidores de vazão e BSW), além de um tanque tratador para separação água/óleo, que possibilita a reutilização da água e do óleo em testes seguintes, sem a necessidade de descartes a cada teste.

A Figura 6.5 mostra a planta simplificada.

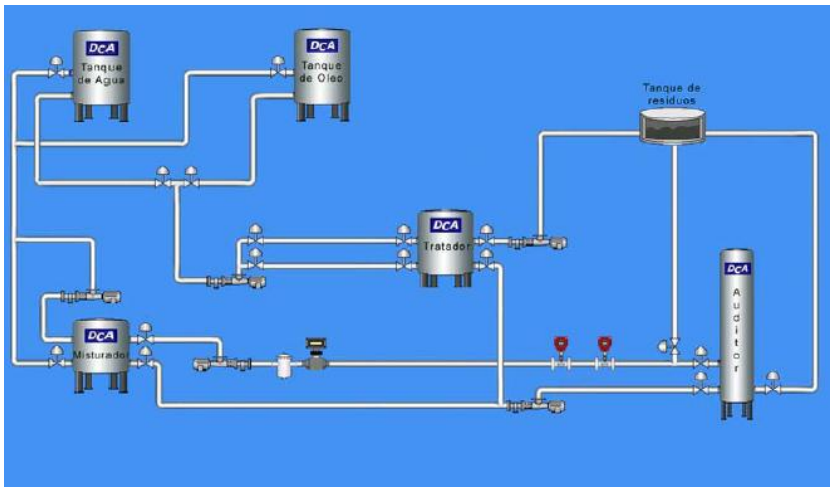

Figura 6.5: Planta de medição de Vazão e BSW

A grande maioria dos dispositivos de campo e o controlador que estão sendo utilizados neste processo utilizam tecnologia *Foundation Fieldbus*. Através do

protocolo OPC, informações de controle e supervisão são disponibilizadas ao supervisório.

Os medidores de temperatura utilizados, estão posicionados no tanque de água, e uma vez coletadas suas medidas elas são processadas pelo algoritmo inteligente.

A Figura 6.6 apresenta um diagrama de blocos do processo de implementação do algoritmo FastICA em blocos funcionais.

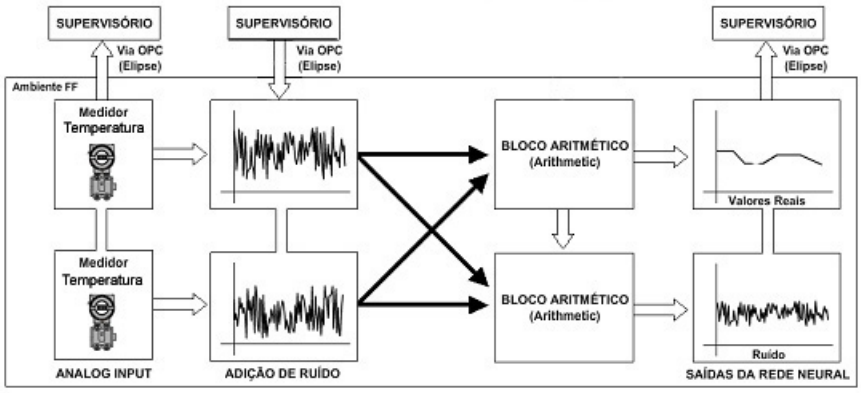

Figura 6.6: Diagrama de blocos da implementação do algoritmo de filtragem para medidas de sensores de temperatura.

Inicialmente foi realizada uma coleta de dados para que fosse formado o conjunto de treinamento do algoritmo. Esse conjunto foi composto por medidas ruidosas, referentes aos sinais dos dois sensores de temperatura, enviadas ao supervisório via OPC pelos transmissores de temperatura. Os ruídos adicionados às medidas foram simulados e enviados ao processo, também, via OPC. A adição de ruídos externos se fez necessária uma vez que as medidas de temperatura quase não apresentam variações, o que dificulta a ocorrência de ruídos, impossibilitando, assim, a aplicação da técnica de filtragem como desejado.

As Figuras 6.7 e 6.8 apresentam, respectivamente, os sinais de temperatura originais, coletados pelos sensores de temperatura, e esses mesmos sinais acrescidos de ruídos.

Com o treinamento off-line do algoritmo, os pesos sinápticos resultantes, matriz W_1, foram instanciados nos blocos funcionais e o resultado das operações desses blocos foram enviadas ao supervisório via OPC para análise e monitoramento.

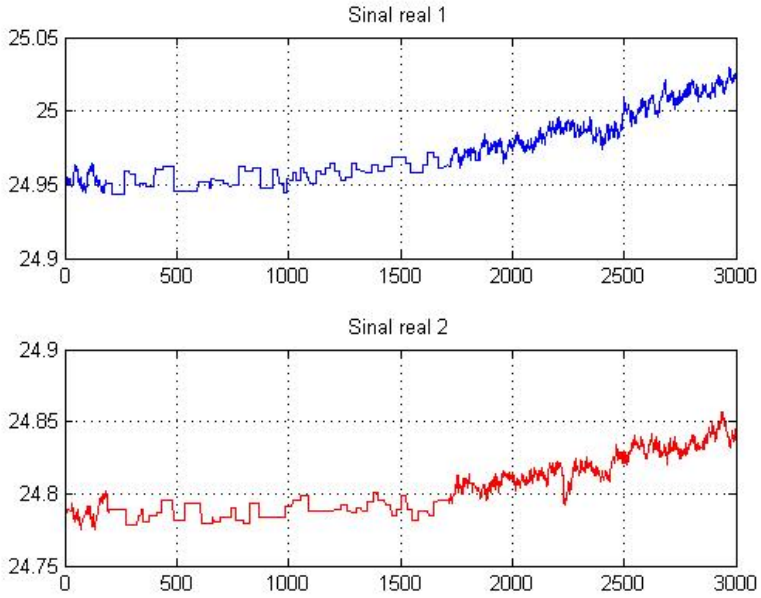

Figura 6.7: Sinais de temperaturas originais.

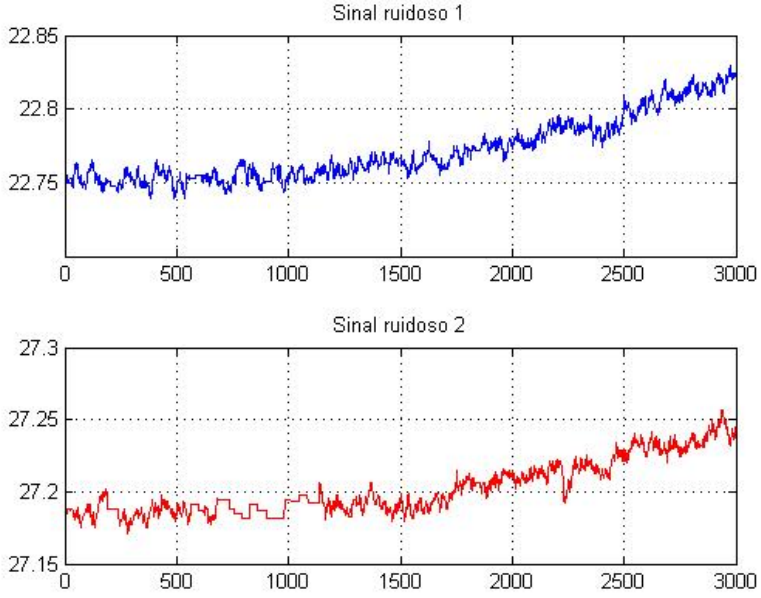

Figura 6.8: Sinais de temperaturas ruidosos.

$$W_1 = \begin{bmatrix} 0.6371 & 0.7708 \\ -0.7708 & 0.6371 \end{bmatrix} \tag{6.3}$$

Na Figura 6.9 é apresentado o esquema de ligação dos blocos funcionais, instanciados nos transmissores de temperatura, para o processo descrito acima.

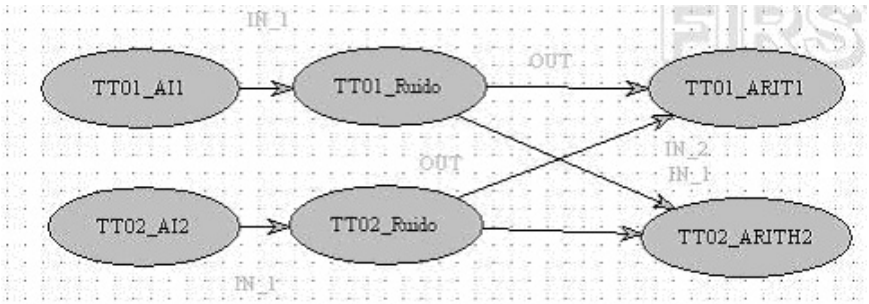

Figura 6.9: Modelo ICA através de blocos funcionais.

Os blocos AI, TT01_AI1 e TT02_AI2, são responsáveis por receber os sinais de temperatura capturados pelos sensores e enviá-los aos blocos aritméticos, TT01_Ruído e TT02_Ruído, para que sejam adicionadas, aos sensores, porcentagens diferentes de ruídos enviados via OPC pelo supervisório. Os blocos TT01_ARIT1 e TT02_ARITH2, são blocos aritméticos que representam os neurônios do algoritmo, e por isso, onde são instanciados os valores dos pesos sinápticos e realizada a multiplicação dos mesmos pelos sinais ruidosos. As saídas desses blocos são coletadas pelo supervisório e analisadas.

Cada bloco aritmético realiza as operações de multiplicação e soma correspondentes ao somatório dos produtos dos elementos da linha "i" da matriz de pesos sinápticos pelos elementos do vetor de sinais de entrada acrescidos dos ruídos, resultando no elemento da linha "i" do vetor dos sinais de saída.

O sinal de temperatura recuperado foi capturado pelo supervisório e é apresentado na Figura 6.10.

Observa-se que o sinal resultante do processo de filtragem recupera o valor original das medidas de temperatura, perdido com a adição dos ruídos.

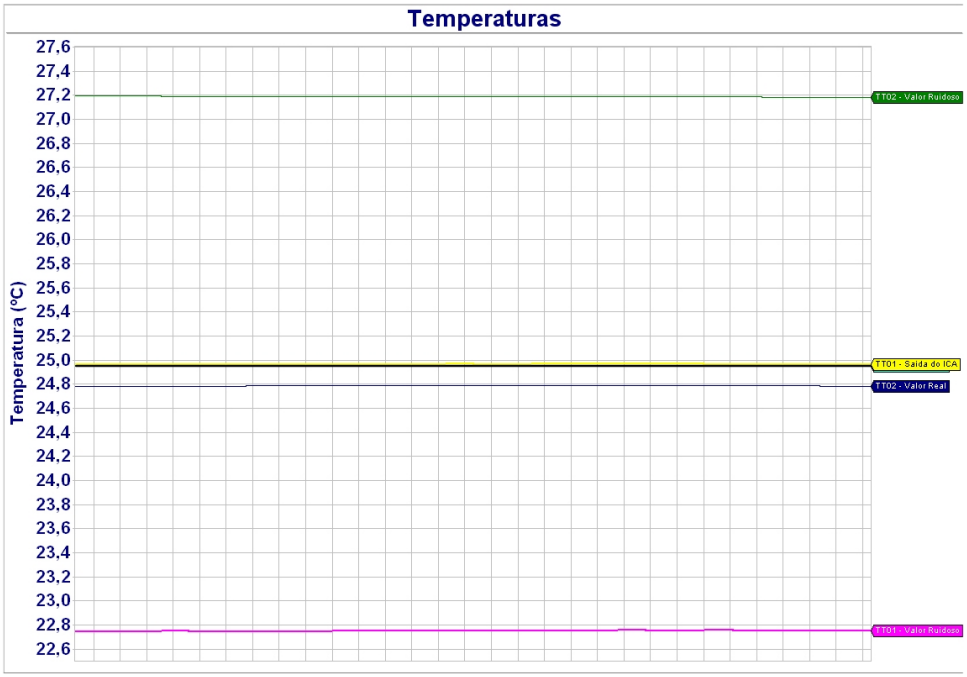

Figura 6.10: Rosa - Sinal ruidoso do sensor 1; Verde - Sinal ruidoso do sensor 2; Preto - Sinal original do sensor 1; Azul - Sinal original do sensor 2; Amarelo - Sinal de temperatura recuperado.

6.2.1 Treinamento *On-line* do Algoritmo FastICA Implementado em Bloco Funcionais FF

Os testes realizados utilizando o processo de treinamento *on-line* do algoritmo FastICA foram realizados seguindo o esquema da Figura 6.11.

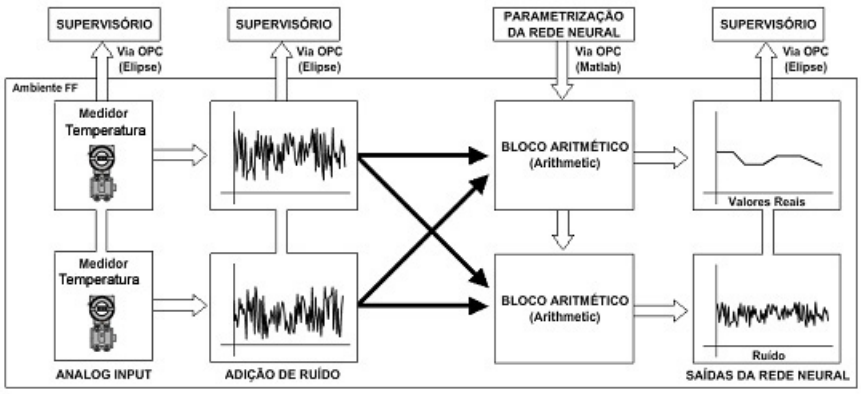

Figura 6.11: Treinamento *on-line* do algoritmo FastICA.

A janela de dados utilizada para os treinamentos foi de 2000 amostras. A

cada janela um treinamento era realizado e a matriz de pesos sinápticos, matriz W_2, resultante era instanciada, via OPC, nos blocos aritméticos TT01_ARIT1 e TT02_ARITH2.

$$W_2 = \begin{bmatrix} 0.6062 & -0.7953 \\ 0.7953 & 0.6062 \end{bmatrix} \tag{6.4}$$

A Figura 6.12 mostra os sinais capturados após o treinamento *on-line* do algoritmo.

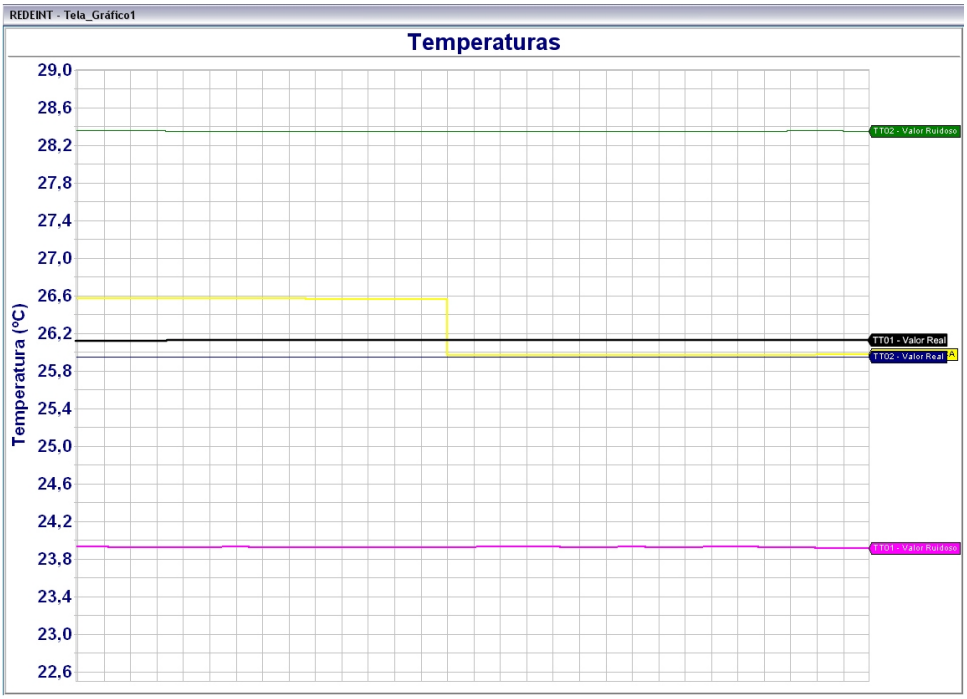

Figura 6.12: Rosa - Sinal ruidoso do sensor 1; Verde - Sinal ruidoso do sensor 2; Preto - Sinal original do sensor 1; Azul - Sinal original do sensor 2; Amarelo - Sinal de temperatura recuperado pelo treinamento *on-line* do algoritmo.

O degrau negativo no gráfico do sinal resultante do processo de filtragem, indica o momento exato do instanciamento dos novos valores dos pesos sinápticos nos blocos funcionais TT01_ARIT1 e TT02_ARITH2. Observa-se que o processo de filtragem recuperou o valor original das medidas de temperatura.

É importante realizar uma avaliação do desempenho de ambos os separadores e isso pode ser feito através de uma medida quantitativa do ruído usando um índice de rejeição global definido por [Amari, 1998]. Essa avaliação não foi realizada

nesse trabalho porém, com esta medida quantitativa é possível avaliar, por exemplo, quando o sensor necessita de calibração.

Capítulo 7

Conclusões

Nesta dissertação foram apresentados a descrição e o funcionamento de um sistema capaz de realizar separação cega de sinais utilizando um algoritmo inteligente baseado na técnica de Análise de Componentes Independentes, bem como sua implementação, interagindo diretamente com dispositivos de redes de campo *Foundation Fieldbus* através de blocos funcionais e tecnologia embarcada em DSP.

Os resultados obtidos foram satisfatórios, uma vez que a técnica de Análise de Componentes Independentes mostrou-se eficiente na separação cega de fontes e quando implementada nas abordagens propostas apresentou bons resultados em ambiente *Foundation Fieldbus*. A capacidade de extração de ruído apresentada por essa técnica viabiliza a implementação em redes de sensores sujeitos a ruídos.

Assim, pode-se citar como principais contribuições desta dissertação os seguintes pontos:

- utilização do método de filtragem baseado em Análise de Componentes Independentes em processos industriais aumentando a confiabilidade das medidas,
- construção do algoritmo inteligente de filtragem, FastICA, via DSP em ambiente FF,
- construção do algoritmo inteligente de filtragem, FastICA, via blocos funcionais padrão,
- treinamento *on-line* do algoritmo de filtragem FastICA em ambiente de rede

FF.

A estrutura fixa dos blocos funcionais padrão não permite a implementação de algoritmos mais complexos nos dispositivos de campo o que favorece à utilização do DSP já que este torna possível a implementação de aplicações ou algoritmos mais complexos no ambiente. A implementação do algoritmo FastICA em blocos funcionais, por sua vez, traz a grande vantagem da robustez em relação à implementação em DSP, uma vez que ao ser implementado em um dispositivo interno da rede, o sistema estará menos propício a adição de ruídos no processo.

Juntas, as abordagens de inserção do algoritmo de filtragem em ambiente FF trazem como vantagens a possibilidade de execução de algoritmos inteligentes em áreas que apresentam periculosidade de explosão; maior robustez a falhas ocorridas entre a ponte e o computador supervisório; possibilidade de processamento paralelo e distribuído.

Como perspectivas futuras é proposto que o bloco DSP com o algoritmo, associado aos conversores D/A e aos conversores tensão-corrente venham a se constituir em um núcleo de propriedade intelectual (IP *Core*) para embarque em sensores de rede *Foundation Fieldbus*. Como proposta de um primeiro protótipo fica a implementação da funcionalidade em FPGA (*Field Programmable Gate Array*) usando uma plataforma de desenvolvimento e simulação de componentes reconfiguráveis das famílias de dispositivos Altera ou Xilinx. Para tal, o estudo da sua arquitetura, do seu comportamento, da sua modelagem e das garantias de desempenho deverão ser exaustivamente avaliadas. Uma outra perspectiva é o estudo dos sinais resultantes da filtragem com o intuito de quantificar o ruído encontrado e identificar as condições dos sensores avaliados.

Referências Bibliográficas

[Amari, 1998] Amari, S. (1998). Superefficiency in blind source separation. *IEEE Transaction on Signal Processing*.

[Amari et al., 1997] Amari, S., Chen, T., and Cichocki, A. (1997). Stability analysis of adaptive blind source separation. *IEEE Transactions on Neural Networks*.

[Amari et al., 1996] Amari, S., Cichocki, A., and Yang, H. (1996). A new learning algorithm for blind source separation. *Advances in Neural Information Processing Systems*, 8.

[Ans et al., 1985] Ans, B., Hérault, J., and Jutten, C. (1985). Adaptive neural architectures: detection of primitives. *COGNITIVA'85*.

[Bell and Sejnowski, 1994] Bell, A. J. and Sejnowski, T. J. (1994). A non-linear information maximization algorithm that performs blind separation. *Advances in Neural Information Processing Systems*, 7.

[Bell and Sejnowski, 1995] Bell, A. J. and Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7.

[Berge, 2001] Berge, J. (2001). Fieldbuses for process control: Engineering, operation and maintenance, the instrumentation, systems and automation. *Society-ISA*.

[Burel, 1992] Burel, G. (1992). Blind separation of sources: A nonlinear neural algorithm. *Neural Networks*, 5.

[Cardoso, 1995] Cardoso, J. (1995). The invariant approach to source separation. *NOLTA'95*.

[Cardoso, 1989] Cardoso, J. F. (1989). Blind identification of independent signals. *Workshop on Higher-Order Spectral Analysis*.

[Cardoso and Souloumiac, 1993] Cardoso, J. F. and Souloumiac, A. (1993). Blind beamforming for non gaussian signals. *IEE Proceedings-F.*, 140 (6).

[Cichocki et al., 1998] Cichocki, A., Douglas, S. C., and Amari, S. (1998). Robust techniques for independent component analysis with noisy data. *Neurocomputing*, 22.

[Cichocki and Moszczynski, 1992] Cichocki, A. and Moszczynski, L. (1992). New learning algorithm for blind separation of sources. *Electronics Letters*, 28.

[Cichocki and Unbehauen, 1996] Cichocki, A. and Unbehauen, R. (1996). Robust neural networks with online learning for blind identification and blind separation of sources. *IEEE Trans. on Circuits and Systems*, 43.

[Cichocki et al., 1994] Cichocki, A., Unbehauen, R., and Rummert, E. (1994). Robust learning algorithm for blind separation of signals. *Electronics Letters*, 30.

[Comon, 1994] Comon, P. (1994). Independent coponent analysis - a new concept? *Signal Processing*, 36.

[Comon et al., 1991] Comon, P., Jutten, C., and Hérault, J. (1991). Blind separation of sources, part ii: Problem statement. *Signal Processing*, 24.

[Donoho, 1981] Donoho, D. L. (1981). On minimum entropy deconvolution. *Applied Time Series Analysis II, Academic Press*.

[E.Sorouchyari, 1991] E.Sorouchyari (1991). Blind separation of sources, part iii: stability analysis. *Signal Processing*, 24.

[FF-581F12, 2001] FF-581F12 (2001). Foundation specification - system architecture.

[FF-890F15, 2001] FF-890F15 (2001). Foundation specification - function block application process - part 1.

[FF-891F15, 2001] FF-891F15 (2001). Foundation specification - function block application process - part 2.

[FF-892F15, 2001] FF-892F15 (2001). Foundation specification - function block application process - part 3.

[Fonseca, 2002] Fonseca, M. O. (2002). Comunicação opc: Uma abordagem prática. *VI Seminário de Automação de Processos.*

[Fonseca and Seixas-Filho, 2005] Fonseca, M. O. and Seixas-Filho, C. (2005). Padrão opc: Aplicação e implementação. *ISA - The Instrumentation, Systems, and Automation Society / District 4 (South America).*

[Foundation, 2003] Foundation, F. (2003). *Foundation Fieldbus technical overview.* Bookman.

[Haykin, 2001] Haykin, S. (2001). *Redes Neurais: Princípios e Prática.* Bookman.

[Hérault and Ans, 1984] Hérault, J. and Ans, B. (1984). Circuits neuronaux à synapses modifiables: décodage de messages composites par apprentssage non supervisé. Master's thesis, C.-R. de l'Académie des Sciences.

[Hérault et al., 1985] Hérault, J., Jutten, C., and Ans, B. (1985). Détection de grandeurs primitives dans un message composite par une architecture de calcul neuromimétique en apprentissage non supervisé. *Actes du Xème colloque GRETSI.*

[Hyvarinen, 1999] Hyvarinen, A. (1999). Fast and robust fixed-point algorithm for independent component analysis. *IEEE. Transactions on Neural Networks*, 10.

[Hyvärinen, 1997] Hyvärinen, A. (1997). A family of fixed-point algorithms for independent component analysis. *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'97).*

[Hywarinen, 1999] Hywarinen, A. (1999). Fast and robust fixed point algorithms for independent component analysis. *IEEE Transactions on Neural Network.*

[Hywarinen and Oja, 1997] Hywarinen, A. and Oja, E. (1997). A fast fixed point algorithm for independent component analysis. *Neural Computation.*

[Hywarinen and Oja, 1999] Hywarinen, A. and Oja, E. (1999). Independent component of analysis: Algorithms and aplications.

[Jutten and Hérault, 1991] Jutten, C. and Hérault, J. (1991). Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24.

[Jutten and Taleb, 2000] Jutten, C. and Taleb, A. (2000). Source separation: From dusk till dawn. *Second International Workshop on Independent Component Analysis and Blind Source Separation (ICA2000).*

[Karhunen, 1996] Karhunen, J. (1996). Neural approaches to independent component analysis and source separation. *4th European Symposium on Artificial Neural Networks.*

[Karhunen and Joutsensalo, 1994] Karhunen, J. and Joutsensalo, J. (1994). Representation and separation of signals using nonlinear pca type learning. *Neural Networks*, 7.

[Lacoume and Ruiz, 1988] Lacoume, J. L. and Ruiz, P. (1988). Sources identification: A solution based on cumulants. *IEEE ASSP Workshop.*

[Nadal and Parga, 1994] Nadal, J. P. and Parga, N. (1994). Nonlinear neurons in the low noise limit: a factorial code maximizes information transfer. *Network: Computation in Neural Systems*, 5.

[Oja et al., 1991] Oja, E., Ogawa, H., and Wangviwattana, J. (1991). Learning in nonlinear constrained hebbian networks. *Int. Conf. Artificial Neural Networks (ICANN).*

[Salazar et al., 2004] Salazar, A. O., Silva, D. S., Quintaes, F. O., Rocha, A. M., Souza, A. J., Vilela, P. S. C., Maitelli, A. L., Lima, F. S., and Jesus, P. F. (2004). Evaluation of measurement processes of flow and bsw. *VI INDUSCON.*

[Shalvi and Weinstein, 1990] Shalvi, O. and Weinstein, E. (1990). New criteria for blind deconvolution of non-minimum phase systems (channels). *IEEE Transactions on Information Theory*, 36.

[Smaragdis, 1998] Smaragdis, P. (1998). Blind separation of convolved mixtures in the frequency domain. *International Workshop on Independence Artificial Neural Networks*.

[Tian et al., 2000] Tian, G. Y., Zhao, Z. X., and Baines, R. W. (2000). A fieldbus-based intelligent sensor. *Mechatronics*.

[Zheng and Nakagawa, 2002] Zheng, L. and Nakagawa, H. (2002). Opc (ole for process control) specification and its developments. *SICE Annual Conference 2002*.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo